FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

LEITURA AUTOMÁTICA DE EXPRESSÕES MATEMÁTICAS – AUDIOMATH

HELDER FILIPE PATRÍCIO CABRAL FERREIRA

LICENCIADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
PELA FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

DISSERTAÇÃO SUBMETIDA PARA SATISFAÇÃO PARCIAL DOS REQUISITOS DO GRAU DE
MESTRE EM ENGENHARIA INFORMÁTICA

DISSERTAÇÃO REALIZADA SOB A SUPERVISÃO DE
PROFESSOR DOUTOR DIAMANTINO RUI DA SILVA FREITAS,
DO DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
DA FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

PORTO, SETEMBRO DE 2005

Página em branco.

Para a minha avó Antonieta,
pela sua simplicidade e pelo seu carinho.

Para o meu amigo Helder,
pelo seu carácter, amizade e coragem.

Página em branco.

Resumo

Nesta dissertação estuda-se o problema do ensino e da falta de acessibilidade dos documentos com conteúdos matemáticos na Internet por parte de pessoas com necessidades especiais, nomeadamente cegos e amblíopes.

A solução proposta nesta dissertação baseia-se na análise, interpretação e conversão de documentos com fórmulas matemáticas em MathML para uma forma escrita por extenso, sendo posteriormente lida com recurso a um conversor texto-fala.

Foram desenvolvidas três aplicações: AudioMathENGINE, AudioMathGUI e AudioMathWEB.

AudioMathENGINE consiste num motor de análise, interpretação e conversão de documentos com conteúdos matemáticos, e que possui módulos de processamento para: numerais, abreviações, acrónimos, referências de rede, assim como, reconhecimento automático dos elementos a converter, interpretação e conversão de expressões matemáticas, e implementação de mecanismos de navegação nas mesmas.

AudioMathGUI consiste numa aplicação gráfica que permite a aplicação do motor de análise, interpretação e conversão de expressões matemáticas, assim como a demonstração de um mecanismo de navegação contextualizado das fórmulas.

AudioMathWEB consiste num serviço público on-line de conversão de documentos com MathML, que se encontra disponível no sítio *web* da tese.

Neste trabalho apresenta-se também um estudo detalhado sobre a prosódia matemática e a forma como os humanos lêem e entendem a leitura das expressões matemáticas.

Ao longo da dissertação são apresentados resultados de trabalhos de pesquisa em diversas áreas, incluindo: engenharia, psicologia, matemática e linguística, com a finalidade de adquirir conhecimentos, conceitos e teorias, que permitem escolher as tecnologias e técnicas mais adequadas à concretização deste trabalho.

O trabalho de dissertação termina com um conjunto de testes objectivos e subjectivos para a determinação da qualidade dos resultados obtidos; assim como um conjunto de conclusões sobre esses mesmos resultados.

Mais informações sobre a dissertação, incluindo publicações e um serviço público de conversão de MathML, encontram-se disponíveis no sítio *web* da tese em: http://lpf-esi.fe.up.pt/~audiomath.

Página em branco.

Abstract

This dissertation focus itself the problem of teaching and the lack of accessibility in documents with mathematical contents over the Internet, aiming people with several types of visual impairment, like blindness.

The proposed solution in this thesis consists in the analysis, interpretation and conversion of documents containing mathematical formulae in MathML into a full written form, prepared to be read by a text-to-speech engine later on.

Three applications have been mainly developed: AudioMathENGINE, AudioMathGUI and AudioMathWEB.

AudioMathENGINE is the engine responsible for the analysis, interpretation and conversion of documents with mathematical contents. This engine includes processing modules for: numerals, abbreviations, acronyms, network references, and also, automatic recognition of certain elements to be converted, interpretation and conversion of mathematical expressions, and the implementation of a navigation mechanism for the existing mathematical expressions.

AudioMathGUI is a graphical user interface that allows the usage of the analysis, interpretation and conversion engine, and also the demonstration of the contextualized navigation mechanism for the mathematical expressions.

AudioMathWEB consists in a public on-line service, available on the thesis website, which converts documents with MathML.

On this thesis a detailed study is presented concerning math prosody and the way humans read and perceive mathematical expressions.

Along this dissertation several research results are presented in numerous areas, including: engineering, psychology, mathematics and linguistics, with the aim of acquiring knowledge, concepts and theories that guided in choosing technologies and techniques that are more adequate to this work's implementation.

The dissertation work ends with a set of objective and subjective tests about the quality of results; as well as a set of conclusions concerning these same results.

More information about the thesis, including publications and the public on-line conversion service of MathML, is available in the thesis website at: http://lpf-esi.fe.up.pt/~audiomath.

Página em branco.

Agradecimentos

Agradeço aos meus pais, Alfredo e Lurdes, por todo o esforço e dedicação ao longo da minha vida, sem os quais nunca teria sido possível realizar esta dissertação. Estou-lhes eternamente agradecido pela educação, amor e carinho.

À Ana por me fazer sentir especial, pelo amor e carinho, e pela inesgotável paciência e compreensão.

Ao professor Diamantino pela oportunidade de realização deste trabalho, pelo seu apoio e orientação durante a minha vida académica, e por sempre me ter mostrado como ser melhor profissional.

À Magda pelos seus esforços, amizade e partilha de conhecimentos na área da psicologia.

Aos meus colegas de investigação, Daniela, Luís, Filipe, António, Fernando, João Paulo e Avelino, pelo seu companheirismo. Um agradecimento especial à Maria João, cuja sugestão há 5 anos atrás deu origem a este projecto.

Ao Thomas Giegold, Thomas Hladschik e Robert Mauerer pelo seu companheirismo, motivação e boa disposição.

Às professoras Margarida e Maria do Rosário do DEEC pelo tempo disponibilizado para uma demonstração da aplicação e pelos seus comentários e palavras de motivação.

Ao meus colegas e amigos, Inês, Sofia e Jorge, pela disponibilidade e ajuda durante a fase de testes. Assim como à Maria da Luz Ribeiro, João Fernandes, Daniel Serra e Domingos Ferreira, pelo tempo disponibilizado para testar e comentar o AudioMath.

Aos meus colegas da Infineon e Critical Software pelo apoio, companheirismo e motivação.

Aos meus professores e colegas do Mestrado em Engenharia Informática, por terem contribuído tão eficazmente para o meu desenvolvimento como aluno, colega e profissional.

Página em branco.

Índice

Índice de figuras

Índice de tabelas

Página em branco.

Abreviaturas e Acrónimos

AKML	*AudioMath Knowledge Markup Language*
API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
CSS	*Cascading Style Sheets*
DLL	*Dynamic Link Library*
DOM	*Document Object Model*
DTB	*Digital Talking Book*
DTD	*Document Type Definition*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
ICEVI	*International Council for Education of People with Visual Impairment*
MathML	*Mathematical Markup Language*
OMS	*Organização Mundial de Saúde*
PDF	*Portable Document Format*
PERL	*Practical Extraction and Report Language*
RTF	*Rich Text Format*
SAPI	*Speech API*
SAX	*Simple API for XML*
SGML	*Standard Generalized Markup Language*
SSML	*Speech Synthesis Markup Language*
SVG	*Scalable Vector Graphics*
URI	*Universal Resource Identifier*
URL	*Universal Resource Locator*
W3C	*World Wide Web Consortium*
WCAG	*Web Content Accessibility Guidelines*
WYSIWYG	*What You See Is What You Get*
XHTML	*eXtensible HyperText Markup Language*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*
XSLT	*eXtensible Style Language Transformation*

Página em branco.

Glossário

API *Application Programming Interface* – Conjunto de funções e rotinas para uso do programador e do utilizador, de modo a permitir o uso da aplicação numa forma fácil, estável e consistente.

Applet Programa em Java que pode ser utilizado num documento on-line através do uso da etiqueta de marcação <OBJECT>.

Braille Trata-se de um processo de escrita em relevo para a leitura táctil, inventado por Luís Braille (1809-1852). Compõe-se de 63 sinais formados por pontos, a partir de um conjunto matricial idêntico a uma sena de dominó, ao alto. Com o Braille representam-se diversos alfabetos, assim como a matemática, a geometria, a química, a fonética, a informática e a música.

Browser Aplicação apropriada para aceder aos e interagir com os conteúdos da Internet. Navegador de páginas on-line. Por exemplo: Internet Explorer, Netscape, Opera ou Mozilla.

Corpus Compilação de documentos ou informações relativos a uma disciplina ou tema; conjunto finito de enunciados representativos de uma determinada estrutura.

GUI *Graphical User Interface* – Aplicação com capacidade de apresentação de conteúdos multimédia. A entrada de dados faz-se geralmente com recurso ao teclado e ao rato.

Leitor de ecrã Aplicação que acede ao texto apresentado visualmente no ecrã do computador, gerando a partir dessa informação saídas de voz sintetizada.

Markup Anotação, conjunto de etiquetas de marcação, ou complemento sintáctico do conteúdo, veiculando-lhe significado estrutural, tornando-o activo ou invocando conteúdo que o substituirá.

Parser Analisador de uma determinada linguagem, com capacidade para interpretar a sua estrutura, permitir a sua navegação e executar acções consoante a semântica dessa linguagem.

Plug-in Mecanismo ou aplicação externa que se adiciona a uma aplicação já existente, com o objectivo de lhe conferir novas funcionalidades. Por exemplo, o *MathPlayer* é um *plug-in* que permite a visualização de fórmulas matemáticas em MathML no Internet Explorer 6.0.

Standard Norma ou normalização de algo.

Unicode Norma internacional que permite codificar os caracteres de um documento, atribuindo-lhe um nome e um número. Suporta quase todas as línguas do mundo.

