UNIVERSIDAD DE CONCEPCIÓN
Facultad de Ingeniería
Departamento de Ingeniería Informática y
Ciencias de la Computación

Profesor Patrocinante:
Dr. Leo Ferres

Accesibilidad en expresiones matemáticas

José Sebastian Fuentes Sepúlveda

Informe de Memoria de Título
para optar al título de
Ingeniero Civil Informático

01 de abril de 2011

Esta memoria de título está dedicada a mis cinco tesoros: Constanza, Roberto, Sebastián, Leandro y Catalina. Que esto les ayude a recorrer sus propios caminos.

Agradecimientos

- Primero que todo, quiero darle las gracias a Dios. Sin importar si creo o no en Él, ni la manera en que lo haga, siempre estuvo acompañándome, ya que mi familia siempre lo quizó así.

- Agradecerle desde lo más profundo de mi ser a mis padres. Se esforzaron al máximo por darme lo necesario para lograr obtener el título, dejando de lado muchas cosas para asegurarme un buen pasar cada mes durante estos seis años. Su apoyo incondicional y su amor es lo que muchas veces me mantuvo de pie en los momentos difíciles.

- Mis hermanas, que en algunas ocasiones eran mis amigas y en otras mis mamás. De alguna manera, cada paso que daba ustedes también lo daban conmigo. Si yo crecía, ustedes también lo hacían junto a mí. Muchas gracias por ese apoyo que nunca sentí decaer, las amo.

- A mis cinco tesoros, los que me daban la fuerza cada día con esa fotografía que siempre estuvo en mi pared. Por ustedes he hecho todo esto y por ustedes seguiré adelante, abriéndoles camino. Cada vez que volvía a mi casa cansado después de varias semanas de estudio, ustedes eran los que me regresaban a la vida con sus juegos y alegrías.

- Al resto de mi familia, tíos, primos, abuelos, etc. A todos y cada uno de ustedes, en especial a aquellos que partieron durante el transcurso de esta carrera. Gracias por confiar en mí, por todos esas pequeñas ayudas, que en realidad significaban mucho para mí.

- Como dejar de nombrar a la que me acompañó practicamente los últimos 4 años de carrera, a mi polola. Sabemos que la vida de un universitario lejos de su casa es mucho más que sólo estudios, también están las penas y alegrías. Tu fuiste, eres y espero seas uno de los pilares más importantes en mi vida. Muchas gracias, de todo corazón, por levantarme cuando estaba deprimido, por liberarme cuando me quedaba atrapado en pensamientos tontos y por hacerme inmensamente feliz. Gracias a tí y a esa esperanza que nos ha acompañado en el último tiempo, llamada Helenita.

- A la familia de mi polola, quienes me aceptaron cuando aun era un estudiante de ingeniería y me apoyaron durante todo este tiempo.

- A mis socios de AlfaSeis, con quienes surgieron conversaciones muy interesantes y productivas (en otras ocasiones no tanto, pero no por eso menos entretenidas). En unos años más volveremos a surgir y seremos el AlfaSeis que siempre quisimos.

- A los demás miembros del laboratorio de inteligencia artificial. Hasta el día de hoy me siento muy cómodo en ese lugar, en el cual me gustaría permanecer por un largo tiempo y ayudar a que crezca cada día más, como debe ser.
- A mis compañeros de carrera, que me acompañaron en los ramos. Con quienes pasé noches enteras sin dormir y tardes eternas en la biblioteca resolviendo guías de cálculo, álgebra y una infinidad de materias que surgían de distintos puntos de la universidad.
- A todos mis amigos de otras carreras, a quienes fuí conociendo por los distintos rincones de la udec y de Concepción.
- A mi profesor guía, quien me demostró que el querer es poder. Sólo me resta decirle una sola palabra, *Namaste*.
- A los profesores que me hicieron clases durante mi carrera, en especial a los del departamento de ingeniería civil en informática. De cada uno aprendí un poco, que ahora forma parte de mi perfil de profesional.
- Y a cada persona que no nombre en esta lista, pero que fueron parte de este desafío. El nombrarlos conllevaría varias hojas más en este documento.

Resumen

En esta memoria de título se presenta el diseño e implementación de un prototipo, el cual intenta hacer accesible las expresiones matemáticas que se encuentran en la Web, a personas con problemas visuales. Para cumplir con esto, se hizo un estudio de las expresiones matemáticas más usadas en Internet, particularmente en Wikipedia. Además, obtener un conjunto de reglas para obtener las verbalizaciones de las expresiones matemáticas también fue necesario, el cual se vió reflejado en un grupo de templates, los que fueron utilizados en la generación de lenguaje natural basada en templates, para obtener las verbalizaciones de las distintas expresiones matemáticas.

También se puede encontrar una evaluación con personas, la que tiene como misión validar el diseño e implementación del prototipo propuesto.

Tabla de contenidos

Lista de tablas

Lista de figuras

1. Introducción

Cuando una persona con discapacidad visual quiere incorporarse a la población activa, debe enfrentarse a varias limitantes, en particular, en el mundo digital en el que estamos inmersos, ven disminuida su accesibilidad a grandes volúmenes de información. En algunos países ya existen iniciativas a nivel gubernamental para aminorar este problema, como es el caso de Estados Unidos y Canadá. En Estados Unidos existe la *Section 508*[1] que, a partir de 1998, obliga a las agencias federales a hacer accesible toda su información electrónica a todas las personas, sin discriminación. Con esto se busca fomentar el desarrollo de nuevas tecnologías que permitan que personas con discapacidades puedan acceder de forma fácil y sencilla a la información que deseen. Por su parte, Canadá en su Canadian Charter of Rights and Freedoms considera la sección *Equality Rights*[2], la cual intenta asegurar que toda persona, sin discriminación de ningún tipo, pueda gozar de sus derechos, entre ellos el derecho al acceso de la información. Un ejemplo de esto es *Statistics Canada*[3], la agencia del gobierno federal canadiense encargada de recoger y compilar datos estadísticos sobre Canadá y los canadienses, la cual ha hecho grandes esfuerzos para mejorar la accesibilidad de su información vía Web, proporcionando diversas formas de navegar su sitio.

Entre los distintos elementos que se pueden considerar como más complejos de hacer accesibles a personas con ceguera o adultos mayores, están las fórmulas matemáticas. El problema es que estas poseen una estructura no lineal y con un componente fuertemente visual. Por ejemplo, cuando tenemos 2 expresiones xy y x^y la posición de la variable y le da un significado distinto a cada expresión. Otra característica de las fórmulas o expresiones matemáticas es que pueden contener implícitamente información semántica, como es el caso de la famosa expresión $E = mc^2$, en la cual esta implícito que entre la variable m y c hay una operación de multiplicación. El problema con la información implícita es que no siempre se conoce, así una persona que no conozca *a priori* la expresión anterior podría pensar que mc es una sola variable.

Hoy en día, la solución más difundida para publicar fórmulas matemáticas en la Web es a través de imágenes raster. Para una persona ciega el contenido de una imagen raster es completamente inaccesible, inclusive si estuviesen representadas textualmente, por ejemplo, en la etiqueta *alt* de HTML.

[1] *Section 508*,Gobierno de Estados Unidos. 1998. `http://www.section508.gov`

[2] Canadian Charter of Rights and Freedoms, Section Equility Rights. 1982. `http://laws.justice.gc.ca/en/charter`

[3] *Statistics Canada*. `http://www.statcan.ca`

1.1. Motivación

1.1.1. Discapacidad en Chile

La encuesta CASEN (Caracterización Socioeconómica Nacional)[4], realizada por el Ministerio de Planificación, es una encuesta de hogares representativa a nivel nacional, regional, urbano y rural y comunal. La encuenta CASEN se ha aplicado desde el año 1985, siendo su última aplicación el año 2009. Su objetivo principal es elaborar diagnósticos de la realidad socioeconómica del país, evaluando los impactos de los programas sociales que promueve el gobierno.

La última encuesta CASEN que trata sobre discapacidad fue realizada en el año 2006[5]. En ella se encuestaron 73.720 hogares, equivalente a 268.873 personas, de 335 comunas de Chile y consideró como discapacidad sólo aquellas discapacidades de larga duración o severas. Uno de los resultados que arrojó fue que en Chile el 6,9 % del total de la población presenta alguna discapacidad, equivalente a 1.119.867 personas.

Se incluyeron dentro de la encuesta varios tipos de discapacidades, siendo los problemas visuales los que alcanzaron un mayor procentaje de personas, un 45,6 % del total de minusválidos, como muestra la figura 1.

Otro punto importante a destacar es que las personas minusválidas poseen, en general, un escaso nivel educacional, con un 43,1 % de discapacitados sin completar su educación básica. Como consecuencia de esto último, la tasa de participación laboral es baja y la tasa de desempleo es alta, respecto a las tasas correspondientes a personas sin discapacidad. Estos hechos explican en parte que la mayoría de la personas minusválidas están presentes en los deciles de menores ingresos, lo que complica aún más su incorporación a la población activa.

1.1.2. Ley 20.422

La situación actual en Chile en materia legal sobre la discapacidad se centra en la ley 20.422, que lleva por título *Establece normas sobre igualdad de oportunidades e inclusión social de personas con discapacidad*[6]. Esta ley fue presentada por el Ministerio de

[4]Encuesta de caracterización socioeconómica nacional, Ministerio de Planificación. 2010. `http://www.mideplan.cl/casen`

[5]Encuesta CASEN: Discapadidad, Ministerio de Planificación. 2006. `http://www.mideplan.cl/casen/publicaciones/2006/Resultados_Discapacidad_Casen_2006.pdf`

[6]Ley 20.422. Establece normas sobre igualdad de oportunidades e inclusión social de presonas con discapacidad, Ministerio de Planificación. 2010. `http://www.munitel.cl/Actualidad_Legislativa/Ley_20.422.pdf`

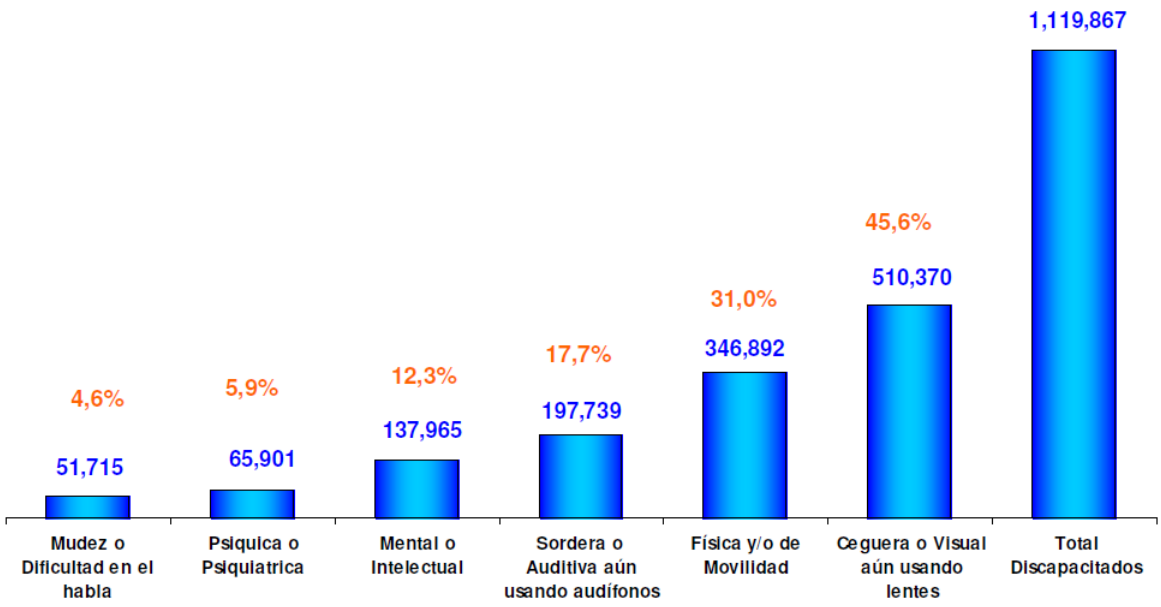

Figura 1: Distribución de discapacidades por tipo. Fuente: Encuesta CASEN 2006.

Planificación y está vigente desde el 10 de febrero del 2010. Su objetivo, presentado en el artículo 1^o, es "Asegurar el derecho a la igualdad de oportunidades de las personas con discapacidad, con el fin de obtener su plena inclusión social, asegurando el disfrute de sus derechos y eliminando cualquier forma de discriminación fundada en la discapacidad".

El principio de *Accesibilidad Universal*, descrito en el artículo 3^o de la ley, se ajusta muy bien al propósito de esta memoria de título, ya que busca que cualquier proceso, herramienta, productos y servicios, *etc.* puedan ser comprendidos, utilizados y practicados por todas las personas de la forma más cómoda y natural posible.

La ley también involucra *exigencias de accesibilidad* en el artículo 8^o, con lo cual exigirá que todos los bienes, procedimientos, productos, servicios, entre otros, *deben* ser accesibles. En el caso que esta condición no se cumpla, también contempla la realización de ajustes necesarios para que se tomen las medidas que garanticen la accesibilidad.

Un ejemplo de lo descrito en el párrafo anterior es, como dice el artículo 24, que "toda persona o institución, pública o privada, que ofrezca servicios educacionales, capacitación o empleo, exigiendo la rendición de exámenes u otros requisitos análogos, deberá realizar los ajustes necesarios para adecuar los mecanismos, procedimientos y prácticas de selección en todo cuanto requiera para resguardar la igualdad de oportunidades de las personas

con discapacidad que participan en ellos".

1.1.3. PSU y SIMCE

Ejemplos de aplicación de la ley 20.422 son la prueba de selección universitaria (PSU) y el sistema de medición de calidad de la educación (SIMCE). El SIMCE fue aplicado por segundo año consecutivo en 3 formatos especiales, Braille para los alumnos con discapacidad visual total, macrotipo (letra ampliada a Arial 24) para alumnos con ceguera parcial y para alumnos con discapacidad auditiva se entregaron las instrucciones en lenguaje de señas [7]. Por su parte la PSU, por primera vez, el año 2010 se va a dar en sistema Braille para que todos los jóvenes con discapacidad visual puedan darla en igualdad de condiciones con los otros jóvenes de Chile [8].

Debido a que las iniciativas para mejorar la accesibilidad para las personas con discapacidad aún siguen siendo insuficientes, es necesario trabajar en nuevos métodos para mejorar dicha accesibilidad. Por esta razón es que hacer accesibles las expresiones matemáticas, en forma hablada, para personas con problemas visuales apoya el cumplimiento de la ley 20.422, ayudando a los discapacitados visuales a desenvolverse mejor en su entorno. Que las matemáticas sean accesibles de manera hablada entrega una alternativa adicional para incorporarse en los métodos de enseñanza en los colegios y en evaluaciones como SIMCE y PSU.

1.2. Objetivos

1.2.1. Objetivo general

Diseñar e implementar un prototipo que apoye la accesibilidad de la personas con problemas visuales a expresiones matemáticas en la Web, utilizando un computador común y sin necesidad de instalar software especializado.