Web Sistema de acesso à informação, apresentada sob a forma de hipertexto, na Internet.

WYSIWYG *What You See Is What You Get* - Diz-se que uma ferramenta é do tipo WYSIWYG se o resultado final, aquando da visualização ou impressão, for exactamente igual aquilo que o criador viu durante a fase de produção do documento.

1 Introdução e Motivação

> "Engenharia não é meramente saber e ter conhecimento, como uma enciclopédia; engenharia não é meramente análise; engenharia não é meramente possuir a capacidade de descobrir soluções elegantes para problemas; engenharia é colocar em prática a arte organizada de forçar as mudanças tecnológicas... os engenheiros operam na interface entre a ciência e a sociedade..."[1] - *Dean Gordon Brown* (1907-1996).

Desde os primórdios dos tempos que o homem sente necessidade de se exprimir e de comunicar. A linguagem permite ao homem estruturar o seu pensamento, traduzir o que sente, registar o que conhece e comunicar-se com outros homens. É uma capacidade que pode ser exprimida de variadas formas, tais como a escrita, a leitura, o desenho, ou até as emoções.

No mundo da ciência e da comunidade técnica e cientifica, a comunicação, a educação, a distribuição e a partilha de conhecimento é realizada através de canais tais como: livros, revistas, artigos ou até mesmo da Internet desde o seu aparecimento em 1973 (1). A quantidade de documentos técnicos ou conteúdos científicos on-line tem crescido substancialmente nos últimos trinta anos, e o paradigma da educação começa a mudar em direcção da era digital (2). Em Junho de 2005 foi estabelecido que 14,6% da população mundial (cerca de 938 milhões de pessoas) utiliza activamente a Internet (3). No entanto, apesar do vasto acesso à informação electrónica, existe um grupo considerável de pessoas com necessidades especiais, para as quais este acesso é pouco facilitado e nalguns casos impossibilitado. São caso disso os cegos ou os amblíopes. Segundo um estudo realizado em 2002, o ICEVI[2] e a OMS[3] concluíram que 35 milhões de pessoas no mundo são cegas e outras 124 milhões de pessoas sofrem de um qualquer tipo de deficiência visual (4).

Geralmente o acesso à educação e à informação técnica é bastante limitado nas pessoas com deficiências visuais (5)(6)(7). Este acesso é feito, tipicamente, com recurso a documentação em Braille, gravações áudio de livros e revistas ou, em casos mais

[1] http://www.quotationspage.com/quotes/Dean_Gordon_Brown/
[2] *International Council for Education of People with Visual Impairment* - http://www.icevi-europe.org/
[3] Organização Mundial de Saúde - http://www.who.int/en/

raros, recorrendo a computadores e a tecnologias de fala, tais como síntese e reconhecimento da fala. Contudo, a percentagem de deficientes visuais que conhece o Braille é extremamente reduzida, e existem poucas publicações técnicas em formato áudio.

É também conhecido que a dificuldade para os cegos aumenta e o grau de acesso diminui, à medida que aumenta o nível de informação técnica no documento, apesar de existirem meios para que estes possam aceder a informações em formato digital (8).

A introdução de tecnologias assistidas e o uso do computador pessoal contribuíram para um melhoramento das condições de acesso à informação dos deficientes visuais, mas levantaram outras barreiras técnicas (9)(10). Aplicações como os leitores de ecrã são ainda pouco eficazes no que diz respeito à Internet e à leitura de conteúdos específicos como a matemática.

1.1 Objectivos da tese

Questões de "Como os cegos acedem à Internet e aos seus conteúdos técnicos?", "Porque o problema ainda não foi resolvido?" e "O que poderia ser feito?", influenciaram, conduziram e motivaram, o trabalho de investigação desta dissertação, assim como os desenvolvimentos concretizados na mesma, sendo que estes se focaram no acesso a conteúdos matemáticos.

Após uma pesquisa sobre o estado da arte da publicação destes conteúdos na Internet, e de acordo com a organização W3C[4], verificou-se que a solução mais acessível na publicação de conteúdos matemáticos consiste na utilização da linguagem de marcação matemática – MathML (11)(12). Sendo assim, esta foi a tecnologia escolhida como ponto de partida para o desenvolvimento desta dissertação.

A solução proposta e implementada consiste no uso de um conversor texto-fala para a reprodução oral de expressões matemáticas codificadas em MathML, uma vez que: o meio áudio encontra-se disponível e é acessível, existem diversos conversores texto-fala no mercado, e em princípio a matemática pode ser descrita oralmente. A interpretação das expressões matemáticas é realizada com recurso a um conjunto de algoritmos de análise, interpretação e conversão do MathML.

[4] W3C – *World Wide Web Consortium* – http://www.w3.org

O processo consiste na descrição por extenso de todos os elementos não-textuais de um documento (incluindo a matemática), após o qual é possível a aplicação de um conversor texto-fala para reproduzir oralmente os seus conteúdos.

A solução proposta apresenta diversos desafios, tais como: a interpretação inteligente de uma expressão matemática, o estudo prosódico da leitura da matemática, e a implementação de um sistema de navegação intra-fórmula contextualizado.

As aplicações, do trabalho desenvolvido no âmbito desta dissertação, são inúmeras e em diversas áreas: desde a integração de motores de conversão texto-fala, à integração em sistemas informáticos promovendo a acessibilidade, passando pela educação e pela formação.

1.2 Organização da dissertação

A presente dissertação encontra-se organizada em treze capítulos e oito anexos. No capítulo 1 apresenta-se uma introdução que contextualiza o trabalho desenvolvido nesta dissertação, com a temática do acesso à informação técnica na Internet, as necessidades especiais, e a educação. São também definidos os objectivos e os desafios que se apresentaram na realização deste trabalho, assim como uma breve descrição da organização desta dissertação.

No capítulo 2 é feita referência ao estado da arte no que diz respeito à produção e publicação de documentos com conteúdos matemáticos na Internet; assim como uma breve descrição dos projectos mais relevantes na área da leitura das expressões matemáticas. É referida a linguagem MathML como a linguagem de marcação matemática usada no trabalho desenvolvido durante esta dissertação.

Sendo a linguagem MathML uma linguagem XML, no capítulo 3 é feita uma breve referência aos conceitos básicos da *eXtensible Markup Language* (XML), introduzindo assim o capítulo seguinte.

No capítulo 4 é abordada a linguagem de marcação matemática (MathML), onde são apresentados os diferentes conjuntos de etiquetas de marcação da linguagem, a importância do Unicode nos documentos matemáticos, e a integração desta linguagem com outras tecnologias de Internet, possibilitando assim a publicação on-line de documentos com conteúdos matemáticos. Neste capítulo apresentam-se também as conclusões resultantes de um estudo relacionado com o uso do MathML para a interpretação oral das fórmulas matemáticas, realizado no âmbito desta dissertação.

Uma vez que a solução apresentada nesta dissertação inclui o recurso à tecnologia de síntese da fala, no capítulo 5 é feita uma breve introdução aos conversores texto-fala e seu modo de funcionamento; assim como uma breve referência a linguagens de marcação usadas em tecnologias de fala.

O facto do trabalho desenvolvido contextualizar-se numa área onde os seus algoritmos estão intrinsecamente relacionados com a aplicação de expressões regulares, resulta numa introdução aos conceitos de análise, interpretação e conversão de padrões num texto, apresentado no capítulo 6, e onde são discutidos os conceitos básicos dos processadores de linguagens, expressões regulares e autómatos finitos.

O capítulo 7 é dedicado a uma das aplicações desenvolvidas durante esta dissertação, denominada AudioMathENGINE. Neste capítulo é apresentada a linguagem de marcação AKML (*AudioMath Knowledge Markup Language*) usada na configuração e definição de regras, a arquitectura da aplicação, e a descrição detalhada dos algoritmos de análise, interpretação e conversão. São também discutidos detalhadamente: o módulo de reconhecimento automático (responsável pela conversão dos documentos) e o módulo de expressões matemáticas (responsável pela interpretação e conversão das fórmulas em MathML).

No capítulo 8 é apresentada mais uma das aplicações desenvolvidas no âmbito desta dissertação, denominada AudioMathGUI. Neste capítulo é abordada a arquitectura da aplicação, brevemente descritas as suas funcionalidades e exemplificada a integração de diversas tecnologias que a compõem.

O capítulo 9 descreve a última das aplicações desenvolvidas neste trabalho de dissertação, denominada AudioMathWEB, onde são descritos os serviços colocados publicamente disponíveis: um editor de expressões matemáticas em MathML e um serviço de acessibilidade para conversão de documentos técnicos (que usem MathML) em texto por extenso, permitindo assim o acesso à informação através de leitores de ecrã.

Após a descrição das tecnologias usadas durante o trabalho de dissertação e a apresentação detalhada de três aplicações desenvolvidas na tese, são apresentados no capítulo 10 os resultados e conclusões de um estudo relacionado com a leitura das fórmulas matemáticas. Neste capítulo é feita uma breve introdução à fala humana, seguido da análise de sinais de fala e extracção de regras e conclusões relacionadas com a prosódia da matemática.

Uma vez apresentadas as soluções desenvolvidas no trabalho e diversos estudos de prosódia da matemática, resta abordar o tema da navegação de expressões matemáticas. Este é apresentado no capítulo 11 onde também são feitas breves considerações sobre percepção, cognição e memória.

No capítulo 12 são apresentados resultados e conclusões de diversos testes objectivos e subjectivos realizados às aplicações desenvolvidas.

O capítulo 13 é dedicado às conclusões e perspectivas de trabalho futuro, onde é realizado um resumo do trabalho desenvolvido e possíveis perspectivas de continuação.

Nos anexos descreve-se informação complementar ao trabalho desenvolvido na dissertação. Nos Anexos A e B são apresentados exemplos de fórmulas matemáticas codificadas em *MathML Presentation Markup* e *MathML Content Markup*, respectivamente.

No Anexo C é disponibilizado o XML *Schema* da linguagem interna do AudioMath – AKML.

O Anexo D apresenta diversas fórmulas matemáticas utilizadas na demonstração dos algoritmos de análise, interpretação e conversão do AudioMathENGINE.

No Anexo E é descrito o *corpus* matemático utilizado na gravação de expressões matemáticas, para análise de sinais de fala e extracção de conclusões sobre prosódia.

Nos Anexos F, G e H são disponibilizados os questionários de interpretação escrita, interpretação oral e navegação, usados durante a fase de testes e validação de resultados.

Página em branco.

2 Estado da Arte – Produção, Publicação e Leitura de Conteúdos Matemáticos na Internet

A palavra comunicação vem do latim "*comunicare*" e significa "transmitir". A comunicação entre humanos visa assim a transmissão de conhecimentos, ideias e emoções em relação a uma particularidade ou a um conjunto de particularidades.

A publicação e distribuição, física ou electrónica, de inúmeros artigos científicos em todo o mundo é uma forma comum de divulgar o conhecimento, promover a educação e fomentar a investigação e a pesquisa. A linguagem escolhida para o efeito é muitas vezes a linguagem matemática, uma vez que é universal e não ambígua.

Neste capítulo pretende-se introduzir os conceitos e técnicas actuais na produção e publicação de conteúdos técnicos e científicos on-line, assim como a leitura dos mesmos, com especial foco para o caso particular das expressões matemáticas.

2.1 Produção e Publicação de Conteúdos Técnicos e Científicos na Internet

A publicação de documentos científicos que incluem expressões matemáticas é extremamente exigente. O aparecimento do sistema *TeX* (13), desenvolvido por Donald Knuth, introduziu um novo conceito de publicação e formatação de documentos, resolvendo a maioria dos problemas dos documentos impressos. Posteriormente surgiram editores do tipo WYSIWYG (*What You See Is What You Get*[5]), como por exemplo o *Microsoft Word* (14).

Com o desenvolvimento da Internet começaram a surgir os primeiros documentos on-line formatados por linguagens de marcação, como o HTML (15). São os denominados hiper-textos. No entanto, o HTML não permite o uso da linguagem matemática directamente no documento. Assim sendo, surgiram alternativas para a publicação de expressões matemáticas na Internet, tais como:

- Uso de imagens a representarem expressões matemáticas;
- Uso de formatos alternativos como: PDF (16), TeX, Postscript (17), Microsoft Word ou RTF (18);
- Uso de *applets* de Java (19) para gerar representações gráficas de equações;
- Uso de *plug-ins* para gerar representações gráficas de equações;

[5] "O que se vê é o que se obtém".

- Uso de HTML e Fontes de Símbolos;
- Uso de XHTML (20) e CSS (21);
- Uso de linguagens de marcação: MINSE (22), MathML (23), SVG (24) e outros.

2.1.1 Uso de imagens na representação de expressões matemáticas

A maioria de documentos técnicos é publicada na Internet sob a forma de documentos HTML ou XHTML, em que as fórmulas matemáticas existentes nos documentos são transformadas e convertidas em imagens. Esta é a abordagem de programas como, por exemplo, `LaTeX2HTML` (25) e `TeX4ht`[6] (26).

No entanto, este processo tem duas grandes desvantagens: a fraca resolução das imagens, e a introdução de um problema de acessibilidade nas mesmas, uma vez que estas terão de possuir uma descrição que permita ler a expressão matemática (12) para que possam ser interpretadas por um cego. A grande vantagem é que o utilizador não necessita de instalar um mecanismo externo (*plug-in*) no navegador de páginas *web* (*browser*), uma vez que o HTML é suportado nativamente.

2.1.2 Uso de formatos alternativos: PDF, TeX, Postscript, Word ou RTF

É frequente a publicação de documentos técnicos sob a forma de ficheiros numa página on-line, disponíveis para descarregamento (*download*) por qualquer pessoa. Alguns dos formatos utilizados são: PDF, TeX, Postscript, Microsoft Word ou RTF.

No entanto, a visualização destes documentos só é possível se o navegador de páginas on-line possuir mecanismos visualizadores externos para cada tipo de formato, ou então usando um editor próprio após o descarregamento do documento.

Existem duas grandes desvantagens no uso destes formatos alternativos: a necessidade de possuir um mecanismo externo instalado, e a falta de acessibilidade na leitura destes documentos, uma vez que a sua navegação depende de um visualizador onde não existe um controlo eficiente por parte de tecnologias assistidas, como por exemplo os leitores de ecrã. Contudo, a Adobe tem feito alguma pesquisa e investimento com o objectivo de tornar o formato PDF mais acessível (27).

As vantagens deste método consistem no facto de estes formatos se encontrarem largamente difundidos e aceites pelos utilizadores; assim como a qualidade de impressão dos mesmos.

---

[6] Recentemente foi implementado um módulo de MathML no TeX4ht, como alternativa às imagens.

2.1.3 Uso de *applets* de Java para gerar representações gráficas de equações

Um *applet* é um programa escrito em Java que pode ser incluído num documento HTML, conferindo interactividade e dinâmica aos seus conteúdos. Esta abordagem é adoptada pelo `WebEQ` (28), constituído por um servidor de *applets* que recebe como parâmetro um conjunto de marcações que representam uma expressão matemática.

Actualmente o `WebEQ` recebe expressões matemáticas especificadas numa linguagem de marcações denominada `WebTeX` (29). Os blocos de `WebTeX` são embebidos no documento HTML, que são transformados nas suas representações gráficas pelo WebEQ. Este processo possui as mesmas vantagens e desvantagens que o uso de imagens que contêm expressões matemáticas. No entanto, o uso de um *applet* torna o acesso ao documento mais lento.

2.1.4 Uso de *plug-ins* para gerar representações gráficas de equações

O uso de mecanismos externos (*plug-ins*) para gerar representações gráficas de expressões matemáticas possui características idênticas ao uso de applets de Java. Um exemplo desta abordagem é o `TechExplorer` (30) da IBM e o `MathPlayer` (31) da Design Science. Ambos os programas suportam a linguagem de marcação da W3C para expressões matemáticas: MathML (*Mathematical Markup Language*).

2.1.5 Uso de HTML e fontes de símbolos

Esta abordagem tem como objectivo fornecer o documento HTML o mais simples possível, comprometendo geralmente o aspecto estético do documento. Um exemplo desta abordagem é o `translator TtH`[7] (32) que converte um documento codificado em TeX para HTML utilizando: fontes de símbolos suportadas pelo HTML 4.0 para representar glifos matemáticos, e etiquetas de marcação de HTML 3.2 para estruturar as fórmulas matemáticas. A grande vantagem deste método encontra-se na simplicidade e na rapidez. A grande desvantagem está na falta de acessibilidade da informação que é produzida em HTML, uma vez que são usadas tabelas para a criação da estrutura das expressões.

[7] Recentemente foi implementado um módulo que converte para MathML.

2.1.6 Uso de XHTML e CSS

Um dos métodos recentemente usados para visualizar expressões matemáticas em documentos on-line consiste no uso de XHTML com CSS.

Este processo possui algumas desvantagens: existe uma acentuada curva de aprendizagem das linguagens de estilo, nem todos os navegadores de páginas on-line (*browsers*) suportam todas as versões de CSS, e a complexidade das equações representadas é limitada, devido ao actual suporte do CSS. As vantagens são: a simplicidade, a rapidez e o facto de não ser necessário instalar nenhum mecanismo externo (*plug-in*) adicional. Actualmente a W3C encontra-se a estudar o uso do CSS 3 para as fórmulas matemáticas, nomeadamente a nível de interpretação oral (33).

2.1.7 Uso de Linguagens de Marcação: MINSE, MathML, SVG e outros

Existem diversas linguagens de marcação para a publicação de expressões matemáticas na Internet. Alguns exemplos são: MINSE, MathML[8], WebTeX, TeX, OpenMath (34) e SVG. Contudo, a maioria delas encontra-se ainda numa fase de experimentação e com falhas em diversas áreas.

A aplicação de MINSE consiste no recurso a um mecanismo externo que converte a notação da fórmula matemática, codificada em MINSE, para uma imagem, o que já foi referido provoca problemas de acessibilidade.

O uso de SVG para a representação gráfica de expressões matemáticas também provoca problemas de acessibilidade, uma vez que as expressões deixam de possuir semântica para apresentarem apenas um conjunto de linhas, polígonos e caracteres.

Apenas o TeX, o MathML e o OpenMath têm conseguido proliferar junto da comunidade científica. Actualmente, as alternativas de conversão de documentos em TeX e LaTeX passam pela transformação e produção de documentos on-line XHTML com MathML. O mesmo se passa com o OpenMath que é uma codificação publicada e incluída na codificação MathML.

Sendo assim, o futuro da publicação on-line de documentos técnicos e científicos com expressões matemáticas passa pela linguagem de marcação matemática – MathML. Por esse motivo, esta foi a tecnologia eleita para o desenvolvimento do trabalho realizado no âmbito desta dissertação.

[8] A linguagem de marcação matemática MathML é abordada num capítulo próprio desta dissertação.

2.1.8 Programas para produção e publicação

Actualmente, os dois grandes formatos de produção de conteúdos técnicos e científicos são: o LaTeX/TeX e o Microsoft Word.