1.2.2. Objetivos específicos

- Analizar mecanismos que apoyen la accesibilidad de expresiones matemáticas para personas con problemas visuales.

[7] *Ministro Lavín inspecciona aplicación de Simce a alumnos con discapacidad visual.* Ministerio de educación. 20 de octubre de 2010. `http://www.mineduc.cl/index2.php?id_portal=1&id_seccion=10&id_contenido=12530`

[8] Joaquín Lavín, Ministro de Educación. *PSU de este año debuta con sistema Braille*, Diario La Nación. 21 de octubre de 2010. `http://www.lanacion.cl/psu-de-este-ano-debuta-con-sistema-braille/noticias/2010-10-20/233249.html`

- Investigar sobre algún lenguaje de representación de expresiones matemáticas que sirva como estándar de representación para el desarrollo de la memoria.
- Diseñar un prototipo para leer las expresiones matemáticas y llevarlas a lenguaje natural que pueda ser escuchado.
- Diseñar un esquema de navegación a través de las expresiones matemáticas utilizando teclas de un teclado de computador.
- Implementación de un prototipo del mecanismo propuesto para realizar las pruebas.
- Evaluar el prototipo.

1.3. Organización

El resto del documento se organizará de la siguiente manera: En la sección 2 se describirá algunas de las soluciones ya planteadas sobre este mismo tema, con el respectivo análisis para cada una de ellas. A continuación, en la sección 3, se explicarán algunos conceptos preliminares para poder comprender de mejor manera el resto del documento. En la sección 4 se presentarán las mejores reglas para leer expresiones matemáticas, acompañadas con el fundamento de su elección, para luego dar paso a la explicación de la implementación del prototipo propuesto, en la sección 5. Una vez explicada la implementación se dará a conocer la evaluación de la misma en la sección 6. Finalmente se dará termino con las conclusiones y el trabajo futuro en las secciones 7 y 8, respectivamente.

2. Discusión bibliográfica

Uno de los estándares más usados para la representación de fórmulas matemáticas es *MathML* [3]. *MathML* es un lenguaje de marcado basado en XML para expresiones matemáticas que entrega 3 formas distintas de representar una fórmula matemática, cada una enfocándose en una característica especial. *MathML* tiene un *marcado de presentación*, el que se encarga de la sintaxis de la fórmula, entregando la estructura de la expresión más cercana a como se leería. También cuenta con un *marcado de contenido*, no considerado en la creación inicial de MathML, el cual refleja la semántica de la fórmula. Como tercer elemento está el *marcado de interfaz*, que se encarga de la forma como el código MathML se podría incrustar en otros códigos, además de temas de visualización como fuentes, tamaño, color, etc. En la sección 4 se hablará más detalladamente sobre *MathML*.

Para apoyar iniciativas como las de Estados Unidos y Canada en materia de accesibilidad, han surgidos distintas propuestas para hacer accesibles las fórmulas matemáticas. Todas estas iniciativas se pueden clasificar en 2 grupos: los *enfoques estáticos* y los *enfoques dinámicos*. Un enfoque estático trata al documento que contiene la fórmula matemática como un ente pasivo, mientras que es el usuario el que coloca la parte activa. Por su parte, un enfoque dinámico le da mayor relevancia al documento que contiene la fórmula, el cual contiene una estructura que permite recorrerlo. Ambos enfoques son complementarios entre sí.

Entre los enfoques de tipo estático se encuentra el código Braille, el cual fue inventado por Louis Braille cerca de 1820. El código original consta de 6 puntos, permitiendo con todas sus combinaciones un alfabeto de 64 símbolos, sin embargo, este código no es lo suficientemente expresivo para codificar la sutileza representacional de las expresiones matemáticas. Como solución a esta limitación del código Braille es que surgió el código Nemeth [4], el cual es usado para material científico y matemático, que contiene símbolos que no están disponibles en el código Braille tradicional o literario. El código Nemeth no corresponde a una versión extendida del código Braille tradicional[9], sino que es un tipo especial de éste. Debido a que el código Braille y el Nemeth tienen una característica lineal para ser escritos, no pueden representar directamente las fórmulas matemáticas. Por ejemplo la expresión $\frac{x+1}{x-1}$ en código Nemeth sería representada por `?x+1/x-1#`, en donde `'?'` indica el inicio de la fracción, `'/'` separa el numerador del denominador y `'#'` indica el fin de la fracción. Uno de los problemas que tiene esta alternativa es que aumenta demasiado la carga cognitiva por parte del usuario del código, basta con imaginarse una expresión matemática más compleja para obtener una codificación muy larga, difícil de retener [5]. Como si esto fuera poco, el código Braille tiene distintas variaciones dependiendo del lugar geográfico en donde se esté utilizando, teniendo versiones francesas, italianas, inglesas,

[9]Unified 8 dot Braille Code 8 dot Braille Cell. `http://8dotbraille.com`

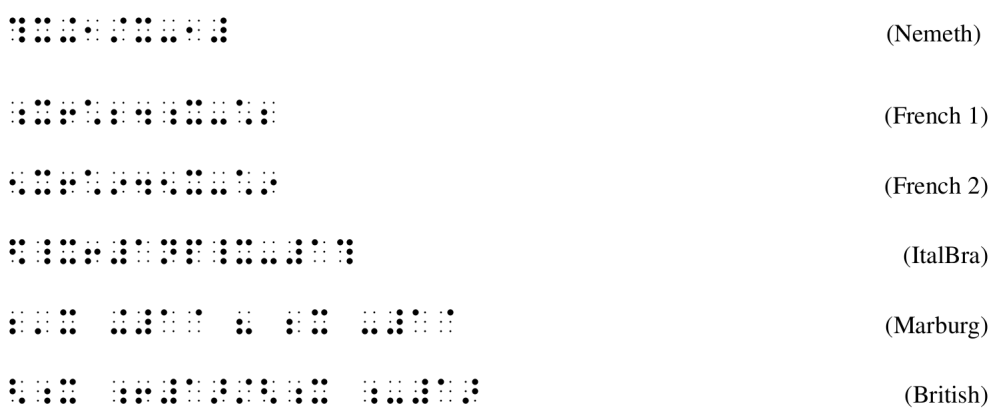

Figura 2: Distintas codificaciones de $\frac{x+1}{x-1}$ en código Braille [1].

etc. Por ejemplo, la misma fracción se codificaría en los formatos más populares como se muestra en la figura 2 [6].

Existen propuestas para traducir automáticamente expresiones matemáticas a código Braille o Nemeth, permitiendo de esta manera que documentos que ya están escritos en un lenguaje que no es el Braille, sean leídos por personas con problemas visuales, reduciendo el trabajo de crear 2 versiones del mismo documento (una para personas con discapacidades y otra para las que no tienen). La mayoría de las iniciativas realizan una conversión de código en LaTeX a código Braille, de esta manera expresiones matemáticas escritas en LaTeX pueden ser traducidas e impresas en código Braille o Nemeth, ejemplo de ello es Labradoor [7] y MathBraille [8]. Los problemas que presenta esta alternativa es que para hacer tangible el código Braille generado a partir del código LaTeX se necesitan aparatos especiales, en particular, una impresora que imprima código Braille y papel especial para la impresora, costos no menores considerando que una impresora Braille de bajo rendimiento cuesta cerca de 1.000.000 CLP[10] y una de alto rendimiento bordea los 4.000.000 CLP, algo que evidentemente atenta contra la accessibilidad de las personas con menos recursos. Otras iniciativas realizan la conversión en el sentido contrario, de Braille a LaTeX, tal es el caso del proyecto *Insight* y su versión para el sistema operativo Windows *Winsight* [9]. El proyecto *Insight* permite traducir expresiones en código Nemeth a LaTeX, tomando una imagen como entrada con el código Nemeth, creando en forma intermedia la versión ASCII del código para luego traducirlo a LaTeX.

Por otro lado, el proyecto *LAMBDA* [10], financiado por la Unión Europea, en el marco del programa IST (Information Society Technologies) utiliza un código que tiene una sin-

[10]CLP: Pesos Chilenos

taxis que se asemeja a la sintaxis de MathML, pues codifica las expresiones matemáticas a través de un lenguaje de marcado. Por ejemplo, para escribir una fracción se genera un código similar a `<inicio frac>numerador<simbolo frac>denominador<fin frac>`. Usar estos tags facilita la comprensión de las expresiones, ya que deja claro su semántica. El lenguaje de marcado que utiliza el proyecto *LAMBDA* tiene una representación directa en código Braille de 8 puntos, una extensión del código Braille tradicional de 6 puntos. No obstante, las expresiones matemáticas más complejas siguen siendo muy largas y demandan una carga cognitiva muy alta por parte de las personas que lo utilizarán.

Una iniciativa que busca hacer las matemáticas universalmente accesibles es la conocida como *sistema UMA (Universal Mathematics Accessibility)* [11]. La propuesta del sistema UMA permite la fácil traducción entre distintos formatos de codificación de fórmulas o expresiones matemáticas, tanto en formatos usados por no videntes como son Nemeth y Marburg (otra versión del código Braille dedicado a matemáticas, ver figura 2), como en formatos usados por personas videntes, por ejemplo MathML y LaTeX, como se muestra en la figura 3. El sistema consta de 2 subsistemas: el primer subsistema - Plataforma de interconversión - maneja todos los aspectos relacionados con la conversión entre los distintos formatos. Para realizar esta conversión se necesita de un formato de intercambio común, usado como puente entre todos los formatos, en este caso, este papel lo cumple el estándar OpenMath[11]. El segundo subsistema corresponde a la plataforma de navegación, la cual interactúa con el usuario, ya sea de una manera visual o auditiva.

Del lado de los enfoques dinámicos, las principales investigaciones se relacionan con matemáticas "habladas". Las investigaciones en matemáticas "habladas" utilizan una de 3 grandes estratégias para poder expresar las expresiones o fórmulas matemáticas: *indicadores léxicos*, *indicadores prosódicos* e *indicadores extralinguísticos*. Los primeros plasman en forma explícita la estructura de la expresión matemática a lenguaje natural, obteniendo resultados como " `inicio raíz cuadrada` x más uno `fin raíz cuadrada`"para la expresión $\sqrt{x+1}$ [12]. El problema con esta alternativa es la sobrecarga a la memoria de trabajo del "lector" u oyente que sería más adecuado. La segunda opción, usar indicadores prosódicos, modifica las características del lenguaje hablado, variando el tono, ritmo, tiempo, etc. Esto hace que el resultado sea un texto hablado que contiene pausas, elevación y disminución del tomo de voz, acercándose más a la forma como lo leería una persona. Así, la misma expresión de antes con esta opción sería "`(pausa al inicio)` raíz cuadrada de x más uno `(pausa al final)`", en este caso las pausas ayudan a que la expresión no se confunda con expresiones que pueden venir antes o después de ella [13]. La última alternativa, indicadores extralinguísticos, agrega elementos extralinguísticos, como el sonido, a la lectura de la fórmula para indicar el inicio y fin de alguna operación. De esta manera, el mismo ejemplo quedaría "`(sonido al inicio)` raíz cuadrada de x más

[11]OpenMath. 2006. `http://www.openmath.org`

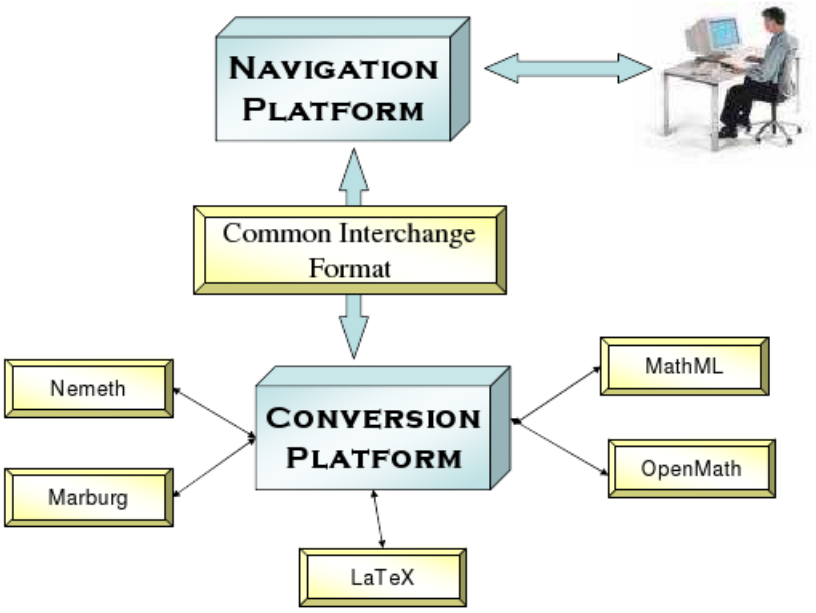

Figura 3: Organización del sistema *UMA* [1].

uno `(sonido al final)`".

Un trabajo que utiliza estas opciones es *AsTeR*, un sistema computacional para producir audio renderizado de documentos electrónicos [14]. Para ello, *AsTeR* toma documentos en LaTeX y genera archivos de audio a partir de ellos. *AsTeR* toma el documento en LaTeX generando una estructura interna en forma de árbol. Esta es mapeada posteriormente a audio a través de un lenguaje basado en reglas llamado *Audio Formatting Language (AFL)*, el cual va asignando a las hojas del árbol ciertos sonidos que permiten obtener un resultado final coherente al recorrer la estructura del árbol. Para representar ciertos hitos dentro de las expresiones matemáticas se utiliza modificaciones de la entonación e inflexiones en la voz. Por ejemplo, se aumenta el tono de la voz para indicar un super-índice y lo disminuye para indicar la presencia de un subíndice. Los riesgos que puede correr esta propuesta es que personas que no tengan un oído muy entrenado podrían no distinguir esas variaciones tan sutiles del sonido, generando confusión.

MathGenie es otro trabajo realizado en esta área. *MathGenie* es un programa que lee ecuaciones, destinado en especial a estudiantes de ciencias con problemas visuales, entregándoles una versión hablada de las expresiones matemáticas, además de código Nemeth refrescable [15]. Si bien *MathGenie* está orientado al uso de estudiantes y profesores, po-

dría ampliar su uso para permitir que las personas que no estén en esas categorías puedan mejorar su accesibilidad. *MathGenie* permite navegar las expresiones matemáticas a través de una simple combinación de teclas, en aparatos y software dedicados a ellos, como los *screen reader* y los *dispositivos de Braille actualizable*. El procedimiento que realiza *MathGenie* comienza recibiendo como entrada código en MathML (presentación) y continúa entregando como salida el registro de voz y código Nemeth equivalentes a la entrada, a través de la dispositivo destinado para ello. Es una de las alternativas más utilizadas, en especial en educación, sin embargo, para su buen funcionamiento necesita de los dispositivos de Braille actualizable, los que si bien prestan un gran servicio, no están disponibles para todas las personas en los lugares en que lo necesiten, como sus hogares. En el caso de los screen readers, necesitan ser instalados, lo cual tiene el inconveniente que al cambiarse de computador no se tiene acceso a la instalación previa, debiendo volver a instalarlo.

Otro enfoque [16] sugiere escuchar y navegar las fórmulas matemáticas utilizando comandos dados con la voz, dejando de lado el teclado. Para ello se traslada expresiones codificadas en MathML a voiceXML[12]. VoiceXML es un lenguaje de marcado basado en XML que sirve para crear audio o voz basados en documentos. Así como un navegador Web renderiza documentos HTML, un intérprete de voiceXML renderiza documentos voiceXML generando audio. La transformación de MathML a voiceXML se realiza a través de XSLT, un lenguaje de transformación para XML[13]. El enfoque extiende el uso de voiceXML, agregándole algunos *voice anchors* al documento, lo que permite ir navegando las expresiones con la simple pronunciación de comandos.

MathPlayer [17] por su parte fue propuesto inicialmente como un plugin para navegadores Web, para permitir la visualización de expresiones matemáticas escritas en MathML. Con el tiempo se fue mejorando incluyendo reglas del habla basadas en indicadores léxicos, intentando mejorar las desventajas que tiene el uso de estos indicadores, simplificando el resultado en algunos casos, por ejemplo, para la expresión $\frac{a}{b}$ genera la expresión "`inicio frac` a `símbolo frac` b `fin frac`" y luego la simplifica a "a `sobre` b". Sin embargo, la inclusión de las reglas del habla aún es demasiado básico como para ser considerado una solución definitiva[14]. Otro problema que surge es que MathPlayer está hecho particularmente para el sistema operativo Windows y aunque han surgido versiones para Mac OS, está limitado a estos ambientes. Además, MathPlayer está hecho para integrarse con screen readers, programas especiales para personas discapacidad visual, pero no disponible en todos lados.

[12]Voice Extensible Markup Language (VoiceXML) Versión 2.0. 2004. `http://www.w3.org/TR/voicexml20`

[13]XSL Transformations (XSLT) Versión 1.0. 1999. `http://www.w3.org/TR/xslt`

[14]MathPlayer. Sección MathML technology. `http://www.dessci.com/en/products/mathplayer`

Uno de los trabajos más recientes [18] propone un framework que permite navegar expresiones matemáticas a través de 2 modalidades distintas: *presentación pasiva* y *presentación activa*. La presentación pasiva se refiere a la lectura lineal de las expresiones matemáticas, de izquierda a derecha, por su parte la presentación activa permite navegar las expresiones de una manera jerárquica, utilizando un árbol que representa las fórmulas. Esta investigación se apoya en el plugin para el navegador Web Firefox llamado *FireVox*[15], por lo que está disponible para ser utilizado en distintas plataformas, como Windows, Linux y Mac. Uno de los inconvenientes que presenta este trabajo es que FireVox dejó de actualizarse el año 2008, lo que lo hace incompatible con versiones más recientes del navegador Firefox, desde la versión 3.6 en adelante. Otro problema que presenta es que la implementación no está disponible para probarla en versiones más antiguas del navegador. Además no existe un listado de las reglas en las que se basa la investigación, salvo un solo ejemplo que aparece en el paper, el cual no es lo suficientemente concluyente. El framework se basa en MathML para hacer el renderizado de las expresiones matemáticas, sin embargo, se utiliza sólo la parte de presentación, perdiendo toda la semántica que entrega la parte de contenido de MathML. La evaluación que se realiza del framework no considera pruebas con personas, basándose en el documento [18], lo que le quita un poco de validez a la hora de calificarlo y compararlo con otras propuestas.

Muy pocas de las propuestas analizadas hasta ahora han considerado el uso de la Web como un gran medio de difusión que tiene problemas de accesibilidad para personas con problemas visuales, en especial cuando se trata de objetos complejos como fórmulas matemáticas o gráficos. Considerando que la Web es una de las más grandes revoluciones de la humanidad y que la mayoría de los contenidos a los que acceden las personas están ahí, debe ser un aspecto a abordar para mejorar la accesibilidad. Además, el uso de aparatos especiales para muchas de las propuestas hace, que en ocasiones, varias buenas ideas no puedan ser difundidas masivamente, es por ello que se hace necesario buscar un nuevo método que no requiera de elementos difíciles de conseguir, idealmente que necesite sólo un computador común, para mejorar la accesibilidad de la personas, a información digital, con problemas visuales y adultos mayores.

[15]Fire Vox: A Screen Reading Extension for Firefox. `http://www.firevox.clcworld.net`

3. Preliminares

3.1. MathML

MathML es una aplicación XML para describir notación matemática, capturando tanto la estructura y como el contenido [3].

MathML entrega la base para desarrollar software como editores o renderizadores de expresiones matemáticas, además de aplicaciones para procesar fórmulas matemáticas, tomando MathML como un protocolo de entrada y/o salida. La principal ventaja, para este trabajo, que distingue a MathML de otras propuestas de estándares para la representación de expresiones matemáticas, como OpenMath, es que permite representar y procesar expresiones matemáticas en la Web. Navegadores Web como Firefox y Safari soportan de manera nativa contenido codificado en MathML. Por su parte, navegadores como Chrome y Explorer sólo necesitan la instalación de complementos para soportar expresiones en MathML.

El elemento raíz de una expresión codificada con MathML es `<math>`, el cual encapsula cada instancia de MathML dentro de un documento. Para que expresiones en MathML sean válidas se debe incluir el namespace identificado por la URI `http://www.w3.org/1998/Math/MathML`. Una vista usual de instancias de expresiones en MathML sería la siguiente:

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  ...
</math>
```

Las ideas matemáticas pueden existir independientemente de la notación que las representan. Sin embargo, la relación entre el significado y la notación es sutil, y parte del poder de la matemáticas para describir y analizar deriva de la posibilidad de representar y manipular ideas en forma simbólica [3]. Atendiendo esto es que MathML codifica tanto la estructura como el contenido de las expresiones, a través del *marcado de presentación*, usado para desplegar expresiones matemáticas, y del *marcado de contenido*, usado para representar el significado de las expresiones matemáticas.

3.1.1. Marcado de presentación

El *marcado de presentación* tiene como objetivo principal describir el layout de las expresiones matemáticas. En la última versión de MathML se entregan 37 tags distintos para poder representar la estructura de las expresiones. Dentro de los tags más usados se encuentran el tag `<mi>` para representar variables, nombres de funciones y constantes,

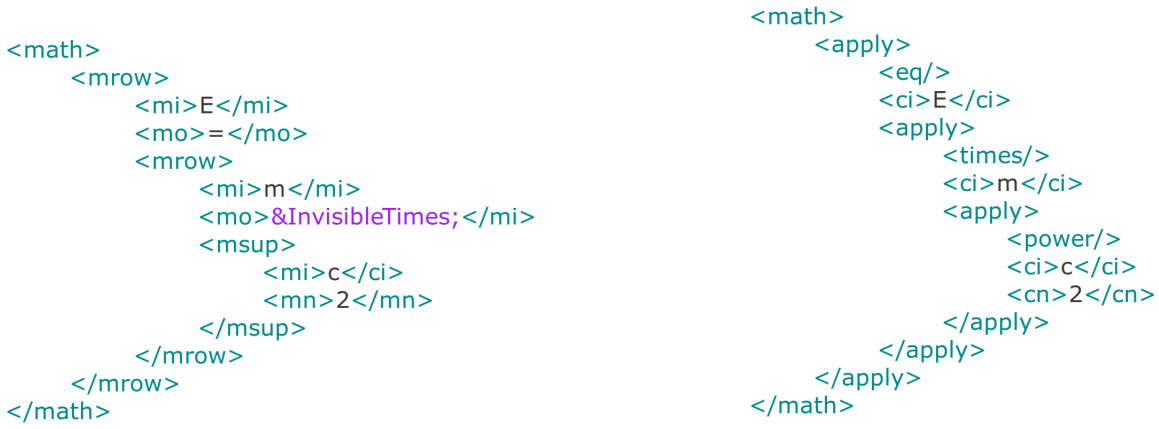

Figura 4: Marcado de presentación (a) y contenido (b) en MathML para la expresión $E = mc^2$

`<mo>` para representar operadores, `<mn>` representando números, `<mrow>` usado para agrupar otros tags, `<msub>` y `<msup>` para indicar subíndices y superíndices, respectivamente, y `<mfrac>` para codificar fracciones.

Un ejemplo del *marcado de presentación* para la expresión $E = mc^2$ está en la figura 4a. Algo interesante a destacar de la figura es el uso de `&InvisibleTimes;`, el cual representa al operador de la multiplicación, que está implícitamente en la ecuación, pero que no se refleja de manera explícita.

3.1.2. Marcado de contenido

El *marcado de contenido* entrega una manera explícita de codificar el significado o semántica de la expresiones matemáticas, de manera no ambigua. Consta de 129 tags para codificar las expresiones, de los cuales los tags `<apply>`, `<ci>` y `<cn>` son los más comúnmente usados. El tag `<apply>` es usado para representar la aplicación de una función o un operador sobre sus argumentos. Por su parte, el tag `<ci>` se utiliza para codificar variables y el tag `<cn>` hace algo similar, pero con los números.

Para la misma ecuación $E = mc^2$, su representación con *marcado de contenido* está en la figura 4b. En la figura se puede apreciar que la manera de aplicar el tag `<apply>` es indicando primero cuál será el operador a aplicar y a continuación indicando cuáles serán los operandos sobre los cuales actuará el operador.

3.1.3. La decisión de usar marcado de contenido para representar las entradas

La representación de las expresiones matemáticas que entrega el *marcado de presentación* es lo más cercano a la manera a como se leería. Sin embargo, el *marcado de presentación* no es suficiente para construir un software de verbalización de expresiones matemáticas, ya que en ocasiones es ambiguo. Esto se debe a que la notación matemática, a pesar de ser más rigurosa que el lenguaje natural, es algunas veces ambigua, dependiente del contexto y varía según área del conocimiento, localización, etc. Por ejemplo, para la expresión f^{-1} se pueden obtener al menos 2 interpretaciones, la función inversa de f o la variable f elevado a -1, las que con el *marcado de presentación* se codificarían como se muestra en la figura 5. Como se puede apreciar, en ambos casos se utiliza como elemento principal el tag `<msup>`, por lo que se puede concluir que no existe ningún elemento significativo que permita hacer la distinción entre la función inversa y la potencia, ya que los otros elementos son comunes para cualquier fórmula.

Usando el *marcado de contenido* se eliminan los problemas de ambigüedad, debido a que es la semántica la que se codifica. Sin embargo, el traslado de expresiones con el *marcado de contenido* a lenguaje natural no es directo. La codificación en *marcado de contenido* para la expresión anterior con sus 2 interpretaciones se muestra en la figura 6. En ella se puede apreciar claramente que el tag `</inverse>` identifica a la función inversa y el tag `</power>` caracteriza a la potencia.

Otra ventaja que posee el *marcado de contenido* por sobre el *marcado de presentación* es que existen herramientas que permiten el paso de contenido a presentación, por ejemplo [19]. En cambio, debido a los problemas de ambigüedad, no existen herramientas para la conversión de presentación a contenido.

Para el desarrollo de esta memoria de título se optó por el *marcado de contenido* para convertirlo a lenguaje natural, debido a que el problema de ambigüedad que presenta el *marcado de presentación* es más complicado de solucionar que convertir *marcado de contenido* en lenguaje natural.

Además de esto, en [20] se demuestra que XML, ya sea como mecanismo de transformación o como medio para la representación de los datos, puede ser usado en generación de lenguaje natural, lo que viene a apoyar aún más la elección de *MathML* como medio de representación de las expresiones matemáticas.

```
<math>
    <msup>
        <mi>f</mi>
        <mrow>
            <mo>(</mo>
            <mn>-1</mn>
            <mo>)</mo>
        </mrow>
    </msup>
</math>
```

(a)

```
<math>
    <msup>
        <mi>f</mi>
        <mn>-1</mn>
    </msup>
</math>
```

(b)

Figura 5: Marcado de presentación para f^{-1} (inversa) (a) y f^{-1} (potencia) (b)

```
<math>
    <apply>
        <inverse/>
        <ci> f </ci>
    </apply>
</math>
```

(a)

```
<math>
    <apply>
        <power/>
        <ci>f</ci>
        <cn>-1</cn>
    </apply>
</math>
```

(b)

Figura 6: Marcado de contenido para f^{-1} (inversa) (a) y f^{-1} (potencia) (b)

3.2. Fórmulas matemáticas en la Web

Las fórmulas matemáticas en la Web, por lo general, se publican en forma de imagen. En el caso de Wikipedia[16], uno de los repositorios de información más grandes que existen, además de mostrar la imagen que representa a cada fórmula matemática, se entrega su codificación LaTeX, usando el atributo `alt` del tag `<img/>` de HTML[17]. Un ejemplo[18] de esto se puede ver en la ecuación 1:

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{1}$$

que es codificada con las siguientes líneas HTML:

```
<img
  class="tex"
  src="image.png"
  alt="\gamma =\frac{1}{\sqrt{1-\frac{v^2}{c^2}}}"
/>
```

[16]Wikipedia: La enciclopedia libre. 2010. `http://www.wikipedia.org/`
[17]HTML: HyperText Markup Language. 2010. `http://www.w3.org/html/`
[18]http://es.wikipedia.org/wiki/Teoría_de_la_relatividad_especial

Wikipedia además puede ser descargada en formato XML[19]. En este formato las expresiones matemáticas están marcadas por el tag <*math*>, codificadas en formato LaTeX. Aprovechando esto, se decidió contar todas las fórmulas matemáticas que están en Wikipedia y luego obtener cuáles son los operadores o símbolos matemáticos más usados. Con este objetivo se desarrolló un script, el cual extraía las expresiones matemáticas de Wikipedia, pudiendo extraer 355.684 expresiones matemáticas, repartidas en 26.174 artículos de Wikipedia. Posteriormente se contaron cuáles eran los operadores más utilizados, llegando a la tabla 1.

En la tabla 1, se entrega en la primera columna el nombre de las palabras claves de LaTeX más usadas en fórmulas matemáticas, entre los que se encuentran principalmente operadores matemáticos. La segunda columna muestra el número total de apariciones de cada palabra clave, mientras que la tercera columna entrega la cantidad de fórmulas en la que aparece cada palabra clave de LaTeX. Finalmente, la cuarta columna muestra un ejemplo de aplicación de cada palabra clave relacionada con fórmulas matemáticas. Se omitieron de la tabla las operaciones básicas (adición, substracción, multiplicación y división), letras griegas y símbolos matemáticos especiales como ∞, pues son parte del uso más común de las expresiones matemáticas. Como casos especiales, las palabras clave `frac` y `over` fueron consideradas como una sola, ya que ambas son utilizadas para representar fracciones. La misma situación ocurrió con `rightarrow - to`, `leq - le` y `bar - overline`. Como se aprecia en la tabla, los subíndices son los más utilizados en las expresiones matemáticas, seguidos de los superíndices y/o potencias. Se puede apreciar una baja notable en la cantidad de apariciones al llegar a la tercera mayoría, las fracciones. Una baja importante, aunque más pequeña que la anterior, se aprecia en la cuarta mayoría, raices.

En la figura 7 se muestra la correlación entre la cantidad total de apariciones de cada palabra clave de LaTeX contra la cantidad de fórmulas en las que aparece cada palabra clave. En la figura de puede apreciar una gran pendiente en los puntos que involucran a subíndices, superíndices y fracciones en ambas líneas, lo que quiere decir que son los más repetidos y los más usados en las fórmulas matemáticas de Wikipedia. El coeficiente de correlación de Pearson entre ambas curvas o series de datos tiene un valor de 0.99, lo que muestra que están muy correlacionados el número de apariciones de cada palabra clave de LaTeX y la cantidad de fórmulas en la que cada palabra es utilizada.

Lamentablemente, no todas la palabras clave u operadores de la tabla 1 tienen su correspondiente representación en *content MathML*. Palabras clave como `_`(subíndice), `\choose` (coeficiente binomial), `\ldots` (elipsis) y `\pm` ($\pm$) no pueden ser representados

[19]English Wikipedia dumps in SQL and XML. 2010. `http://download.wikimedia.org/enwiki/`

Palabra clave	Total apariciones	Total fórmulas	Ejemplo
_ (subíndice)	287168	132419	a_{n+2}
^(superíndice)	167479	87456	$(1+x)^{k+1}$
frac + over	77425	48763	$\frac{z-1}{z+1}$
sqrt	14647	11279	$\sqrt[9]{3/2}$
partial	14576	5067	$\frac{\partial f}{\partial x}$
rightarrow + to	13316	11057	$h \to 0$
sum	13225	10754	$\sum_{i=1}^{n} w_i$
in	12316	10683	$x \in X$
int	9961	7629	$\int_0^{2\pi} f(x)$
times	8428	6134	$X \times Y$
leq + le	7327	5432	$\|z+w\| \leq \|z\| + \|w\|$
bar + overline	6555	4666	$\bar{a}$
cos	6091	4371	$\cos(x+y)$
sin	6083	4380	$\sin\theta$
hat	5866	3808	$\hat{c}$
vec	4600	2578	$\vec{x}$
ldots	4463	3638	$1, 2, 3, \ldots, 100$
cdots	4211	3426	$1 + r + r^2 + r^3 + \cdots + r^n$
log	4187	3047	$\log_{10} 1000 + 1 = 3 + 1$
dots	3629	2927	$x_1, x_2, x_3, \ldots, x_n$
ln	3271	2336	$\ln x - 4$
otimes	2552	1695	$X \otimes Y$
lim	2464	2126	$\lim_{i\to\infty} \mu(E_i)$
choose	2361	1269	$\binom{n}{k}$
approx	2326	2200	$5\sqrt{3}a^2 \approx 8{,}66025404a^2$
dot	2308	1664	$x\dot{y} - y\dot{x}$
forall	2271	1809	$\forall x : \phi(x)$
circ	2153	1608	$h \circ (g \circ f) = (h \circ g) \circ f$
pm	2026	1561	$a \pm b$
equiv	2001	1779	$n \equiv 2, 3$
geq	1874	1724	$x_{n_2} \geq x_{n_1}$

Tabla 1: Palabras clave de LaTeX más usadas en Wikipedia para representar expresiones matemáticas

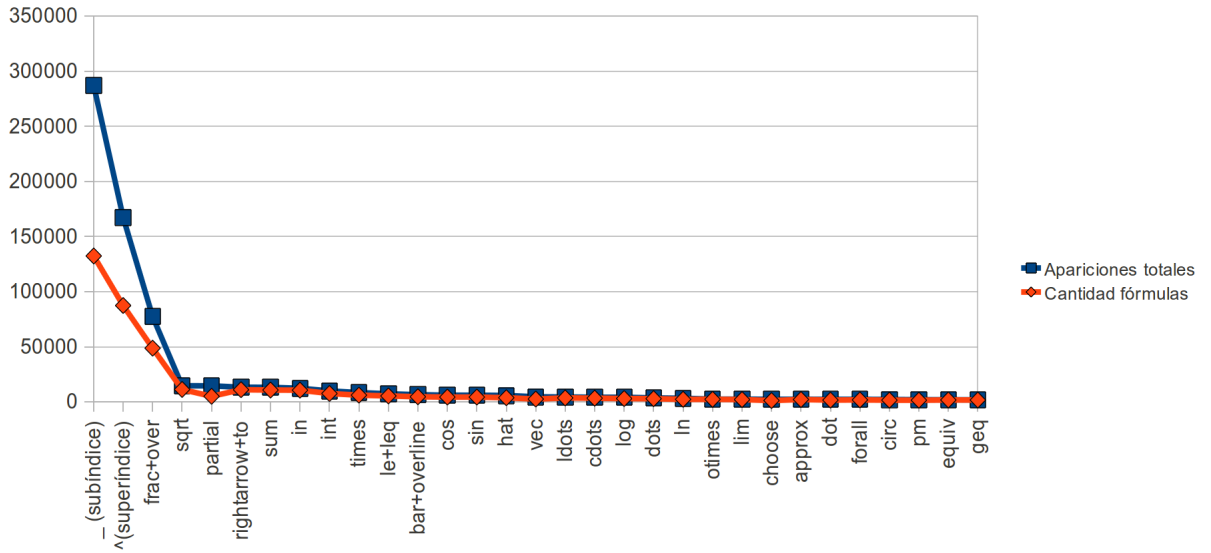

Figura 7: Correlación entre el total de apariciones de las palabras claves de LaTeX versus la cantidad total de fórmulas en las que aparece cada palabra clave

a través de *content MathML*. En la tabla 2 se muestran las palabras clave de LaTeX de la tabla 1, usadas para representar expresiones matemáticas, y su equivalente en *content MathML*.

3.3. WebAnywhere

Los *screen readers*, software que permiten conocer el contenido que se muestra en la pantalla de un computador en forma auditiva, son de gran utilidad para personas que tienen alguna discapacidad visual. Lamentablemente, el costo de estos software es muy elevado, cercado a los 1000 dólares por cada instalación, debido a su complejidad, mercado relativamente pequeño y altos costos de soporte [2]. El alto costo hace que los *screen readers* no estén al alcance de la mayoría de las personas y que las instituciones que pueden adquirirlos sólo los instalen en un número reducido de computadores, limitando las opciones de personas con discapacidad visual de acceder a uno de estos equipos. Otras alternativas de *screen readers* son portables, de manera tal que no estén atadas a la instalación en un sólo computador, sin embargo, no siempre se cuenta con permisos para instalar nuevos programas en todos los equipos, en especial en computadores públicos, como los que están en bibliotecas, laboratorios, etc. Un buen *screen reader* debería tener un bajo costo, para que pueda ser usado por la mayoría de las personas con problemas visuales, y no necesitar instalación, para que pueda ser utilizado en cualquier computador.

La figura 8 muestra la oferta de *screen readers* según su portabilidad y costo por

LaTeX	Content MathML	LaTeX	Content MathML
^	<power/>	sin	<sin/>
frac + over	<divide/>	log	<log/>
sqrt	<root/>	ln	<ln/>
partial	<partialdiff/>	otimes	<outerproduct/>
rightarrow + to	<tendsto/>	lim	<limit/>
sum	<sum/>	approx	<approx/>
in	<in/>	forall	<forall/>
int	<int/>	circ	<compose/>
times	<cartesianproduct/>	equiv	<equivalent/>
leq + le	<leq/>	geq	</geq>
cos	<cos/>		

Tabla 2: Palabras clave de LaTeX y su representación en Content MathML

usuario. En la gráfica se observan tres categorías de dispositivos que apoyan la accesibilidad según su portabilidad: software, dispositivos móviles y los que pueden ser usados en cualquier computador. El primer grupo presenta tanto alternativas de bajo costo, como *FireVox*, como de un costo mayor, como es el caso de *JAWS* y *Windows Eyes*. Este grupo es el que presenta la menor portabilidad de todos. En el grupo de los dispositivos móviles se pueden encontrar opciones con una mayor portabilidad que el primer grupo, sin embargo es donde se encuentran los dispositivos de mayor costo, pues se tratan de hardware. Entre las soluciones planteadas están los computadores portátiles con algún *screen reader* instalado, teléfonos móviles y la más costosa de todas, PDA basadas en Braille. Finalmente, el tercer grupo provee de alternativas muy portables y baratas. Dentro de ellas, la que más se ajusta al objetivo de esta memoria es *WebAnywhere*, pues no requiere instalación en los equipos utilizados por las personas con problemas visuales, además de ser gratuito. Por estas razones, *WebAnywhere* será el screen reader utilizado en el desarrollo de esta memoria.

WebAnywhere es un *screen reader* basado en la Web, que puede ser usado por personas con discapacidad visual para acceder la Web desde casi cualquier computador que tenga tanto conexión a Internet como salida de audio [2]. La portabilidad de *WebAnywhere* se debe a que se instala en un servidor, al cual se conectan los usuarios que deseen utilizarlo, desde cualquier computador que tenga conexión a Internet, por lo que no es necesario que se instale en cada equipo. Con los niveles de penetración de Internet, conseguir un computador con conexión a Internet no debería ser una limitante, sin embargo, para el caso de computadores sin conexión a Internet, *WebAnywhere* puede ser instalado en el mismo computador (aunque no fue pensado para ser usado de esta manera). Con esto, cualquier institución, ya sea pública o privada, o persona natural que desee apoyar la accesibilidad

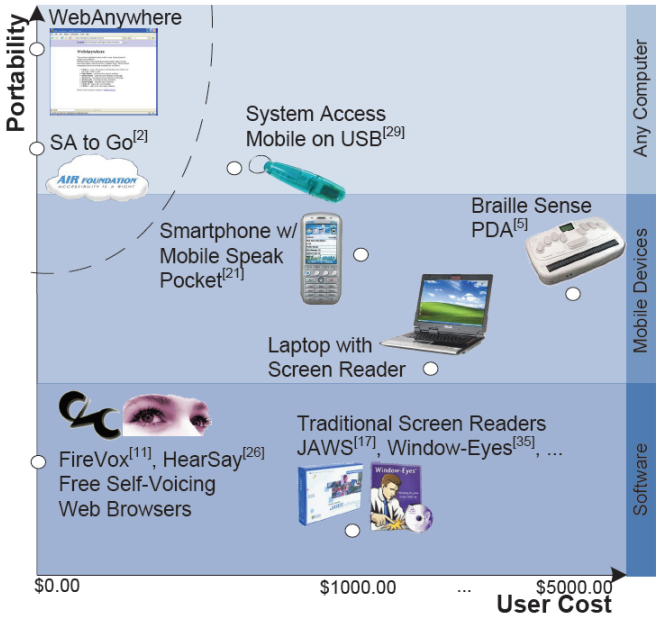

Figura 8: Clasificación de dispositivos que apoyan la accesibilidad, entre ellos los *screen readers*, según su portabilidad y costo [2].

de la información puede tener alojado *WebAnywhere* en sus servidores para que cualquier persona pueda acceder a él.

El funcionamiento de *WebAnywhere* es el siguiente: recibe como entrada la URL de la página que el usuario quiere visitar, obteniendo la estructura DOM de esa página. A través de un modelo *off-screen*. provee varias combinaciones de teclas que permiten navegar esa estructura DOM, hablando cada nodo de la estructura a medida que se va navegando. Si el usuario no utiliza ninguna combinación de teclas, por defecto *WebAnywhere* realiza un recorrido *Depth First Search (DFS)* de la estructura DOM. Hay ciertos nodos de la estructura (etiquetas HTML), como `<img/>` y `<a>` que son indicados de manera explícita por *WebAnywhere*, para que los usuarios noten la existencia de estos. Así, cuando *WebAnywhere* se encuentre con un link a Google, por ejemplo, lee "Link Google" y cuando se encuentra con una imagen de la forma `<img src="..." alt="fórmula matemática" />` lee "Imagen fórmula matemática".

Una de las limitaciones de *Webanywhere* es que sólo puede ser utilizado para leer contenido que está presente en la Web y no documentos que pueden estar en el computador que se esté utilizando, como archivos de texto, pdf, entre otros.

3.4. Generación de lenguaje natural

La generación de lenguaje natural puede ser clasificada, según su utilización, en dos categorías distintas. La primera corresponde a sistemas dependientes de la aplicación, los que tienen una fundamentación teórica-lingüística casi nula y, por otro lado, están los sistemas que tienen una buena fundamentación teórica-lingüística, siendo más generales [21].

Los *sistemas basados en templates* forman parte de la primera categoría, los cuales mapean una entrada no lingüistica, como datos, sin usar representaciones intermedias, a una estructura lingüística que sirve como salida. Un ejemplo de esto, obtenido de [21], es el siguiente: un sistema basado en templates podría generar lenguaje natural que esté relacionado con la salida de trenes desde estaciones tomando como entrada el número del tren, la localidad donde está situada la estación y la hora en la que deja la estación. De esta manera, con una entrada como $Salida(tren_{306}, localizacion_{Concepcion}, hora_{10:00})$ y un template como `El tren [tren] está dejando [localizacion] ahora` se pueden obtener textos como *El tren 306 está dejando Concepción ahora*. Este template es aplicable sólo para aquellos trenes que estén saliendo de alguna estación en el presente, debiendo generar otros templates para referirse a situaciones futuras o pasadas.

Por otro lado, en la segunda categoría se encuentran los llamados *sistemas estándar* de generación de lenguaje natural, los cuales tienen un mapeo menos directo de la entrada a la salida, en comparación con el primer grupo. Los sistemas con este enfoque toman la entrada, sometiéndola a sucesivas transformaciones hasta obtener el resultado final. Estas transformaciones consideran la mayoría de los aspectos de la lingüística, como tiempo, género, etc, para ser aplicados a la información de entrada. Utilizando un ejemplo similar al anterior, de la siguiente sucesión de transformaciones:

$$Dejar_{presente}(tren_{demostrativo}, Concepcion, ahora)$$

se puede generar el texto *este tren está dejando Concepción ahora*. La principal gracia de esta alternativa es la generalidad que puede dar, pues basta con considerar $Dejar_{futuro}$ para obtener oraciones en tiempo futuro, sin embargo es más complejo de tratar que el enfoque basado en templates, ya que considera más variantes lingüísticas.

A pesar de ser considerados como grupos distintos, los enfoques basados en templates y estándar son *Turing equivalentes* [21] entre sí. A pesar de esto, el enfoque basado en templates ha sido considerado como inferior al enfoque estándar, con respecto a lo bien fundado, calidad y variación de la salida y mantenimiento.

Con las fórmulas matemáticas pasa algo particular, debido a que las verbalizaciones de las expresiones matemáticas no consideran tiempo, género, adjetivos, entre otros. En

otras palabras, al momento de leer fórmulas matemáticas no hay distinción entre presente, pasado y futuro, no son ni masculino ni femenino y no possen calificativos. Esta simplicidad lingüística de las expresiones matemáticas hace que el enfoque estándar para generar lenguaje natural a partir de alguna representación de expresiones matemáticas sean más de lo que se necesita realmente, con lo que el enfoque basado en template surge como una mejor alternativa debido a su simplicidad.