A publicação destes materiais, de uma forma directa, na Internet exige uma conversão e transformação dos documentos originais em documentos on-line usando HTML ou XHTML (discutido nas secções anteriores). Concluiu-se também que de entre todas as tecnologias apresentadas, o MathML aparece como a referência para a publicação de expressões matemáticas na Internet. Sendo assim, de seguida são apresentados exemplos de aplicações que permitem a produção de conteúdos matemáticos em MathML:

- `MathType` (35)
    - Propriedade da *DesignScience*.
    - Versão profissional do conhecido *Microsoft Equation Editor* e como tal, integrado no Microsoft Word.
    - Produz documentos com os formatos: HTML+Imagens e XHTML+ MathML.
    - Os documentos gerados por esta aplicação necessitam do mecanismo externo MathPlayer, para poderem ser correctamente visualizados no Internet Explorer 6.0.
    - Foi a aplicação maioritariamente utilizada nas experiências realizadas no âmbito desta dissertação.
- `WebEQ`
    - Propriedade da *DesignScience*.
    - Subdivide-se nos produtos: *WebEQ Editor* e *WebEQ Publisher*.
    - Produz documentos com os formatos: HTML+Imagens, HTML+Applets e XHTML+MathML.
- `Mathematica` (36) e `Publicon` (37)
    - Propriedade da *Wolfram Research*.
    - Produzem documentos com os formatos: XHTML+MathML, XML, TeX e HTML.
- `Scientific Word` (38)
    - Propriedade da *Mackichan Software*.
    - Produz documentos com os formatos: XHTML+MathML e LaTeX.

- `SciWriter` (39)
    - Propriedade da *Soft4Science*.
    - Produz documentos com os formatos: XHTML+MathML e LaTeX.

A Tabela 2.1 compara as aplicações enumeradas anteriormente, em relação ao: tipo de linguagem de marcação de MathML usado; codificação MathML em fórmulas individuais ou documentos completos; e tipo de aplicação (comercial ou gratuita).

Tabela 2.1 - Comparação entre editores de MathML.

Produto	**Presentation MathML**	**Content MathML**	**Fórmulas individuais**	**Documentos completos**	**Comercial?**
MathType	Sim	Não	Sim	Não	Sim
WebEQ	Sim	Sim	Sim	Não	Sim
Mathematica	Sim	Sim	Sim	Sim	Sim
Publicon	Sim	Sim	Sim	Sim	Sim
Scientific Word	Sim	Não	Não	Sim	Sim
SciWriter	Sim	Não	Não	Sim	Sim

Existem também soluções para os autores que produzem documentos técnicos em LaTeX ou TeX. Alguns exemplos são:

- `Itex2mml` (40)
    - Programa em C, gratuito, que converte fórmulas matemáticas escritas num dialecto de TeX (IteX), em MathML.
    - Produz documentos no formato: XHTML+MathML.
    - Usa *MathML Presentation Markup*.
- `TeX2MML` (41)
    - Conversor on-line que transforma equações matemáticas codificadas em LaTeX para MathML.
    - Usa *MathML Presentation Markup*.

- `TeX4ht`
    - Converte documentos em LaTeX ou TeX, em documentos HTML.
    - Produz documentos com os formatos: HTML+Imagens, XHTML, XHTML+MathML.
    - Usa *MathML Presentation Markup*.
- `TtM` (42)
    - Programa comercial que converte documentos codificados em LaTeX e TeX, em documentos com o formato XHTML+MathML.
    - Usa *MathML Presentation Markup*.
- LaTeX2HTML
    - Programa que converte documentos codificados em LaTeX, em documentos HTML.

Recomenda-se a consulta da lista oficial de aplicações que processam MathML (43), na página on-line da linguagem de marcação matemática – MathML. O trabalho desenvolvido no âmbito desta dissertação também se encontra aí listado (44).

2.2 Leitura de Expressões Matemáticas

Apesar da existência de diversas tecnologias, como por exemplo Braille ou leitores de ecrã, que permitem um cego "ler" um documento on-line, poucas são as que conseguem processar conteúdos matemáticos. Na realidade existem escassos projectos que abordam a temática da leitura de expressões matemáticas. Tanto quanto se conhece, os únicos trabalhos de referência nesta área são:

- ASTER de T.V. Raman;
- MathTalk de Robert Stevens;
- MathSpeak de Abraham Nemeth;
- MathGenie de Arthur Karshmer;
- MathPlayer da Design Science;
- Projecto LAMBDA;
- E por fim, o projecto de investigação AudioMath.

2.2.1 ASTER

ASTER (45) (*Audio System for Technical Readings*) é uma aplicação desenvolvida por T.V. Raman em 1994 durante a preparação da sua tese de doutoramento (46) cuja finalidade consiste na reprodução oral de documentos com conteúdos matemáticos em LaTeX. Trata-se de uma aplicação LINUX que usa o programa Emacs[9]. O ASTER é constituído por três componentes:

- Analisador de LaTeX – cuja finalidade é a criação de uma representação interna do documento e das fórmulas matemáticas, para facilitar a manipulação e a interpretação;
- Linguagem de marcação áudio – AFL (*Audio Formatting Language*) – cuja finalidade é a marcação do texto convertido para a reprodução áudio;
- Navegador – usado para navegação nos documentos.

O trabalho desenvolvido por Raman é ainda hoje considerado uma referência na leitura das expressões matemáticas.

2.2.2 MathTalk

MathTalk (47)(48) é uma aplicação desenvolvida por Robert Stevens em 1996 durante a preparação da sua tese de doutoramento (49) cuja finalidade consiste na aplicação de conceitos relacionados com a memória e o controlo de fluxo da informação, para a reprodução áudio de expressões algébricas codificadas em LaTeX.

O trabalho realizado por Stevens inclui o estudo da prosódia de expressões de álgebra simples e a implementação de um método de auxílio à navegação. Este método consiste numa leitura de alto nível da fórmula matemática, antes da leitura detalhada.

Apesar das fórmulas matemáticas terem sido estudadas de forma simplista, o trabalho de Stevens foi um dos primeiros a apresentar resultados sobre a prosódia da matemática.

[9] GNU Emacs - http://www.gnu.org/software/emacs/emacs.html

2.2.3 MathSpeak

O projecto `MathSpeak` (50) foi inicialmente desenvolvido por Abraham Nemeth, o inventor do código Braille Nemeth (51)(52), em 2000. O objectivo do `MathSpeak` consiste na especificação de um conjunto de regras de discurso oral para a leitura de fórmulas matemáticas, facilitando a transcrição de documentos matemáticos para Braille. Usando o dialecto de MathSpeak, a fracção $1/2$ seria lida como: "`B-frac 1 over 2 E-frac`", sendo que `B-frac` significa "*begin-fraction*" (inicio de fracção) e `E-frac` significa "*end-fraction*" (fim de fracção).

Em 2004, foi iniciado um projecto de integração do `MathSpeak` na especificação DTB (*Digital Talking Books*) (53) onde foi desenvolvido um conversor de MathML para a linguagem MathSpeak.

2.2.4 MathGenie

Os estudos realizados por Karshmer e sua equipa (54)(55)(56)(57) baseiam-se na forma como a matemática é percepcionada pelo ser humano, e na interacção multimodal para utilizadores com necessidades especiais. Com base nesses projectos de investigação, Karshmer disponibilizou, em Março de 2005, uma aplicação denominada `MathGenie` (58), cujos objectivos consistem na:

- leitura não ambígua de expressões matemáticas usando uma notação normalizada e publicada pelo projecto MathSpeak;
- identificação de níveis semânticos numa expressão matemática;
- possibilidade de navegação entre esses níveis.

A aplicação `MathGenie`, desenvolvida para o sistema WINDOWS, recebe como fonte de informação uma expressão matemática em MathML e converte-a na notação MathSpeak. Posteriormente esta é reproduzida oralmente através de um conversor texto-fala.

2.2.5 MathPlayer

`MathPlayer` é um mecanismo externo (*plug-in*) desenvolvido pela *Design Science*, em 2002, com o objectivo de permitir a visualização de expressões matemáticas codificadas em *MathML Presentation Markup*, no Internet Explorer 6.0.

Em 2004 surge o `MathPlayer` 2.0 (59) com capacidades de interpretação oral de uma fórmula matemática através de um conversor texto-fala, assim como um sistema de ampliação visual das expressões matemáticas. Esta aplicação faz uso do *Microsoft Standard Accessibility Interface* (MSAA), o que lhe permite a integração com diversos leitores de ecrã (*screen-readers*).

O `MathPlayer` é actualmente um dos principais mecanismos de visualização e leitura de fórmulas matemáticas em MathML. Encontram-se em desenvolvimento mecanismos de navegação e pesquisa de fórmulas matemáticas.

2.2.6 Projecto Lambda

`LAMBDA` *(Linear Access to Mathematics for Braille Device and Audio synthesis*) (60)(61) é um projecto financiado pela União Europeia e com o objectivo de criar uma ferramenta de escrita e leitura de textos matemáticos para cegos, com inicio em 2002 e fim em 2005. Até ao momento da escrita desta dissertação, ainda não tinham sido publicadas as conclusões da investigação.

As aplicações desenvolvidas no âmbito deste projecto utilizam um código intermédio, denominado código Lambda, para transformar as expressões matemáticas em MathML ou LaTeX, e vice-versa. São utilizados como periféricos dispositivos Braille e dispositivos de síntese de fala.

2.2.7 Projecto AudioMath

Em 2003, na conclusão da licenciatura e como projecto final de curso do autor desta dissertação, foi desenvolvida uma ferramenta com capacidade de conversão de documentos de texto em voz. A esta ferramenta deu-se o nome de `AudioMath` (62)(63)(64)(65)(66).