En esta memoria de título, debido a las particularidades de la verbalización de las expresiones matemáticas, se utlizará un enfoque basado en templates para realizar la generación de lenguaje natural a partir de la codificación en *MathML* de las expresiones.

4. Reglas para leer fórmulas matemáticas

En la tabla 2 se muestran los operadores matemáticos más utilizados en Wikipedia que tienen una representación en *content MathML*. Estos operadores serán considerados como el conjunto mínimo de operadores que podrán ser verbalizados por el prototipo final de esta memoria. De esta manera, al verbalizar la mayoría de los operados más usados, el prototipo abarcará la mayoría de las expresiones matemáticas. Como se explicó en la sección 3.4, se usará un generador de lenguaje natural basado en templates, por lo que son necesarios templates que representen la manera más usual de leer expresiones matemáticas.

Con el objetivo de encontrar un conjunto de reglas que permitan leer expresiones matemáticas, se entrevistó a 3 profesoras de la Universidad de Concepción, con experiencia en matemáticas, que pudieran dar luces de cuales serían las reglas para leer las fórmulas. Las personas entrevistadas fueron Myriam Vicente[20], profesora asistente del Departamento de Matemática, Lilian Salinas[21], profesora asistente del Departamento de Ingeniería Informática y Ciencias de la Computación, y Anahí Gajardo[22], profesora asistente del departamento de ingeniería matemática. La conclusión de las tres entrevistas fue que no existe tal conjunto de reglas para leer fórmulas matemáticas, ya que es muy dependiente del lugar geográfico, colegio y edad de las personas el como leen expresiones matemáticas. Por ejemplo $\frac{a}{b}$ puede ser leída como "a dividido por b" o "a sobre b", mientras que para otras personas "a sobre b" es la verbalización del coeficiente binomial $\binom{a}{b}$. Además, personas de mayor edad conocían a las fracciones como "números quebrados", término que con el tiempo derivó en el término "fracciones".

Debido a la inexistencia de las reglas para leer expresiones matemáticas, se decidió realizar una encuesta de la forma en la cual las personas leen las expresiones matemáticas. La encuesta incluyó los operadores presentes en la tabla 2 y se le solicitó a 60 personas responderla, de los cuales 38 la respondieron efectivamente. La encuesta fue aplicada entre el 10 de Noviembre del 2010 y el 2 de Diciembre del mismo año. Entre las personas que respondieron la encuesta se encontraban estudiantes de distintas áreas de la ingeniería (informática, industrial, electrónica, civil), estudiantes de derecho, estudiantes de matemática, profesores universitarios y estudiantes de educación media. Las fórmulas que se usaron para realizar la encuesta fueron 21 y se muestran en la tabla 3; las fórmulas en la tabla incluyen todos los operadores de la tabla 2. Para aplicar la encuesta, se creó una página Web[23], en la cual, las fórmulas se codificaron en *presentation MathML*. Para cada expresión se disponía de un cuadro de texto para que los encuestados pudieran escribir la

[20] `http://dmat.cfm.cl/faculty/mvicente.html`

[21] `http://www.inf.udec.cl/index.php/departamento/academicos/lilian-salinas-ayala-phd`

[22] `http://www.ing-mat.udec.cl/~anahi/`

[23] La encuesta se encuentra disponible en `http://atenea.inf.udec.cl/~jfuentess/encuesta`

Fórmula	Fórmula
$\frac{x+1}{x-1}$	$3 \leq 3 \leq 4$
$x^3 + y^5$	$\sin(\cos x + x^3)$
$\sqrt[n]{a}$	$\log_3 x + \ln a$
$\frac{\partial^2 f(x,y)}{\partial x \partial y}$	$A \otimes B$
$x^2 \to a^2$	$\lim_{x \to 0} \sin x$
$\sum_{x=a}^{b} f(x)$	$\pi \approx 22/7$
$\sum_{x \in B} f(x)$	$\forall x : (x - x = 0)$
$a \in A$	$(f \circ g)(x) = f(g(x))$
$\int \sin = \cos$	$a \equiv \neg\neg a$
$\int_0^1 x^2 dx$	$4 \geq 3 \geq 3$
$A \times B$	

Tabla 3: Fórmulas usadas en la encuesta para obtener la manera más usual de leer expresiones matemáticas

o las maneras en las que leían las distintas expresiones. Un fragmento de la página Web utilizada se puede ver en la figura 9. En la figura se puede apreciar las instrucciones para responder la encuesta, entre las cuales las más importantes son:

- Para cada fórmula matemática, por favor, escriba la(s) manera(s) en la que usted la lee habitualmente.
- Si usted lee de distintas maneras un expresión, escríbalas en orden de uso (de las más usada a la menos usada).
- Por ejemplo $F = ma$ corresponde a "F igual a m por a".

Se recomendaba utilizar el navegador Web Firefox, debido a que Firefox soporta de manera nativa *presentation MathML*, a diferencia de otros navegadores, como Internet Explorer y Chrome, en los que se debe instalar complementos adicionales.

Como resultado de la encuesta se recibieron 772 respuesta en total, repartidas en cada fórmula como se muestra en la tabla 4

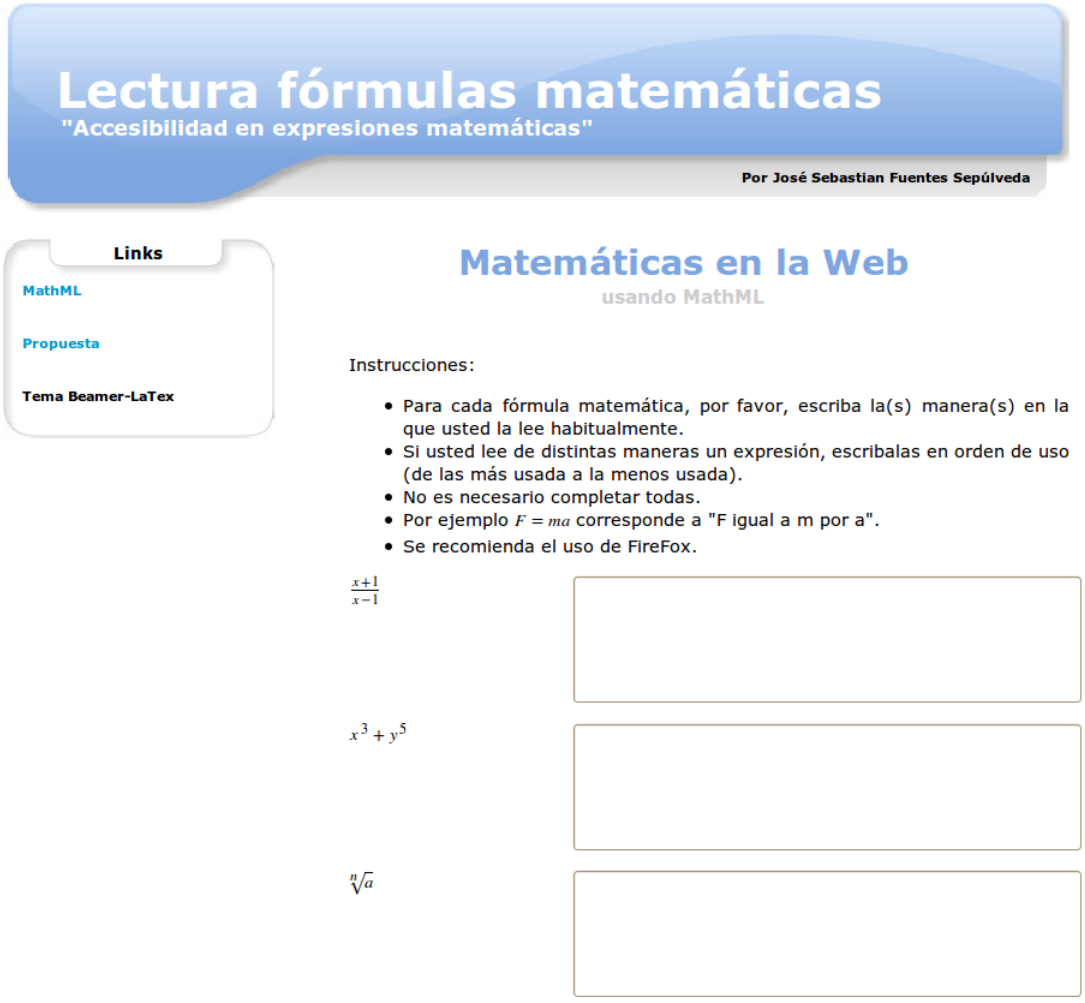

Figura 9: Fragmento de la página Web diseñada para aplicar la encuesta

Fórmula	Cantidad respuestas	Fórmula	Cantidad respuestas
$\frac{x+1}{x-1}$	42	$3 \leq 3 \leq 4$	38
$x^3 + y^5$	49	$\sin(\cos x + x^3)$	40
$\sqrt[n]{a}$	42	$\log_3 x + \ln a$	37
$\frac{\partial^2 f(x,y)}{\partial x \partial y}$	38	$A \otimes B$	12
$x^2 \to a^2$	32	$\lim_{x \to 0} \sin x$	36
$\sum_{x=a}^{b} f(x)$	41	$\pi \approx 22/7$	36
$\sum_{x \in B} f(x)$	33	$\forall x : (x - x = 0)$	36
$a \in A$	43	$(f \circ g)(x) = f(g(x))$	34
$\int \sin = \cos$	36	$a \equiv \neg\neg a$	29
$\int_0^1 x^2 dx$	37	$4 \geq 3 \geq 3$	41
$A \times B$	40		

Tabla 4: Cantidad de respuestas de la encuesta aplicada

Se puede apreciar en la tabla 4 que la fórmula con menor cantidad de respuestas fue $A \otimes B$, pudiendo ser por desconocimiento del operador $\otimes$ o por la ambigüedad del mismo, lo que se discutirá más adelante.

Cada una de las 772 respuestas fue agrupada según su similaridad sintáctica y semántica. De esta manera, por ejemplo, "x más 1, dividido en x menos 1" es sintácticamente diferente a "x más 1 dividido por x menos 1", pero ambas son semánticamente iguales. Una vez agrupadas las respuestas se procedió a contar cuántos grupos se formaron y cuántas respuestas había en cada grupo. En la mayoría de los casos, la respuesta más común (la que tiene el grupo sintáctico más grande) fue escogida como el template que representa a la fórmula y a todas las fórmulas similares. La lista de templates derivados de las respuestas de la encuesta se encuentran plasmados en la tabla 5. La segunda columna de la tabla corresponde a los templates derivados de cada fórmula de la primera columna. Los "comodines" utilizados en la definición de los templates son: \$OPx\$, con $x \in \mathbb{N}$, representa a los *operandos* sobre los cuales los operadores con aplicados. Por ejemplo, para la expresión $x + 1$ los operandos son x y 1, mientras que $+$ es el operador. \$GRADO\$ es el grado de una raíz. \$VAR\$ representa a las variables independientes involucradas en las integrales, derivadas, límites, etc. \$LIMx\$, con $x \in \mathbb{N}$, es el rango de aplicación de un operador dado. Para terminar, \$BASE\$ es la base de un logaritmo. La tercera columna corresponde al nivel de "confianza" del template escogido. El número de la izquierda es el porcentaje de las respuestas que tiene la misma o muy similar sintaxis, es decir, que pertenecen al mismo grupo sintáctico. Por su parte, el número de la derecha corresponde a la "confianza semántica" del número de la izquierda. La "confianza semántica" corresponde a la cantidad total de grupos semánticos presentes en cada fórmula, por lo que tiene un sentido de dispersión. De esta manera, un template que tenga una "confianza semántica" elevada se considera poco confiable, pues las respuestas de las que se derivó están más dispersas (es una fórmula que genera ambigüedad), en cambio, si la "confianza semántica" es 1, se está en presencia de un concenso perfecto.

El template de la expresión $x^3 + y^5$ no corresponde al template del grupo con mayor cantidad de respuestas, sino que corresponde al que obtuvo la segunda mayoría. El template que contaba con la primera mayoría es "\$OP\$ al \$GRADO\$" (por ejemplo, "x al cubo"), sin embargo, este template no es lo suficientemente general, ya que para grados mayores, por ejemplo, un polinomio grado 30 no se usa este template para leerse las expresiones. Por esta razón se optó como template definitivo a aquel que contaba con la segunda mayoría, pues es más general. Por su parte, el template que captura la expresión $x^2 \to a^2$ no es el que obtuvo la mayoría. El template que contaba con la mayoría para esta expresión es "\$OP1\$ implica \$OP2\$". Esta confusión por parte de los encuestados se debió a que el símbolo $\to$ es ambigüo, ya que se usa tanto en implicaciones lógicas como en el operador "tiende a". Donde no hubo un concenso fue en la expresión $A \otimes B$.

Fórmula	Template	Confianza
$\frac{x+1}{x-1}$	$OP1$ dividido por $OP2$	0.40 (1)
$x^3 + y^5$	$OP1$ elevado a $OP2$	0.33 (1)
$\sqrt[n]{a}$	raiz $GRADO$ de OP	0.55 (2)
$\frac{\partial^2 f(x,y)}{\partial x \partial y}$	[$GRADO$] derivada de OP, con respecto a VAR(, VAR)*	0.13 (6)
$x^2 \to a^2$	$OP1$ tiende a $OP2$	0.25 (4)
$\sum_{x=a}^{b} f(x)$	Sumatoria desde $LIM1$ hasta $LIM2$ de OP	0.20 (1)
$\sum_{x \in B} f(x)$	Sumatoria de OP, con DOM	0.15 (2)
$a \in A$	$OP1$ pertenece a $OP2$	0.65 (1)
$\int \sin = \cos$	Integral de OP	0.5 (1)
$\int_0^1 x^2 dx$	Integral desde $LIM1$ hasta $LIM2$ de OP respecto a VAR(, VAR)*	0.19 (1)
$A \times B$	$OP1$ cruz $OP2$	0.55 (1)
$3 \leq 3 \leq 4$	$OP1$ menor o igual que $OP2$ (menor o igual que OP)*	0.55 (1)
$\sin(\cos x + x^3)$	Seno de OP | Coseno de OP	0.60 (1)
$\log_3 x + \ln a$	Logaritmo en base $BASE$ de OP | Logaritmo natural de OP	0.65 (1)
$A \otimes B$	Producto tensorial entre $OP1$ y $OP2$ $OP1$ ex or $OP2$ $OP1$ cruz $OP2$	0.17 (8)
$\lim_{x \to 0} \sin x$	Límite de OP, cuando VAR	0.58 (1)
$\pi \approx 22/7$	$OP1$ es aproximadamente $OP2$	0.53 (2)
$\forall x : (x - x = 0)$	Para todo VAR, OP	0.53 (1)
$(f \circ g)(x) = f(g(x))$	$OP1$ compuesta $OP2$	0.24 (2)
$a \equiv \neg\neg a$	$OP1$ es equivalente a $OP2$	0.62 (3)
$4 \geq 3 \geq 3$	$OP1$ mayor o igual a $OP2$ (mayor o igual a OP)*	0.32 (2)

Tabla 5: Templates derivados de la encuesta

Aquí se obtuvieron tres templates, los tres con una frecuencia muy baja y una alta dispersión (confianza semántica de 8). Esto significa que el operador $\otimes$ es muy ambigüo y poco conocido. Finalmente, el template de la expresión $(f \circ g)(x) = f(g(x)))$ es la segunda mayoría, siendo la primera "$OP1$ o $OP2$". La razón por la que se eligió finalmente la segunda mayoría en desmedro de la primera fue porque esta última se puede confundir con la verbalización del operador lógico *or*.

Como conclusión, en la tabla 5 se aprecia que, en general, existe una alta variabilidad sintáctica a la hora de verbalizar las expresiones matemáticas, es decir, hay varias maneras de leer las expresiones, sin embargo, existe una baja variabilidad semántica, en general, lo que quiere decir que a pesar de que se lean de maneras distintas las expresiones, la intención o concepto es el mismo. Esto es un punto a favor en la metodología de "leer" las expresiones matemáticas usando screen readers, ya que lo importante es que los usuarios entiendan las expresiones matemáticas (semántica), a pesar de la manera en la que se diga (sintaxis). Esto valida la alternativa de escoger como template a aquel que se derive del grupo sintáctico que tiene la mayor cantidad respuestas, siempre y cuando ese template tenga una baja variabilidad semántica.

Como ya se tienen los templates de las expresiones matemáticas más usadas, el siguiente paso es implementar un generador de lenguaje natural que utilice estos templates para verbalizar las expresiones matemáticas.

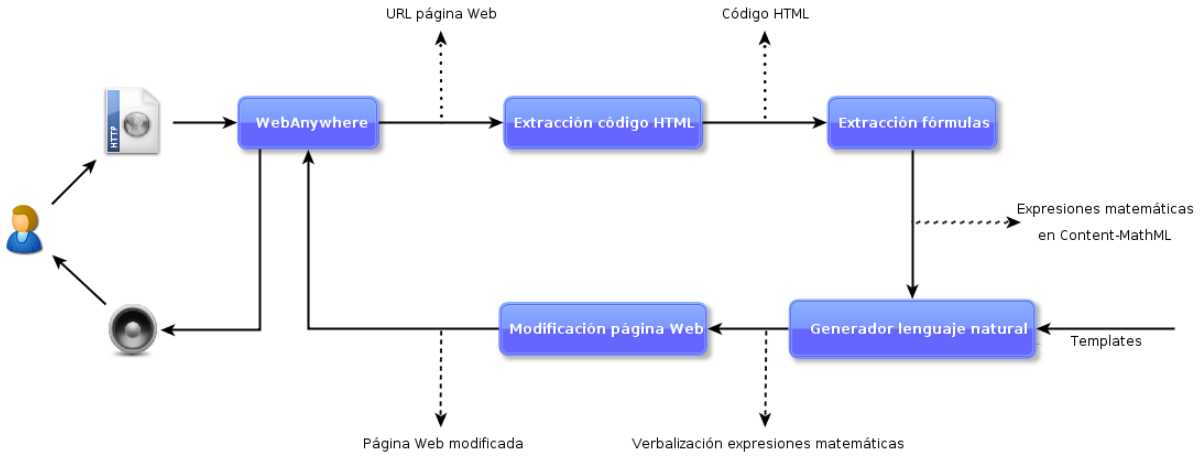

Figura 10: Arquitectura del prototipo

5. Implementación

La arquitectura del prototipo implementado para la desarrollo de esta memoria de título se muestra en la figura 10.

En la arquitectura, *WebAnywhere* cumple el papel de interfaz, recibiendo la URL de la página que el usuario desea visitar y retornando el audio del contenido de la página dada como entrada. Hasta este punto nada distinto a lo que ya realiza *WebAnywhere* por si solo. Lo que se agregó en este memoria fue un *preprocesado* del contenido de la página dada como entrada, antes de ser generado el audio respectivo. El preprocesado comienza con el módulo llamado *Extracción código HTML*, el cual toma como entrada la URL que el usuario suministró a través de *WebAnywhere* y retorna el código HTML correspondiente a esa URL. Con el código HTML capturado, el siguiente paso es extraer todas las expresiones matemáticas contenidas en la página Web, lo cual se lleva a cabo en el módulo *Extracción fórmulas*. Este módulo busca fórmulas matemáticas contenidas en imágenes, con su respectiva codificación en *Content-MathML*, en el atributo `alt` del tag `<img/>` y el atributo `class` instanciado con *math*, similar a como lo hace Wikipedia. Las razones para tomar esta opción para expresar las fórmulas matemáticas se discutirá un poco más adelante. Por ejemplo, para la expresión $x^2 + 1$ el código HTML esperado sería:

```
<img
  class="math"
  src="formula.png"
  alt="<math><apply><plus/><apply><power/><ci>x</ci>
  <cn>2</cn></apply><cn>1</cn></apply></math>"
```

```
/>
```

Con las fórmulas matemáticas codificadas en *Content-MathML* el paso siguiente en la arquitectura es el módulo de *Generación de lenguaje natural*. Aquí se implementó un generador de lenguaje natural basado en templates o plantillas, utilizando las plantillas descritas en la tabla 5. Por cada expresión matemática obtenida de la página Web, se aplica el módulo de generación de lenguaje natural, obteniendo de esta manera la verbalización de cada expresión. La manera en la que este módulo entrega la verbalización de la expresiones es utilizando el tag `<p>`, para reflejar el anidamiento de las expresiones. Por ejemplo, para la expresión anterior, $x^2 + 1$, la salida del módulo de generación de lenguaje natural sería:

```
<p>
  <p> x elevado a 2</p>
  más 1
</p>
```

La inclusión de los tags `<p>` servirá posteriormente para recorrer las expresiones matemáticas utilizando el teclado, opción que trae incorporada *WebAnywhere*. De esta manera, se podrán navegar las distintas partes de las expresiones, recorriendo las estructura DOM de la página Web, lo que ayudará a tener una mejor comprensión de la estructura y semántica de la fórmula. Con la verbalización lista, el módulo *Modificación página Web* se encarga de incluir en la página Web la verbalización de las fórmulas, en el lugar donde está insertada la imagen con la fórmula, acompañada con un CSS que mantiene oculta la verbalización para no cambiar el aspecto original de la página. Esto último da término al preprocesado de la página Web, la cual ahora tiene explícitamente la verbalización de todas las expresiones matemáticas contenidas en ella. A partir de aquí, *WebAnywhere* puede seguir funcionando normalmente, navegando la estructura DOM de la página Web modificada, permitiéndole a los usuarios conocer las expresiones matemáticas que están contenidas en la página.

La decisión de pedir que las expresiones matemáticas estuvieran como imágenes, junto con su codificación en *Content-MathML*, se basa es que las imágenes son soportadas por todos los navegadores Web, sin importar el sistema operativo con que trabajen los usuarios. La codificación en *Content-MathML* le da la semántica de la expresión a la imagen que la representa. Otra alternativa que se evaluó fue utilizar integramente *MathML*, representando las fórmulas tanto con *Presentation-MathML* como con *Content-MathML*, sin embargo, *Presentation-MathML* aun no es renderizado por todos los navegadores Web por defecto. La elección de esta última opción para el uso en esta memoria habría condicionado

su buen funcionamiento a sólo algunos navegadores, como *FireFox* y *Safari*.

La principal ventaja de realizar el preprocesamiento de la página Web, antes de que se genere el audio, es que entrega una independencia del screen reader que se quiera usar. El resultado del preprocesamiento es la página Web inicial más la verbalización explícita de las expresiones matemáticas. El hecho de que la verbalización esté explícita en el código HTML de la página permite que se pueda utilizar cualquier screen reader para leer la página Web modificada por el preprocesamiento. En particular, para utilizar *WebAnywhere* sólo se debió capturar la URL que el usurio sumistraba a través de *WebAnywhere* y posteriormente retornarle a *WebAnywhere* la página modificada, por lo que la intervención al código de *WebAnywhere* es casi nula.

Concretamente, una vez que *WebAnywhere* recibe la URL, ésta es pasada como parámetro a una página escrita en PHP, la cual es la encargada de aplicar los distintos módulos presentados en la arquitectura, todos ellos escritos en Python. De esta manera, si la página que se desea leer es `http://www.formulas.cl`, la página modificada, que contiene la verbalización de las expresiones, sería algo como `http://www.url.cl/preprocesamiento.php?url=http://www.formulas.cl`, claro que esto último es trasparente para el usuario.

5.1. Seguimiento

Para explicar con más detalle el funcionamiento del prototipo, en especial del módulo de *generación de lenguaje natural*, se hará el seguimiento de un ejemplo. Asumamos que contamos con una página Web que contiene la fórmula $E = mc^2$, es decir, dentro del código HTML de la página se encuentra

```
<img
  class="math"
  src="formula.png"
  alt="<math><apply><eq/><ci>E</ci><apply><times/><ci>m</ci>
  <apply><power/><ci>c</ci><cn>2</cn></apply></apply></apply></math>"
 />
```

Todo es fácil hasta el módulo de *Extracción de fórmulas*, el cual extrae el contenido del atributo `alt`. Con la codificación de la fórmula en *Content-MathML* se da inicio al módulo de *generación de lenguaje natural* basado en templates.

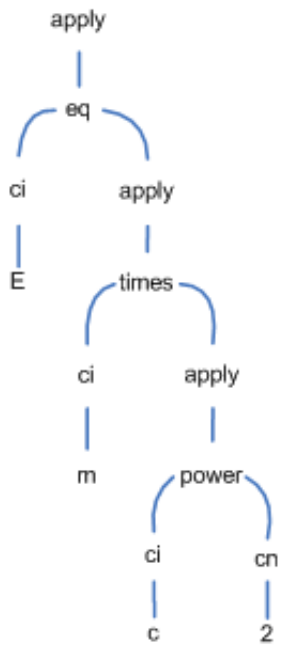

Figura 11: Árbol de la ecuación $E = mc^2$ en *Content-Mathml*

Lo primero que se hace en el módulo de *generación de lenguaje natural* es aplicar un proceso de parsing a la entrada, utilizando xml.etree.ElementTree[24] de Python. La expresión luego de ser *parseada* puede ser recorrida fácilmente a través del árbol que la representa, como se muestra en la figura 11.

A medida que se recorre el árbol usando el método *DF*(depth first), los elementos del mismo se van apilando, a partir de la raíz, en una pila (de ahora en adelante llamada *stack*). Las reglas para ir apilando los nodos del árbol, llamadas *reglas de apilación*, son las siguientes:

- Cada vez que se encuentra un nodo "apply" se apila en *stack* el string "start", el cual indica que se dió inicio a la aplicación de un operador.
- Cada vez que se encuentra un nodo "ci" o "cn" se apila el contenido de su nodo hijo.
- En cualquier otro caso se apila el nodo sin modificaciones.

Cada vez que se termine de recorrer y apilar todos los nodos bajo un nodo "apply", según las *reglas apilación* o bien ya no queden más nodos que recorrer, se aplica el proceso de verbalización. Esto se realiza desapilando los elementos que están en *stack* según las siguientes reglas, llamadas *reglas de desapilación*:

[24]http://docs.python.org/library/xml.etree.elementtree.html

- Desapilar todos los elementos de la pila hasta encontrar el primer elemento “start”.
- Luego, a los elementos desapilados se le aplica el proceso de generación de lenguaje natural.
- El texto resultante del punto anterior es apilado en *stack*.
- Volver a aplicar las reglas para apilar elementos en *stack*, si es que aún quedan nodos por recorrer. En caso contrario, volver a aplicar las *reglas de desapilación* hasta que no queden más elementos que verbalizar.

Para el árbol del ejemplo el recorrido es el siguiente:

- El nodo raíz es “apply”, por lo tanto se apila en *stack* el string “start”.
- Luego se avanza hasta el único hijo que tiene, el cual es “eq”, por lo que es apilado. Hasta este punto *stack* tiene la siguiente forma [‘start’, ‘eq’], considerando el extremo derecho como el tope de la pila.
- Avanzando a los hijos del nodo “eq”, escogiendo el de la izquierda nos encontramos con el nodo “ci”. Según las reglas se debe apilar el contenido de su nodo hijo, en este caso se apila “E ”.
- A continuación se avanza hasta el nodo derecho de “eq”, el que corresponde a un nodo “apply”, por lo que se apila el string “start”.
- Se apila “times”, único hijo del nodo “apply” anterior. Hasta aquí *stack* tiene la siguiente forma: [‘start’, ‘eq’, ‘E’, ‘start’, ‘times’].
- El hijo de más a la izquierda del nodo “times” es el nodo “ci”, por lo que se debe apilar el nodo hijo de “ci”, siendo este el nodo “m”.
- Como no se puede seguir avanzando por el lado izquierdo, ahora se toma el hijo derecho del último nodo “apply”. El hijo derecho es el nodo “apply”, por lo que se apila “start” en *stack*.
- A continuación se apila “power”, único hijo del nodo “apply” anterior.
- El nodo “power” tiene como hijo izquierdo a “ci” y como derecho a “cn”, por lo que se apilan los hijos de ambos, quedando finalmente *stack* como [‘start’, ‘eq’, ‘E’, ‘start’, ‘times’, ‘m’, ‘start’, ‘power’, ‘c’, ‘2’].

Hasta aquí, se recorrieron todos los nodos del árbol que representa la expresión matemática, por lo que el siguiente paso es aplicar las *reglas de desapilación*. El proceso es el siguiente:

- La primera regla a utilizar es desapilar todos los elementos de *stack* hasta llegar al primer elemento "start". Los elementos que se van desapilando desde *stack* se irán apilando en una pila auxiliar. Aplicando la primera regla de desapilación *stack* quedaría ['start', 'eq', 'E', 'start', 'times', 'm'] y la pila auxiliar quedaría ['2', 'c', 'power']. Cabe señalar que el elemento "start" es desapilado de *stack*, más no es apilado en la pila auxiliar.

- Luego, una vez lista la pila auxiliar, es el turno del *proceso de generación de lenguaje natural*. Este proceso toma el tope de la pila auxiliar, el que asume es el operador de la expresión matemática y busca el template o plantilla adecuada a ese operador. Las plantillas están almacenadas en un diccionario, el cual usa como índice el nombre de los operadores. En este caso se busca el template que tiene como índice a "power", el cual es "$OP1$ elevado a $OP2$" (un operador binario, ya que contiene dos "comodines", $OP1$ y $OP2$).

- Luego de obtener el template adecuado, se procede a reemplazar los "comodines" con los elementos restantes de la pila auxiliar. El elemento del tope de la pila reemplaza al "comodín" de más a la izquierda, repitiendo este paso hasta que no queden elementos en la pila auxiliar. Según esto, el "comodín" $OP1$ es reemplazado por "c" y $OP2$ por "2", quedando como resultado "c elevado a 2".

- A continuación, al terminar cualquier iteración del *proceso de generación de lenguaje natural*, se añaden en los extremos del texto generado el tag `<p>`. Para este caso queda " c elevado a 2 "

- El texto generado es apilado en *stack*, con lo que *stack* queda como sigue: ['start', 'eq', 'E', 'start', 'times', 'm', ' c elevado a 2 '].

- Como ya no quedan más elementos que apilar, desde el árbol, en *stack*, debido a que todos los que se debían apilar ya fueron apilados, se vuelven a aplicar las *reglas de desapilación*.

- Desapilando los elementos de *stack* hasta el siguiente elemento "start", la nueva pila auxiliar queda [' c elevado a 2 ', 'm', 'times'] y *stack* queda ['start', 'eq', 'E'].

- Según el *proceso de generación de lenguaje natural*, el elemento del tope de la pila auxiliar es el operador, por lo que debe buscar en el diccionario de templates al template con el índice "times", el cual corresponde a "OP *". Este template tiene algo particular, el asterisco en su interior, lo que quiere decir que en un operador n-ario. Todo template n-ario tiene asociado un segundo template, el cual tiene como objetivo sustituir el asterisco tantas veces como sean necesarias, permitiendo que el template pueda soportar una cantidad variable de operandos. El segundo template es

obtenido de un diccionario adicional, que también usa como índices a los operandos. En este caso el template adicional para el operador n-ario "times" es "por OP *". Siguiendo con el ejemplo, se desapila el elemento del tope de la pila auxiliar, "m", el cual reemplaza al "comodín" del template principal, quedando "m *". Como quedan elementos en la pila auxiliar, el asterisco es reemplazado por el segundo template, obteniendo "m por OP *". Luego, se desapila un nuevo elemento de la pila auxiliar y se reemplaza el "comodín" OP, resultando en "m por c elevado a 2 *". Como ya no quedan elementos en la pila auxiliar, el asterisco que aun queda es eliminado, obteniendo "m por c elevado a 2 ".

- Se agrega el tag `<p>`, " m por c elevado a 2 ", el cual es apilado en *stack*.
- Finalmente se llegó al último operador. Se desapilan los elementos de *stack*, el cual queda vacío y la pila auxiliar queda [' m por c elevado a 2 ', 'E', 'eq'].
- Se busca el template asociado a "eq", el cual corresponde a "$OP1$ es igual a $OP2$".
- Se reemplazan los "comodines", $OP1$ por "E" y $OP2$ por " m por c elevado a 2 ", obteniendo "E es igual a m por c elevado a 2 ".
- Se añade en los extremos el tag HTML `<p>`, " E es igual a m por c elevado a 2 ".
- Se apila en *stack*.
- Como ya no quedan más elementos a los que se les debe aplicar las *reglas de desapilación*, se toma al único elemento de *stack* como la verbalización final de la expresión matemática inicial, $E = mc^2$.

Con la salida del módulo de *generación de lenguaje natural* lista, se da paso al módulo de *modificación de la página Web*. En este módulo se toma la verbalización obtenida en el módulo anterior y se inserta en el código de la página Web original, a continuación de la imagen correspondiente a la expresión matemática. Además, la verbalización va acompañada por el estilo CSS:

```
.hidden {
  display:none;
}
```

Este código CSS hace que la verbalización no se muestre a través del navegador Web, a pesar de que es parte de la estructura de la página Web y con ello visible para *WebAnywhere* o cualquier otro screen reader. Esto permite que la página Web que se quiere leer no deforme su estructura visible original, entregando la verbalización de una manera transparente para el usuario final.

En el ejemplo, el resultado final queda de la siguiente manera:

```
<img
  class="math"
  src="formula.png"
/>
<div class="hidden">
  <p>
      E es igual a
    <p>
      m por
      <p>
        c elevado a 2
      </p>
    </p>
  </p>
</div>
```

Cabe señalar que el atributo `alt` del tag `<img>` se elimina, debido a que los screen readers, por lo general, leen lo que está contenido en el atributo `alt` de los distintos tags de HTML, lo que ahora ya no es necesario, pues se tiene la versión en lenguaje natural la expresión matemática lista para que cualquier screen reader, en particular WebAnywhere, pueda leer en voz alta.

Además de los templates que se muestran en la tabla 5, el prototipo utiliza otros templates adicionales, asociados a las operaciones básicas de suma, resta, multiplicación y división, y otras operaciones que no están consideradas dentro de las más utilizadas, pero que aun así fueron incluídas para darle un mayor alcance al prototipo, como es el caso del valor absoluto, factorial, operadores lógicos, operadores de conjuntos y funciones trigonométricas. Adicionalmente, el prototipo considera el uso de letras griegas en la definición de las expresiones. De esta manera, el prototipo verbaliza un conjunto mayor de expresiones matemáticas que el abarcado por las expresiones matemáticas más usadas en Wikipedia.

6. Evaluación

El objetivo de esta evaluación es medir, qué tan útil resulta el prototipo al momento de asistir a personas que no pueden ver expresiones matemáticas.

6.1. Participantes

El ideal para realizar la evaluación es contar con el apoyo de personas que tienen discapacidad visual. Con este propósito, se realizó el contacto con *Coalivi*[25], para intentar realizar las evaluaciones con las personas que ayuda. *Coalivi* cuenta con asistencia médica, óptica y un pequeño colegio dedicados a personas con discapacidad visual. El rango de edad de los integrantes del colegio llega hasta los 18 años de edad, con asignaturas que están incluídas en los programas del ministerio de educación, sin abarcar necesariamente los programas completos. Esto último representa una limitante, ya que la evaluación incluirá integrales, límites, etc, contenidos que no forman parte de los programas de matemáticas del ministerio de educación. De este modo, realizar la evaluación con niños de edad escolar obligará a reducir el espectro de tipos de expresiones que se pueden evaluar, lo que limitaría bastante los resultados de la evaluación. Debido a esto, es que decidió no realizar la evaluación con los alumnos de *Coalivi*, aplazándolo para futuras evaluaciones de esta memoria.

Para realizar un buena evaluación del prototipo se necesitan participantes que tengan experiencia usando los tipos de expresiones matemáticas usadas en la evaluación, de manera tal que si existe un error en la evalución se deba a que el prototipo falló al apoyar la accesibilidad y no a que el participante desconocía las expresiones. Con esto en mente, se le pidió a 20 personas, estudiantes de ingeniería con al menos 3 años de antiguedad en sus respectivas carreras y profesionales de la ingeniería, que evaluarán el prototipo. Entre los participantes que ayudaron en la evaluación no se encontraban impedidos visuales.

6.2. Estímulo

El estímulo utilizado en la evaluación consistió de dos páginas Web, cada una con 15 expresiones matemáticas distintas, obtenidas desde *Wikipedia* de manera aleatoria. Las 15 expresiones de cada página estaban conformadas por operadores contenidos en la tabla 1, además de otros que no estaban en esa tabla para darle una mayor variabilidad, como es el caso del operador factorial, tangente y subconjunto. La primera página Web fue usada para evaluar indicadores prosódicos (en este caso sólo pausas) y la segunda página permitió evaluar inidicadores léxicos. Cada una de las fórmulas en ambas páginas fueron ocultadas de la vista de los participantes usando CSS, de manera tal que no pudieran

[25] *Corporación de ayuda al limitado visual*, http://www.coalivi.cl

Fórmula	Fórmula	Fórmula
$(1-\alpha)^k$	$\gamma = \frac{1}{\sqrt{1-\beta^2}}$	$\forall x, f(x) \leq g(x)$
$\sqrt{8a^3}$	$\tan(x) = \frac{\sin(x)}{\cos(x)}$	$\lim_{n\to\infty} \frac{1}{L_n} = \lambda$
$M(x) = exp \int a(x)dx$	$\frac{s}{s^2+w^2}$	$(\ln(x!) \approx x\ln(x) - x)$
$g \in A_e$	$\frac{\partial \phi}{\partial t} + c^2 \frac{\partial u}{\partial x} = 0$	$g \circ f = e = f \circ g$
$c_2 \to t_2$	$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$	$x_u + x_v \geq 1$

Tabla 6: Fórmulas utilizadas en la evaluación de indicadores prosódicos.

Fórmula	Fórmula	Fórmula
$z_1(x,y) = \frac{x^2+y^2}{2}$	$\int \sum_{r=a}^{b} f(r,x)dx$	$D = M\frac{\partial f}{\partial c}$
$\frac{\pi}{180}$	$z = a_x * x + a_y * y + d$	$x - x \times x \equiv (1-x) \times x$
$E(Y) = \sigma\sqrt{\frac{2}{\pi}}$	$\ln(x) = \ln(m) + n\ln(2)$	$x < N \to x + 1 \leq N$
$f(U) \subset V$	$\sin(\frac{\pi}{2} - \theta) = \cos\theta$	$f(tx + (1-t)y) \geq tf(x) + (1+t)f(y)$
$y_n(x) = -(-x)^n(\frac{1}{x})^n\frac{\cos x}{x}$	$(n+1)^n > n^n n$	$\lim_{a\to 1} \frac{a-1}{a+1} = 0$

Tabla 7: Fórmulas utilizadas en la evaluación de indicadores léxicos.

acceder a la fórmula de manera visual, intentando recrear lo que pasa con personas que sí tienen problemas visuales. Ambas páginas Web fueron leídas a los participantes usando *WebAnywhere*, leyendo las expresiones de una manera más cercana a como se leen normalmente en el caso de la página con indicadores prosódicos y leyendo de manera explícita el inicio y final de operadores como fracción, seno, sumatoria, etc, para el caso de la página con indicadores léxicos.

Las fórmulas utilizadas en la evaluación de los indicadores prosódicos están reflejadas en la tabla 6 y las correspondientes a los indicadores léxicos están en la tabla 7.

El sintetizador de voz utilizado para generar las verbalizaciones en español fue el mismo que utiliza *Google translate*, el tiene una versión de español de España.

6.3. Procedimiento

La evaluación se realizó en un ambiente controlado, el cual considera el uso de un computador común con acceso a Internet y parlantes o audífonos, para escuchar la lectura de las expresiones.

Los participantes recibieron instrucciones al inicio de ambas páginas. Estas instrucciones indicaban que a continuación se leerán, en lenguaje natural, 15 expresiones matemáticas, las que deberán escribir en notación matemática, según creen que han escuchado. Los participantes escribieron las distintas expresiones en una hoja que se les facilitó, para que pudieran escribir a mano alzada las expresiones. También, como parte de las instrucciones, se les informó que por defecto el prototipo leería de comienzo a fin todas las expresiones, pero que si en algún momento creían conveniente detener el avance de la lectura podían hacerlo, usando la *tecla arriba* para retroceder en la lectura y la *tecla abajo* para avanzar en la lectura. Una vez comenzada la lectura no se le prestó más apoyo a los participantes, para no influir involuntariamente en sus respuesta. También cada participante tenía la opción de realizar sus comentarios al final de la aplicación de la evaluación, ya sea de manera escrita en la misma hoja de resultados y de manera oral.

6.4. Resultados

Con las respuestas de los participantes listas, el paso de obtención de los resultados se realizó de la siguiente manera: Para cada expresión se comparó la expresión correcta contra la expresión que escribió cada participante para aquella expresión. Si eran iguales se contaba como un acierto, en caso contrario como un error. De esta manera, la situación ideal de la evaluación se daría si es que todos los participantes acertaran a la expresión correcta.

La cantidad de respuestas correctas al revisar las expresiones de la página con indicadores prosódicos está en la tabla 8. Con estos números se obtiene que el prototipo tiene cerca de un 76 % de efectividad. Esto considera sólo el uso de pausas al momento de leer las expresiones en lenguaje natural, por ejemplo, para la expresión $\frac{x+1}{3}$, se obtendría una lectura similar a "x más uno (pausa) dividido en tres".

Por su parte, las respuestas correctas para las pruebas con indicadores léxicos están en la tabla 9. Se puede obtener de esta tabla, que aproximandamente, el prototipo tiene un 79,3 % de de efectividad usando estos indicadores. Por ejemplo, para la misma expresión $\frac{x+1}{3}$, su verbalización usando indicadores léxicos sería algo como "`inico fracción` x más uno (pausa) dividido en 3 `fin fracción`". Para estas pruebas, también se usaron pausas para que pareciera más similar a la manera en la que se leen las expresiones.

La intención de fondo para hacer dos evaluaciones es comprobar qué tan efectivos son el

Fórmula	Correctas	Fórmula	Correctas
$(1-\alpha)^k$	14	$\frac{\partial \phi}{\partial t} + c^2 \frac{\partial u}{\partial x} = 0$	18
$\sqrt{8a^3}$	20	$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$	19
$M(x) = exp \int a(x) dx$	13	$\forall x, f(x) \leq g(x)$	20
$g \in A_e$	8	$\lim_{n \to \infty} \frac{1}{L_n} = \lambda$	14
$c_2 \to t_2$	17	$(\ln(x!) \approx x \ln(x) - x)$	14
$\gamma = \frac{1}{\sqrt{1-\beta^2}}$	19	$g \circ f = e = f \circ g$	17
$\tan(x) = \frac{\sin(x)}{\cos(x)}$	17	$x_u + x_v \geq 1$	10
$\frac{s}{s^2 + w^2}$	8		

Tabla 8: Cantidad respuestas correctas para indicadores prosódicos.

uso de indicadores prosódicos simples (pausa) y los indicadores léxicos, unidos con pausas.

6.5. Discusión

Observando los resultados, se puede apreciar que el uso de indicadores léxicos entrega mejores resultados que el uso de indicadores prosódicos simples. Sin embargo, esta superioridad es baja, por lo que se necesita hacer un análisis más minucioso de los resultados para lograr concluir algo más acertado.

En la tabla de respuestas correctas usando sólo indicadores prosódicos, se aprecia que existen 3 expresiones matemáticas que obtvieron puntajes muy bajos, inferiores a 13. En el caso de las expresiones $g \in A_e$ y $x_u + x_v \geq 1$, la razón del por qué obtuvieron bajos puntajes es similar. El problema ocurre en dos lugares, el primero de ellos es el sintetizador de voz, debido a que los subíndices los lee unidos o pegados a la base del subíndice, por ejemplo, x_u es leído como "xsubu". Esto causa que no se pueda distinguir claramente de qué expresión se trata. Además, actualmente, *WebAnywhere* es incapaz de controlar el tiempo, tono, timbre, velocidad, etc, de la verbalización. Si esto último fuera posible, al momento de leer una expresión que contiene subíndice, se podría reducir la velocidad para dejar más claro cuál es la expresión. Para el caso de la expresión $\frac{s}{s^2+w^2}$, los 12 participantes que respondieron mal contestaron $\frac{s}{s^2} + w^2$. Esto quiere decir que el problema pasa porque el anidamiento de esta expresión no pudo ser capturado usando sólo pausas.

Fórmula	Correctas	Fórmula	Correctas
$z_1(x,y) = \frac{x^2+y^2}{2}$	13	$\sin(\frac{\pi}{2} - \theta) = \cos\theta$	20
$\frac{\pi}{180}$	19	$(n+1)^n > n^n n$	13
$E(Y) = \sigma\sqrt{\frac{2}{\pi}}$	18	$D = M\frac{\partial f}{\partial c}$	18
$f(U) \subset V$	15	$x - x \times x \equiv (1-x) \times x$	20
$y_n(x) = -(-x)^n(\frac{1}{x})^n\frac{\cos x}{x}$	3	$x < N \rightarrow x + 1 \leq N$	20
$\int \sum_{r=a}^{b} f(r,x)dx$	15	$f(tx + (1-t)y) \geq tf(x) + (1+t)f(y)$	7
$z = a_x * x + a_y * y + d$	18	$\lim_{a\to 1} \frac{a-1}{a+1} = 0$	19
$\ln(x) = \ln(m) + n\ln(2)$	20		

Tabla 9: Cantidad respuestas correctas para indicadores léxicos.

Con respecto a los resultados de tabla 9, a pesar de que en promedio tiene una mejor evaluación, también tiene las expresiones con más baja cantidad de respuestas correctas, $y_n(x) = -(-x)^n(\frac{1}{x})^n\frac{\cos x}{x}$ con 3 respuestas correctas y $f(tx+(1-t)y) \geq tf(x)+(1+t)f(y)$ con 7 respuestas correctas. Los indicadores léxicos que se usaron trabajaban sobre fracciones, sumatorias, integrales, derivadas, raíces, senos, cosenos, logaritmos y límites, indicando el inicio y fin de cada uno, cada vez que son aplicados. Observando las expresiones que obtuvieron menos respuestas correctas se aprecia que tienen varios paréntesis, los que no fueron representados en la verbalizados con indicadores léxicos, sino que sólo usando pausas. Esto último puede ser la causa del por qué obtuvieron tan malas respuestas. Sería interesante repetir la evaluación utilizando indicadores léxicos para los paréntesis y observar si las respuestas positivas aumentan. La razón de por qué esto no fue implementado ahora, sino que fue dejado para el futuro, es debido a que *Content-MathML* no representa explícitamente los paréntesis en su notación, sino que usa el anidamiento de sus etiquetas para representar los paréntesis. De hecho, cada vez que hay anidamiento en *Content-MathML*, se podría entender que existen paréntesis que engloban los distintos argumentos, lo que no quiere decir necesariamente que serán representados de manera visual, ya que también pueden ser omitidos. Para poder hacer explícitos los paréntesis no basta con usar solamente *Content-MathML*, sino que habría que usar además *Presentation-MathML*, lo que está fuera del alcance de esta memoria.