Neste projecto foram adquiridos conhecimentos de: metalinguagens, XML, MathML, conceitos de expressões regulares e noções de autómatos.

A aplicação desenvolvida consistiu numa biblioteca (DLL[10]) do tipo ActiveX integrada no módulo de pré-processamento de um conversor texto-fala. Esta DLL era capaz de interpretar parcialmente expressões matemáticas simples, codificadas em *MathML Presentation Markup*, convertendo-as para texto por extenso, sendo posteriormente sintetizadas em fala.

Tanto quanto se conhece, `AudioMath` foi a primeira aplicação a usar MathML para a interpretação oral das fórmulas matemáticas.

No entanto, alguns dos objectivos ficaram por cumprir, como por exemplo: a implementação de um sistema de navegação das fórmulas matemáticas, o estudo da prosódia da leitura das expressões matemáticas, o suporte completo de MathML, e o aperfeiçoamento dos algoritmos de conversão utilizados.

Esta dissertação é a continuação deste trabalho desenvolvido pelo autor no seu projecto final de curso.

[10] DLL – *Dynamic Link Library*

Página em branco.

3 Introdução à eXtensible Markup Language – XML

XML, *Extensible Markup Language* (67), é uma linguagem de marcação e estruturação que descreve uma classe de objectos de dados (documentos XML) e parcialmente a forma como estes são processados.

O XML é uma forma restrita do SGML, *Standard Generalized Markup Language* [ISO 8879] (68), que foi desenvolvido pela W3C, aparecendo pela primeira vez em 1996 e tendo sido publicado como recomendação em 21 de Fevereiro de 1998 (versão 1.0) e mais tarde, em 2004, corrigido e aperfeiçoado. Esta metalinguagem foi desenvolvida com os seguintes objectivos:

- Ser directamente utilizada via Internet e suportar uma vasta variedade de aplicações;
- Ser compatível com o SGML;
- Permitir a criação de documentos XML que são facilmente produzidos, processáveis e minimamente legíveis por um ser humano (*human-readable*).

Existem diversas aplicações derivadas do XML, tais como:

- *DocBook* – descreve documentos, tais como: livros, manuais e artigos;
- *Chemical Markup Language* (CML) – descreve a estrutura de moléculas;
- *Wireless Markup Language* (WML) – descreve dados de redes sem fios;
- *Scalar Vector Graphics* (SVG) – descreve gráficos bidimensionais;
- *Mathematical Markup Language* (MathML) – descreve expressões matemáticas;
- *Extensible Style Language* (XSL) – permite a formatação e transformação de documentos XML;
- *Extensible Hypertext Markup Language* (XHTML) –transição entre HTML e XML.

Neste capítulo pretende-se apresentar sumariamente alguns conceitos básicos sobre o XML e tecnologias que lhe estão associadas, tais como o DTD e o XML Schema. No entanto, não é objectivo deste capítulo apresentar todas as características e propriedades destas tecnologias. Para tal, recomenda-se a leitura de (69) ou (70).

3.1 Documentos XML

Apesar das facilidades que o XML oferece a um utilizador, na definição de qualquer conjunto de etiquetas de marcação, existem um conjunto de regras que têm de ser seguidas na produção de um documento XML, para que este seja considerado como *bem formado*.

Ao contrário do HTML[11], se um documento XML não for considerado bem formado, este não é analisado, nem interpretado ou visualizado, emitindo uma mensagem de erro. Os analisadores (*parsers*) de XML são intolerantes com a sintaxe do documento.

Um documento XML pode ser um simples ficheiro de texto, ou um documento gerado dinamicamente, que contém um conjunto de marcas dispostas de acordo com algumas regras. Todos os documentos XML possuem um cabeçalho do tipo:

```
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”yes|no”?>
```

Figura 3.1 - Cabeçalho de documento XML.

A restante parte de um documento XML é composta por texto e etiquetas de marcação que descrevem a estrutura do documento. O atributo `encoding`, na Figura 3.1, tem como objectivo informar o analisador de XML que o documento contém caracteres especiais da língua portuguesa. Desta forma, pode ser evitado o uso de códigos Unicode (71). Na Figura 3.2 é exemplificado o uso de um caracter Unicode num documento XML.

```
&#<código do caracter em Unicode>;
<?xml version=”1.0” encoding=”ISO-8859-1” standalone=”yes|no”?>
<doc>
  A palavra “matemática” é representada por “matem&#225;tica”.
</doc>
```

Figura 3.2 – Uso de caracter Unicode no XML.

[11] HTML – *Hypertext Markup Language*

A Figura 3.3 exemplifica a forma como são declarados os comentários num documento XML.

```
<!-- Isto é um comentário -->
```

Figura 3.3 – Exemplo de comentário no XML.

Os comentários não devem ser colocados na primeira linha do documento, nem dentro de marcas ou englobando marcas em zonas de comentário. Também não se deve usar os caracteres "`--`" dentro de uma linha de comentário, para que estes não se confundam com o início e fim de comentário.

A Figura 3.4 exemplifica o uso de um bloco do tipo CDATA. Estes blocos de texto são usados quando se pretende introduzir um texto puro num documento XML, sem que este corra perigo de ser interpretado pelo analisador de XML.

```
<![CDATA[ ...Texto que não é interpretado.... ]]>
```

Figura 3.4 – Exemplo de CDATA num documento XML.

Um documento XML é composto por uma ou mais unidades lógicas (*entidades*). Uma *entidade* pode ser vista como uma representação abreviada de um determinado conteúdo. Existem entidades externas e internas (72), mas não serão abordadas nesta dissertação.

```
<!ENTITY nome "conteúdo da entidade">
```

Figura 3.5 - Declaração de entidade no XML.

As etiquetas de marcação, ou *elementos*, de um documento XML têm de respeitar as seguintes regras:

- Têm de ser sempre colocadas aos pares, definindo uma zona demarcada entre um início (`<marca>`) e um fim (`</marca>`). Existe uma excepção nas marcas vazias onde pode não ser colocado um par, fechando-se assim a marca de início (`<marca/>`).
- O nome das marcas não pode incluir espaços em branco, tem de começar por letras ou pelo caracter *underscore* "_" e não pode incluir caracteres que não sejam letras, números, *underscores*, hífenes ou pontos.

- Os nomes em XML são sensíveis ao uso de maiúsculas ou minúsculas (*case-sensitive*), isto é, "`Marca`" é diferente de "`marca`".
- Tem de existir uma estrutura em árvore começando por uma `etiqueta de raiz`, e as marcas não se devem entrelaçar.

```
<marca raiz>
   <marca1>
     <marca2>...</marca2>...<marca3>...</marca3><marca4/>
   </marca1>
</marca raiz>
```

Figura 3.6 – Exemplo de estrutura em árvore XML.

As etiquetas de marcação XML podem possuir *atributos*. Estes estão sempre associados a uma marca e de alguma forma qualificam-na ao funcionarem como propriedades da mesma. Os valores dos atributos devem estar delimitados por aspas "`″″`" ou plicas "`′′`".

```
<marca1 atributo1=″texto sem aspas″>
  <marca2 atributo2=′texto com “aspas” ′>...</marca2>
</marca1>
```

Figura 3.7 - Exemplo de atributos em XML.

A Figura 3.8 exemplifica um documento XML.

```
<?xml version=″1.0″ encoding=″ISO-8859-1″ standalone=″yes″?>
<listagem>
  <universidade local=″Porto″ url=″http://www.up.pt″>
    <faculdade nome=″Engenharia″ sigla=″FEUP″ />
    <faculdade nome=″Ciências″ sigla=″FCUP″ />
  </universidade>
  <universidade local=″Lisboa″ url=″http://www.ul.pt″>
    <faculdade nome=″Direito″ sigla=″FDUL″ />
  </universidade>
</listagem>
```

Figura 3.8 - Exemplo de um documento XML.

3.2 Document Type Definition - DTD

Foi já referido que o XML é uma metalinguagem que permite descrever outras linguagens. Essa descrição contém a enumeração das etiquetas de marcação, bem como as relações entre elas. Em XML, uma das formas de descrição de uma linguagem pode ser realizada pela elaboração de um documento denominado de "Definição de Tipo de Documento" (*Document Type Definition* – DTD). Um DTD pode estar incluído no próprio documento XML ou ser um ficheiro exterior, podendo ser partilhado por múltiplos documentos que o utilizam como referência de estrutura documental.

Um DTD começa sempre pela instrução XML "<!DOCTYPE", seguido do nome do elemento principal do documento. Entre os caracteres "[" colocam-se as definições de todos os elementos, entidades e atributos que compõem o DTD.

```
<!DOCTYPE TPML [ <!ELEMENT MARCA(#PCDATA)> ]>
é usado no documento:
<MARCA>...</MARCA>
```

Figura 3.9 - Exemplo de DOCTYPE num documento XML.

Um DTD é construído com base na regra de que tudo o que não é explicitamente autorizado, é "proibido" de ser usado em documentos XML. A Figura 3.10 demonstra a declaração de elementos num DTD.

```
<!ELEMENT elemento ANY> - conteúdo do elemento por especificar.
<!ELEMENT elemento (#PCDATA)> - conteúdo do elemento tem de ser
do tipo #PCDATA.
<!ELEMENT elemento (filho1,filho2,filho3)> - elemento possui
filhos.
<!ELEMENT equipa(nome,jogador+)> - sinal + indica uma ou mais
ocorrências de jogador.
<!ELEMENT equipa(nome,jogador*)> - sinal * indica zero ou mais
ocorrências de jogador.
<!ELEMENT estado_civil(solteiro|casado)> - sinal | indica o
operador lógico ou.
<!ELEMENT elemento EMPTY> - define um elemento como vazio.
```

Figura 3.10 – Declaração de elementos num DTD.

A Figura 3.11 demonstra a declaração de entidades de XML.

```
<!ENTITY parametro “(#PCDATA)”> - define parâmetros dentro de
um DTD.
<!ELEMENT nome %parametro; > - usa a definição anterior de
parâmetro.
<!ENTITY nome_autor “Helder”> - define uma entidade.
<nome>&nome_autor</nome>- aplicação da entidade numa marcação.
<!ENTITY nome SYSTEM “endereço”> - declaração de uma entidade
externa.
<!ENTITY vogal SYSTEM “vogal.xml”> - aplicação da declaração.
```

Figura 3.11 – Declaração de entidades num DTD.

A declaração de atributos é demonstrada na Figura 3.12.

```
<!ATTLIST nome_elemento nome_atributo tipo valor_predefinido>
```

Figura 3.12 – Declaração de atributos num DTD.

Os atributos podem ser dos seguintes tipos:

- *CDATA*: designa um atributo constituído por texto, normalmente colocado entre aspas, e excluindo os caracteres que possam ser confundidos com os caracteres de controlo do XML.
- *ENTITY* e *ENTITIES*: este tipo de atributo consiste numa forma de ligação do documento XML a um ficheiro que é exterior ao documento e que seja constituído por dados que não são XML.
- *ENUMERATED*: conjunto de atributos, no qual se pode seleccionar os atributos a usar.
- *ID*: usado no XML para identificar um elemento de forma unívoca e clara.
- *IDREF* e *IDREFS*: atributos cujo valor é o ID de outro elemento no documento.
- *NMTOKEN* e *NMTOKENS*: restringe o conteúdo de um atributo a um nome que seja válido em XML.
- *NOTATION*: referência a uma entidade externa que não seja um documento XML.

O *valor_predefinido* de um atributo é o valor que este assume se não for inicializado explicitamente ao ser utilizado no documento XML. Este campo pode ser substituído por uma palavra reservada que tome um determinado valor, consoante as situações. Por exemplo:

- *#REQUIRED*: quando o valor do atributo for obrigatório.
- *#IMPLIED*: quando o valor do atributo for facultativo.
- *#FIXED*: o valor do atributo não pode ser mudado.

A Figura 3.13 apresenta um exemplo de um DTD e respectivo documento XML.

```
Declaração do DTD: doc.dtd
<!ELEMENT documento (cabecalho, corpo)>
<!ELEMENT cabecalho (titulo, assunto)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT assunto (#PCDATA)>
<!ELEMENT corpo (titulo_corpo, texto, assinatura)>
<!ELEMENT titulo_corpo (#PCDATA)>
<!ATTLIST titulo_corpo estilo (Bold|Italico) #REQUIRED>
<!ELEMENT texto (#PCDATA)>
<!ELEMENT assinatura (#PCDATA)>

Documento XML: doc.xml
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<!DOCTYPE documento SYSTEM “doc.dtd”>
<documento>
  <cabecalho>
     <titulo>Memorando</titulo>
     <assunto>Mudança do número de telefone</assunto>
  </cabecalho>
  <corpo>
     <titulo_corpo estilo=”Bold”>Informação</titulo_corpo>
     <texto>O novo número é agora o 1765.</texto>
     <assinatura>O Director Geral.</assinatura>
  </corpo>
</documento>
```

Figura 3.13 – Exemplo de DTD e uso num documento XML.

3.3 XML Schema Definition - XSD

Para além de um DTD, existe outra forma de especificar as regras de uma linguagem do tipo XML, através de um XML *Schema* que surgiu pela primeira vez em 2001. Um XML *Schema*, ou XML *Schema Definition* (XSD) pode ser visto como um sucessor de um DTD, uma vez que possui as seguintes vantagens em relação aos anteriores:

- Suporte a tipos de dados comuns;
- Sistema de classes que favorece a reutilização de estruturas dentro do *Schema*;
- Suporte nativo de XML *Namespaces* (73);
- Facilidade para tratar elementos de conteúdo misto;
- Melhor especificação de restrições aos conteúdos do documento XML.

A Figura 3.14 exemplifica a estrutura XML com a qual é definido um XSD.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://lpf-esi.fe.up.pt"
xmlns="http://lpf-esi.fe.up.pt"
elementFormDefault="qualified">
...
</xs:schema>
```

Figura 3.14 – Estrutura XML de um Schema.

A Figura 3.15 exemplifica a definição de um elemento num XSD.

```
<element name=”nome” type=”string”/> - define um elemento. Se não for incluído o atributo type, significa que o elemento é vazio.
```

Figura 3.15 – Definição de um elemento no XSD.

Um elemento pode ser de vários tipos. Os mais comuns são: `xs:string`, `xs:decimal`, `xs:integer`, `xs:boolean`, `xs:date` e `xs:time`.

Existem dois tipos de elementos num XSD:

- Simples – `simpleType` - só podem conter texto;
- Complexos – `complexType` - podem conter outros elementos ou atributos.

```
<xs:element name=”numero”>
  <xs:simpleType>
    <xs:restriction base=”xs:integer”>
      <xs:maxInclusive value=”100”/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name=”capa_da_tese”>
  <xs:complexType>
    <xs:sequence>
      <xs:element name=”titulo”/>
      <xs:element name=”autor”/>
      <xs:element name=”orientador”/>
      <xs:element name=”data”/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figura 3.16 – Exemplos de elementos de tipos Simples e Complexos no XSD.

A Figura 3.17 exemplifica a definição de um atributo num XSD, onde:

- `name` – indica o nome do atributo;
- `type` – indica tipos de dados já referidos no exemplo do elemento;
- `use` – pode ser `optional` (opcional) ou `required` (obrigatório);
- `default` – valor por defeito do atributo.

```
<attribute name=”” type=”” use=”” default=”” />
```

Figura 3.17 – Definição de um elemento no XSD.

A Figura 3.18 exemplifica o uso de um XML Schema num documento XML.