Los participantes tenían la alternativa de dar sus opiniones y/o comentarios sobre el prototipo, destacando 2 observaciones, ya que fueron mencionadas por más de un par-

ticipante. La primera fue que la evaluación con indicadores prosódicos fue más fácil de completar, lo cual se vió reflejado en menos tiempo usado para responder. Esto se debe a que el uso de pausas, para describir expresiones matemáticas, es más cercana a la manera tradicional que las personas utilizan para leer fórmulas. La segunda observación trata sobre el esfuerzo necesario para responder ambas encuestas. Resulta que para muchos de los participantes, completar las 15 expresiones con indicadores léxicos, les resultó más agotador, debido a que las verbalizaciones de las expresiones son más largas y que se debe estar atento en cada momento para saber cuándo termina una operación y cuándo comienza otra, ya que se dicen de manera explícita. Sin embargo, a pesar de la carga cognitiva adicional que agrega el uso de indicadores léxicos, también les dió a los participantes, o al menos a la mayoría, una mayor confianza en sus respuestas, pues estaban seguros que habían iniciado y terminado correctamente los distintos operadores.

La navegación de las fórmulas por teclado fue de mucha ayuda y bien valorada por los participantes. Ya que los participantes no estaban acostumbrados a utilizar herramientas como la presentada en el prototipo, no lograban captar toda la información de las páginas Web en un recorrido lineal. Gracias a que existía la alternativa de usar el teclado para avanzar y retroceder en las expresiones, los participantes pudieron analizar de una manera más profunda las expresiones, avanzando a su propio ritmo. Esto ayudó a que la cantidad de respuestas correctas aumentaran, aumentando el porcentaje de expresiones que el prototipo hizo accesible correctamente.

Finalmente, la evaluación da como mejor alternativa el uso de indicadores léxicos, debido a que obtuvo una mayor cantidad de respuestas acertadas, sin embargo, conlleva una carga cognitiva más grande, lo que hace que sea una alternativa más lenta de utilizar y más agotadora. La alternativa que obtuvo menos resultados positivos, en la evaluación, resultó ser la alternativa más cómoda para que los participantes pudieran acceder a las expresiones. En definitiva, ninguna alternativa se puede considerar completamente predominante sobre la otra. Quizá una buena alternativa sea agregar indicadores extralingüísticos, por ejemplo, reemplazar las pausas o paréntesis por un sonido que sea característico, como un *bip*, *click*, *etc*, permitiendo hacer una distinción entre las pausas inherentes al lenguaje natural y las pausas generadas para reflejar anidamiento en las expresiones matemáticas.

7. Conclusiones

La manera en la que se abordó el problema de hacer accesible las expresiones matemáticas, realizando un *preprocesado* de la página que se quiere hacer accesible, permite que la solución sea más general. Esta generalidad se ve reflejada de varias maneras; una de ellas es que la solución planteada es independiente del screen reader que los usuarios utilicen, lo que representa una gran ventaja, ya que los potenciales usuarios no deben adaptarse a esta solución (de hecho, si la solución necesitara que los usuarios cambiaran su manera de trabajo habitual, probablemente no la usarían), si no que la solución es lo suficientemente general para ser integrada con cualquier screen reader. Otra manera en la que la solución planteada es general, es cuando hablamos de las distintas codificaciones que existen para codificar expresiones matemáticas, pues si bien el prototipo utiliza *Content-MathML* para realizar la verbalización, existen varias herramientas disponibles para trasladar de otros formatos a *Content-MathML*, por lo que la codificación de las expresiones no debería ser una limitante.

La evaluación del prototipo demostró que es necesario definir delimitadores, que sean claros, para describir las expresiones matemáticas sin caer en errores debido al anidamiento de las expresiones, como los encontrados en la evaluación. De la evaluación también se puede concluir que esos delimitadores que son necesarios pueden ser tanto léxicos, prosódicos o incluso extralingüísticos. Definir delimitadores que sean claros permitirá que las interpretaciones que los usuarios hagan de las verbalizaciones tengan mayor probabilidad de ser correctas.

En otro punto, el estudio de las reglas para leer expresiones matemáticas fue bastante claro: No existe un conjunto de reglas establecido que permita verbalizar expresiones matemáticas. Esto se debe, principalmente, a que la manera en la que se leen las expresiones es muy dependiente del lugar geográfico, del tiempo y del nivel de educación de las personas. Si bien la semántica de cada expresión es única, la verbalización que la representa no lo es necesariamente. Pensar en definir ese conjunto de reglas y presentarlas como un estandar es una utopía prácticamente, por las razones anteriormente mencionadas. La mejor manera para obtener las verbalizaciones de expresiones matemáticas, basado en la experiencia del desarrollo de esta memoria, es asignar un template a la semántica de los operadores, ya que la semántica no varía sin importar la sintaxis que se muestre. El uso de templates es suficiente para realizar estas verbalizaciones, debido a que la lectura de expresiones matemáticas no está influenciada por conjugaciones, tiempos, adjetivos, etc.

En la Web, la cantidad de páginas que contienen expresiones matemáticas no es menor, en especial en sectores como educación y ciencia. Si cada una de esas expresiones matemáticas, comúnmente como imágenes, estuviesen acompañadas con su codificación,

sin importar mucho cuál fuese ésta, se podrían hacer accesibles grandes volúmenes de información matemática siguiendo el mismo esquema presentado en esta memoria. Esta misma idea dió como resultado una publicación [22] que permitió hacer accecible cerca del 60 % de las expresiones matemáticas de Wikipedia. Esta publicación viene a demostrar, que siguiendo el mismo esquema de funcionamiento mostrado en esta memoria y en su prototipo en particular, se pueden verbalizar grandes cantidad de fórmulas que ya están en la Web, por lo que no hay necesidad de editar documentos que ya están disponibles para que pueden ser verbalizados.

El objetivo presentado en un comienzo de esa memoria, el cual buscaba crear un prototipo para hacer accesibles las expresiones matemáticas, que además fuera gratuito y sin necesidad de instalación, se cumplió con la ayuda de *WebAnywhere*. Si bien es cierto, aun falta mucho que mejorar, tanto por el lado de esta memoria como por el lado del screen reader, se logró demostrar que es posible hacer accesible las expresiones matemáticas de una manera lo más general posible, considerando la manera en las que las personas leen las fórmuas matemáticas en un determinado lugar.

8. Trabajo Futuro

Este trabajo puede ser considerado como el inicio de un gran proyecto, ya que a partir de él pueden salir desafíos muy interesantes. Un de ellos es mejorar el screen reader utilizado en esta memoria, añadiéndole la posibilidad de controlar la velocidad, tono y timbre de la voz. Una buena manera de comenzar sería estudiando el uso de una recomendación de la W3C llamada SSML[26], la cual provee de un lenguaje para marcar pausas, velocidad, tono, etc, del habla.

Otra arista interesante de abordar en el futuro es mejorar el estudio que se realizó sobre las fórmulas matemáticas en la Web. Un estudio más acabado de las fórmulas ayudaría no sólo a saber cuáles son los operadores más utilizados, sino también permitiría saber cuáles son las áreas del conocimiento que más publican fórmulas. Además permitiría saber cuál es la manera más común de publicar expresiones matemáticas, de una manera cuantitativa y se podría proponer una serie de recomendaciones para publicar fórmulas de manera tal que la accesibilidad se vea garantizada.

Un punto importante que se puede abordar en un futuro es realizar evaluaciones con personas con problemas visuales. Sería muy interesante acotar el rango de fórmulas que se pueden verbalizar a aquellas incluídas en los programas de educación, para poder realizar evaluaciones con niños como los de *Coalivi*. Una herramienta de estas características sería útil en colegios, para realizar la prueba SIMCE e incluso para tomar la PSU. Si se pudiera completar esto, se podría avanzar a un nivel más alto e intentar hacer accesible los contenidos matemáticos de carreras universitarias, de manera tal que personas con problemas visuales y de escasos recursos tengan la opción de estudiar carreras cercanas a las matemáticas.

[26]Speech Synthesis Markup Language (SSML) Version 1.0, `http://www.w3.org/TR/speech-synthesis/`

Referencias

[1] G. K. A. Gupta and E. Pontelli, "Mathematics and accessibility: A survey," tech. rep., University of Texas at Dallas, 2007.

[2] J. P. Bigham, C. M. Prince, and R. E. Ladner, "Webanywhere: a screen reader on-the-go," in *Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A)*, W4A '08, (New York, NY, USA), pp. 73–82, ACM, 2008.

[3] "Mathematical markup language (mathml) version 3.0." `http://www.w3.org/TR/MathML`, 2010. Recomendación W3C.

[4] A. Nemeth, *The Nemeth Braille code for mathematics and science notation: 1972 revision*. Produced in braille for the Library of Congress, National Library Service for the Blind and Physically Handicapped by the American Printing House for the Blind., 1972.

[5] G. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *The Psychological Review*, vol. 63, pp. 81–97, March 1956.

[6] D. Archambault and M. Batusic, "The universal maths conversion library: an attempt to build an open software library to convert mathematical contents in various formats," in *Universal access in human-computer interaction*, (Las Vegas), 2005.

[7] M. Batusic and K. Miesenberger, "Labradoor: a contribution to making mathematics accessible for the blind," in *International Conference on Computers Helping People (ICCHP)*, (Vienna. Austria), Springer.

[8] M. M. N. W. A. K. K. H. S. Ohtake, N. Higuchi and H. Sato, "Mathbraille; a system to transform latex documents into braille," *SIGCAPH Comput. Phys. Handicap.*, no. 66, pp. 17–20, 2000.

[9] G. C. S. G. H. G. D. Wang, Q. Gupta and A. Karshmer, "Winsight: Towards completely automatic backtranslation of nemeth code," in *Universal Access in Human-Computer Interaction, Applications and Services.*, vol. 4556, Springer, 2007.

[10] N. E. B. S. W. Bernareggi, C. Jessel and M. Gut, "Lambda: A european system to access mathematics with braille and audio synthesis," in *Computers Helping People with Special Needs* (K. Miesenberger, J. Klaus, W. Zagler, and A. Karshmer, eds.), vol. 4061 of *Lecture Notes in Computer Science*, pp. 1223–1230, Springer Berlin / Heidelberg, 2006. 10.1007/11788713_176.

[11] A. I. Karshmer, G. Gupta, E. Pontelli, K. Miesenberger, N. Ammalai, D. Gopal, M. Batusic, B. Stöger, B. Palmer, and H.-F. Guo, "Uma: a system for universal mathematics accessibility," *SIGACCESS Access. Comput.*, no. 77-78, pp. 55–62, 2004.

[12] L. Chang, *Handbook for Spoken Mathematics: (Larry's Speakeasy).* University of California, Lawrence Livermore National Laboratory, 1983.

[13] D. Fitzpatrick, "Speaking technical documents: Using prosody to convey textual and mathematical material," in *ICCHP '02: Proceedings of the 8th International Conference on Computers Helping People with Special Needs*, (London, UK), pp. 494–501, Springer-Verlag, 2002.

[14] T. V. Raman and D. Ph., "Audio system for technical readings," tech. rep., Cornell University, 1994.

[15] P. Stanley and A. Karshmer, "Translating mathml into nemeth braille code," in *ICCHP*, pp. 1175–1182, 2006.

[16] H. Reddy and G. Gupta, "Dynamic aural browsing of mathml documents with voicexml," in *Human-computer interaction*, Lawrence Erlbaum and Associates, 2005.

[17] N. Soiffer, "Mathplayer: web-based math accessibility," in *In Assets '05: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pp. 204–205, ACM Press, 2005.

[18] I. Abu Doush and E. Pontelli, "Building a programmable architecture for non-visual navigation of mathematics: Using rules for guiding presentation and switching between modalities," in *UAHCI '09: Proceedings of the 5th International Conference on Universal Access in Human-Computer Interaction. Part III*, (Berlin, Heidelberg), pp. 3–13, Springer-Verlag, 2009.

[19] O. S. Team, "Xsl transform of mathml content to mathml presentation version 1.0." `http://www.orcca.on.ca/MathML/software/mmlctop2_0.zip`, 2003.

[20] G. Wilcock, "Pipelines, templates and transformations: Xml for natural language generation," in *Proceedings of the 1st NLP and XML Workshop*, pp. 1–8, 2001.

[21] K. Van Deemter, E. Krahmer, and M. Theune, "Real versus template-based natural language generation: A false opposition?," *Comput. Linguist.*, vol. 31, pp. 15–24, March 2005.

[22] L. Ferres and J. Fuentes, "Improving accessibility to mathematical formulas: The wikipedia math accesor," in *W4A 2011 - 8th International Cross-Disciplinary Conference on Web Accessibility*, 2011.