XML Schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
<xs:element name="note">
    <xs:complexType>
      <xs:sequence>
     <xs:element name="to" type="xs:string"/>
     <xs:element name="from" type="xs:string"/>
     <xs:element name="heading" type="xs:string"/>
     <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Documento XML:

```
<?xml version="1.0"?>
<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Figura 3.18 – Exemplo de um XML Schema e respectivo documento XML.

3.4 Análise, Interpretação e Conversão de XML - Parsers

O processamento de documentos XML é feito por analisadores de XML (*parsers*). Existem dois tipos de analisadores:

- Analisadores baseados em árvores – DOM[12] (74)
- Analisadores baseados em eventos – SAX[13] (75)

Os analisadores do tipo DOM mapeiam os documentos numa árvore XML, onde é possível navegar através de funções pré-implementadas. Estes analisadores são bastante versáteis e fornecem uma visão de alto nível do documento. No entanto, uma vez que a árvore é construída com recurso a estruturas de dados, esta abordagem pode consumir bastantes recursos em memória se o documento for de grande dimensão.

Os analisadores do tipo SAX reportam eventos relacionados com o inicio e fim dos elementos XML, sendo processados sempre que um evento é chamado. Estes analisadores são bastante simples de se usar e possuem uma visão de baixo nível do documento. São especialmente úteis nos casos em que os documentos a processar requerem uma estrutura de dados própria, que não a de uma árvore XML.

Existem diversos analisadores de documentos XML que podem ser adaptados pelos programadores. Por exemplo:

- *Xerces*[14] - Suporta DOM e SAX. Possui uma versão JAVA e outra em C++.
- *XP*[15] - Suporta apenas SAX. Funciona em JAVA.
- *Microsoft XML Parser*[16] - Suporta apenas DOM. Funciona em C++.
- *XML::Parser*[17] –Suporta apenas SAX. Funciona em PERL.

A maioria dos analisadores XML deriva do analisador *Expat*, desenvolvido por Clark Cooper em 1999. Para as aplicações desenvolvidas no âmbito desta dissertação, foi utilizado o analisador *XML::Parser*.

[12] DOM – *Document Object Model*
[13] SAX – *Simple API for XML*
[14] Java - http://xml.apache.org/xerces-j/index.html; C++ - http://xml.apache.org/xerces-c/index.html
[15] http://jclark.com/xml/xp/index.html
[16] http://msdn.microsoft.com/xml/default.asp
[17] http://search.cpan.org/dist/XML-Parser/

Página em branco.

4 Introdução à Linguagem de Marcação Matemática - MathML

MathML, linguagem de marcação matemática (*Mathematical Markup Language)*[18], é uma aplicação do XML, desenvolvida pelo W3C para a codificação de expressões matemáticas na *web*. A primeira versão desta linguagem, *MathML 1.0*, foi publicada como recomendação do W3C em Abril de 1998. A versão adoptada durante a realização desta dissertação, *MathML 2.0*[19], foi publicada em 21 de Outubro de 2001.

Esta linguagem fornece uma sintaxe simples e concisa para codificar a estrutura visual e o significado semântico de uma expressão matemática. A utilização de MathML permite desenvolver páginas *web* com matemática, e adicionar interacção às suas expressões, tais como: aumento visual de uma expressão (*zoom*), navegação entre sub expressões, descrições na barra de estados de um visualizador de páginas *web* (*browser*), e inserção de comentários numa fórmula. Também, e uma vez que os conteúdos são marcados e identificados por esta metalinguagem, a criação de uma base de dados de documentos científicos com MathML tornaria os seus conteúdos facilmente pesquisáveis, indexáveis ou arquiváveis.(23)

Ao nível de acessibilidade na *web* e acesso à informação, o MathML desempenha um papel importante, uma vez que o desenvolvimento de programas de conversão texto-fala com suporte para MathML permitirão uma interpretação auditiva das expressões matemáticas (76), sendo este um dos principais objectivos do presente trabalho.

O facto de o MathML ser uma aplicação XML facilita o processamento por parte de qualquer editor ou analisador (*parser*) de XML; como se trata apenas de texto puro, duas vantagens são-lhe conferidas: a portabilidade e a independência de plataformas. O uso de MathML aplica-se a cinco categorias principais:

- Visualização de conteúdos matemáticos em páginas web.
- Criação de conteúdos dinâmicos e interactivos on-line.
- Publicação de informação técnica em formato electrónico.
- Troca de dados matemáticos entre aplicações.
- Interpretação de expressões matemáticas em meios não visuais.

[18] Página oficial do MathML em: http://www.w3.org/Math/ .
[19] MathML 2.0 2nd Edition, W3C Recommendation: http://www.w3.org/TR/MathML2/ .

Fórmula:	$(a+b)^2$

MathML Presentation Markup:	`<math>`
	`  <mrow>`
	`  <msup>`
	`  <mfenced>`
	`  <mrow>`
	`  <mi>a</mi><mo>+</mo><mi>b</mi>`
	`  </mrow>`
	`  </mfenced>`
	`  <mn>2</mn>`
	`  </msup>`
	`  </mrow>`
	`</math>`

MathML Content Markup:	`<math>`
	`  <mrow>`
	`  <apply>`
	`  <power/>`
	`  <apply>`
	`  <plus/>`
	`  <ci>a</ci><ci>b</ci>`
	`  </apply>`
	`  <cn>2</cn>`
	`  </apply>`
	`  </mrow>`
	`</math>`

Figura 4.1 - Exemplo de uma expressão matemática em MathML.

Neste capítulo pretende-se apresentar sumariamente alguns conceitos básicos sobre o MathML, descrever os diferentes conjuntos de etiquetas de marcação, apresentar questões relacionadas com *Unicode* e caracteres especiais nos documentos MathML, comentar sobre os métodos de publicação de MathML, e por fim abordar o uso desta tecnologia para a interpretação áudio de fórmulas matemáticas (área onde o trabalho desta dissertação foi desenvolvido).

Actualmente existem diversas ferramentas que lidam com o MathML. Uma listagem com as ferramentas mais conhecidas pode ser encontrada na página de Internet do MathML - http://www.w3.org/Math/Software/mathml_software_by_name.html.

4.1 Etiquetas de Marcação de MathML

Uma vez que o MathML é uma aplicação XML, as regras apresentadas no capítulo anterior dedicado ao XML aplicam-se ao vocabulário do MathML. No entanto foram introduzidas regras adicionais que encaixam nas seguintes categorias:

- Restrições ao tipo de valor que um atributo de MathML pode conter.
- Restrições quanto ao número de filhos de um elemento, e associação de diferentes significados consoante a ordem pela qual esses filhos são apresentados. Geralmente, nestes casos, os filhos são denominados de argumentos.

Embora um analisador (*parser*) de XML não reconheça as violações a estes dois pontos, o não cumprimento destas regras deverá ser reconhecido como erro por um analisador de MathML. O MathML, tal como o XML, também é sensível ao uso de maiúsculas e minúsculas (*case-sensitive*), no entanto todos os elementos e atributos foram declarados no seu DTD (77) como sendo usados em minúsculas.

O elemento raiz do MathML é `<math></math>`. Todas as expressões matemáticas isoladas deverão estar incluídas dentro deste elemento raiz.

Existem dois conjuntos de etiquetas de marcação no MathML: *Presentation Markup* e *Content Markup*. Mais adiante, neste capítulo, serão aprofundados os seus conceitos e propriedades.

O MathML possui características importantes em relação aos espaços em branco nos conteúdos das suas marcações e nas suas próprias marcas:

- Todos os espaços em branco, espaçamentos e símbolos de espaçamento ou mudança de linha (*tab*, *new line* e *carriage return*) são ignorados.
- Os espaços em branco dentro das etiquetas de marcação são eliminados no principio e no fim da etiqueta.
- Quando existe uma sequência de espaços brancos dentro de uma etiqueta, estes são substituídos por um só espaço. Quando é propositado o uso de uma sequência de espaços brancos, deverão ser usados códigos de formatação próprios, como por exemplo: ` ` ou ` `.

Todos os elementos de MathML têm de aceitar, pelo menos, seis atributos:

- `class`, `style` e `id`: facilitam o uso de folhas de estilo como CSS e XSL.
- `xlink:href`: usado para a inserção de hiper-ligações.
- `xref`: usado para referenciar documentos marcados paralelamente.
- `other`: usado para especificar atributos não normalizados mas usados por outras aplicações.

Os caracteres de MathML são definidos no MathML DTD (77), que inclui mais de 2500 declarações de caracteres especiais. O uso de *Unicode* é também muito importante para o MathML e para a escrita de expressões matemáticas. Este tema será abordado mais adiante neste capítulo.

4.1.1 Presentation Markup vs. Content Markup

A representação e expressão de fórmulas e outros conteúdos de carácter matemático, são perceptíveis ao ser humano através de dois conceitos distintos: a apresentação ou notação da expressão matemática, e o conceito ou ideia que a mesma representa.

No entanto, estes dois conceitos podem ser perceptíveis de modos diferentes, isto é, o mesmo conceito pode possuir diferentes notações (Tabela 4.1) e a mesma notação pode transmitir diferentes conceitos matemáticos (Tabela 4.2).

Tabela 4.1 - Matemática: mesmo conceito, diferentes notações.

Conceito	**Notação**
Derivada de f em ordem a x.	$\frac{df}{dx}$ ou $f'(x)$ ou $\frac{\Delta y}{\Delta x}$ ou $y = f(x)$
Divisão de a por b.	a/b ou $\frac{a}{b}$ ou ab^{-1}

Tabela 4.2 - Matemática: mesma notação, diferentes conceitos.

Conceito	**Notação**
Produto entre a variável H e a variável e. Símbolo químico do Hélio.	He
Um dos componentes do vector X. Derivada da variável X.	X'

Sendo assim, tornou-se claro para a W3C que a especificação do MathML deveria suportar dois conjuntos distintos de etiquetas de marcação que permitissem transmitir de forma não ambígua a representação e aparência de uma expressão matemática, do conceito e significado inerente à mesma. Surgiram então dois subconjuntos de marcação:

- *Presentation Markup* – este subconjunto de etiquetas de marcação é focado na representação visual com que uma expressão matemática é representada. Por exemplo, é usado na publicação de conteúdos matemáticos num documento on--line.
- *Content Markup* – este subconjunto de etiquetas de marcação é focado no conceito ou significado que uma expressão matemática transmite. É usado entre aplicações matemáticas como forma de exportação da fórmula entre documentos ou aplicações.

Estes dois subconjuntos de etiquetas de marcação complementam-se, uma vez que a sua combinação fornece tanto o significado da expressão matemática como a sua notação. Ao conjunto de etiquetas combinado dá-se o nome de *Parallel Markup*.

De seguida exemplifica-se a aplicação de cada subconjunto de marcação em MathML para a seguinte expressão matemática: $f(x+y)$.

Utilização do *Presentation Markup*:

O objectivo seria apenas visualizar correctamente a expressão matemática num documento on-line. A interpretação oral da fórmula poderia ser:

- "*éfe multiplica a expressão chis mais ipsilone*", ou;
- "*função éfe de chis mais ipsilone*".

Esta última versão poderia ser implementada pelo seguinte código em *MathML Presentation Markup*:

```
<math>
  <mrow>
    <mi>f</mi>
    <mo>&ApplyFunction</mo>
    <mo>(</mo>
    <mrow>
      <mi>x</mi><mo>+</mo><mi>y</mi>
    </mrow>
    <mo>)</mo>
  </mrow>
</math>
```

Figura 4.2 - Exemplo em MathML Presentation Markup.

De notar o uso do operador `&ApplyFunction;` sem o qual, um analisador (*parser*) não seria capaz de detectar que a notação acima deveria ser dita: "*função éfe de...*". Sendo assim, o *Presentation Markup* não parece ser muito adequado para uma conversão áudio de uma expressão matemática, uma vez que este tipo de marcação prende-se mais com pormenores de visualização e não com o conceito e estrutura da expressão. Para um cego não importa a aparência ou notação, mas sim o significado da fórmula. O mesmo já não se pode dizer de um amblíope, onde apesar de a visão ser reduzida ou diminuída, existe a possibilidade de observar a expressão matemática, sendo a interpretação oral apenas um auxilio à interpretação semântica.

Utilização do *Content Markup*:

O objectivo seria caracterizar a estrutura da expressão matemática, independentemente da notação utilizada. Agora, a interpretação oral da expressão seria sempre dita como: "*função éfe de chis mais ipsilone*".

Isto poderia ser implementado pelo seguinte código em *MathML Content Markup*:

```
<math>
  <apply>
    <ci>f</ci>
    <apply>
      <plus/>
      <ci>x</ci><ci>y</ci>
    </apply>
  </apply>
</math>
```

Figura 4.3 - Exemplo em MathML Content Markup.

Com este subconjunto de etiquetas de marcação é possível saber a operação antes dos seus argumentos, o que facilita a compreensão da estrutura da expressão matemática, assim como a construção da sua forma escrita completa por extenso e respectiva leitura. O mesmo não acontecia com o *Presentation Markup* onde o analisador (*parser*) tinha de navegar a fórmula em toda a sua extensão e tirar conclusões de possíveis relações entre parâmetros e operadores. Sendo assim, o *Content Markup* parece ser mais flexível e apropriado para se usar em aplicações cujo objectivo sejam a leitura de expressões matemáticas. (76)

Utilização do *Parallel Markup*:

Esta é a forma ideal de se representarem expressões matemáticas em documentos on--line, uma vez que este conjunto de etiquetas de marcação reúne as propriedades dos subconjuntos anteriormente apresentados. Isto é, existe uma não-ambiguidade tanto em notação como em significado.

Deste modo seria possível recolher informação de aparência e de estrutura permitindo uma visualização optimizada combinada com a possibilidade de leitura navegando na expressão. A fórmula seria representada como: $f(x+y)$ e seria lida como: "*função éfe de chis mais ipsilone*".

Isto poderia ser implementado pelo seguinte código em *MathML Parallel Markup*:

```
<math>
  <semantics>
   <mrow>
     <mi>f</mi>
     <mo>&ApplyFunction;</mo>
       <mo>(</mo>
         <mrow>
            <mi>x</mi><mo>+</mo><mi>y</mi>
         </mrow>
       <mo>)</mo>
   </mrow>
   <annotation-xml encoding=”MathML-Content”>
    <apply>
      <ci>f</ci>
      <apply>
        <plus/>
        <ci>x</ci><ci>y</ci>
      </apply>
    </apply>
   </annotation-xml>
 </semantics>
</math>
```

Figura 4.4 - Exemplo em MathML Parallel Markup.

Esta forma de marcação é a mais completa possível pois co-relaciona o significado da expressão matemática com a sua notação.

Compreendidas as aplicações de cada um dos conjuntos de etiquetas de marcação do MathML, resta explorar as suas propriedades; assim como vantagens e desvantagens, nas suas aplicações práticas para a interpretação, navegação e leitura de expressões matemáticas em documentos técnicos.

Compilação das propriedades do *Presentation Markup*:

- Vocacionado para a apresentação visual de uma expressão matemática.
- É suportado nativamente pelos sistemas de navegação de páginas *web* (*browsers*) Mozilla[20], Amaya[21] e Netscape[22].
- Conjunto de etiquetas de marcação é composto por 30 elementos e 50 atributos.
- É ambíguo em termos de significado da expressão matemática.
- Não é vocacionado para a conversão áudio de uma expressão matemática, embora seja possível a sua adaptação.
- A transformação de *Presentation Markup* em *Content Markup* não é aconselhada. No entanto, um grupo da organização *OpenMath*[23] tem procurado criar uma folha de transformação XML (XSLT) para esse efeito, embora ainda sem sucesso.

Compilação das propriedades do *Content Markup*:

- Vocacionado para a estrutura e significado de uma expressão matemática, independentemente da notação utilizada.
- É suportado pelos sistemas de navegação de páginas *web* (*browsers*), via uso de programas externos (*plug-ins*), Netscape e Internet Explorer[24].
- Usado para transferir MathML entre aplicações.
- Possui cerca de 150 elementos e 12 atributos.
- É ambíguo em termos de notação matemática.
- Vocacionado para se realizarem conversões áudio de uma expressão matemática.
- Este tipo de marcação é sempre convertido em *Presentation Markup* por uma folha de transformação XML (XSLT) antes de ser visualizado no browser. No entanto, o código fonte do documento apresenta-se como *Content Markup*.
- É limitado nas etiquetas que apresenta. Devido à enorme quantidade de operadores que existem actualmente, a equipa de MathML da W3C decidiu escolher um conjunto limitado que permitisse definir a maior parte das expressões matemáticas nas seguintes áreas: aritmética, álgebra, lógica e

[20] Mozilla Firefox – http://www.mozilla.org/
[21] Amaya, W3C browser - http://www.w3.org/Amaya/
[22] Netscape Browser - http://browser.netscape.com/
[23] OpenMath Standard - http://www.oasis-open.org/cover/openMath.html
[24] Internet Explorer Browser - http://www.microsoft.com/windows/ie/downloads/default.mspx

relações, cálculo e cálculo vectorial, teoria de conjuntos, sequências e séries, funções elementares clássicas, estatística e álgebra linear.

Falta apenas referir um aspecto relevante: a maioria dos programas de edição e criação de documentos técnicos com MathML utiliza *Presentation Markup* para a publicação dos seus documentos, embora alguns possam utilizar *Content Markup* na troca de dados entre aplicações. Isto deve-se ao facto de serem editores do tipo WYSIWYG[25] onde é valorizada a apresentação visual da expressão matemática, com prioridade relativamente ao conceito ou significado da mesma.

4.1.2 Descrição sumária das etiquetas de marcação de *MathML Presentation Markup*

O conjunto de etiquetas de marcação de *Presentation Markup* possui cerca de 30 elementos e 50 atributos que descrevem a notação e a representação visual de uma expressão matemática.

A maioria dos elementos de *Presentation Markup* podem ser divididos em duas categorias: *token elements* e *layout schemata*. Existem ainda outros elementos que não pertencem a nenhuma das duas categorias: `none` e `prescripts` (que são elementos vazios); e `maction` (que é usado para adicionar interactividade às equações matemáticas).

Token elements são marcas que representam os mais pequenos blocos de construção de notação matemática, isto é, números, operadores e identificadores que representam nomes de variáveis ou funções. Também podem ser usados para representar texto, espaços brancos ou glifos.

Este tipo de elementos são os únicos neste conjunto de etiquetas de marcação, que podem conter directamente dados, sejam caracteres de *Unicode* ou referências para entidades MathML.

[25] *What You See Is What You Get* ("o que vê é o que vai obter")

Tabela 4.3 - Tipos de *Token Elements* (*Presentation Markup*).

Elemento	Papel desempenhado
`mn`	número
`mo`	operador, separador ou barreira
`mi`	identificador
`mtext`	texto
`mspace`	espaço em branco
`ms`	*string*
`mglyph`	glifo

Layout schemata são marcas que especificam modelos para construir expressões a partir de pequenos blocos de construção de notação matemática. Este tipo de elementos só podem conter outros elementos. As regras de interpretação destes elementos são mais complexas do que os *token elements* uma vez que especificam formas de construir uma estrutura bidimensional a partir de expressões mais pequenas.

Tabela 4.4 - Tipos de *Layout Schemata* (*Presentation Markup*).

Elemento	Papel desempenhado
`mrow`	agrupa expressões pequenas na linha horizontal
`mfrac`	fracção
`msqrt`	raiz quadrada
`mroot`	raiz
`mstyle`	aplica estilos
`mphantom`	torna o conteúdo invisível
`mfenced`	agrupa os conteúdos com alguma forma de barreira, ex.: parêntesis
`mpadded`	ajusta espaços em volta dos conteúdos
`menclose`	aplica símbolo de agrupamento de conteúdos
`merror`	mensagem de erro
`msub`	adiciona um *subscript* à base
`msup`	adiciona um *superscript* à base
`msubsup`	adiciona um par *subscript-superscript* à base
`munder`	adiciona um *underscript* à base

`mover`	adiciona um *overscript* à base.
`munderover`	adiciona um par *underscript-overscript* à base
`mmultiscripts`	adiciona múltiplos *prescripts* e *postscripts* à base
`mtable`	tabela ou matriz
`mtr`	linha de uma tabela
`mlabeledtr`	linha de uma tabela com nome
`mtd`	célula de uma tabela
`maligngroup`	grupo de alinhamento
`malignmark`	marca de alinhamento

Anteriormente foi referido que o MathML impunha, como regras extras às do XML, o número de filhos de um elemento, denominados *argumentos*. A Tabela 4.5 resume o número de argumentos de determinados elementos do *Presentation Markup*:

Tabela 4.5 - Argumentos para cada elemento de *Presentation Markup*.

Elemento	**Nº de argumentos**	**Papel desempenhado pelo argumento**
`mrow`	0 ou mais	
`mfrac`	2	numerador denominador
`msqrt`	1*	
`mroot`	2	base raiz
`mstyle`	1*	
`merror`	1*	
`mpadded`	1*	
`mphantom`	1*	
`mfenced`	0 ou mais	
`menclose`	1*	
`msub`	2	base *subscript*
`msup`	2	base *superscript*
`msubsup`	3	base *subscript superscript*
`munder`	2	base *underscript*
`mover`	2	base *overscript*
`munderover`	3	base *underscript overscript*

`mmultiscripts`	1 ou mais	base (*underscript overscript*)* [<mprescripts/>(*underscript overscript*)*]
`mtable`	0 ou mais	elementos `mtr` ou elementos `mlabeledtr`
`mtr`	0 ou mais	elementos `mtd`
`mlabeledtr`	2	etiquetas ou elementos `mtd`
`mtd`	1*	
`maction`	1 ou mais	depende do atributo `actiontype`

O sinal 1* possui uma propriedade especial: significa que o elemento em questão actua como se existisse uma marca `mrow` a envolver uma expressão mais pequena. Por exemplo:

```
<math><msqrt><mo>-</mo><mn>1</mn></msqrt></math>
é equivalente a:
<math><msqrt><mrow><mo>-</mo><mn>1</mn></mrow></msqrt></math>
```

Figura 4.5 - Exemplo do uso de <mrow> em *Presentation Markup*.

No Anexo A são apresentados exemplos de *Presentation Markup*.

4.1.3 Descrição sumária das etiquetas de marcação de *MathML Content Markup*

O conjunto de etiquetas de marcação de *Content Markup* possui cerca de 150 elementos e 12 atributos que descrevem o conceito e significado de uma expressão matemática. A maioria dos elementos de *Content Markup* pode ser dividido nas seguintes categorias:

- *Token elements* – são os únicos elementos a conter dados. Todos os outros podem conter apenas outros elementos. Existem três elementos: `cn` (contém números), `ci` (contém identificadores) e `csymbol` (contém símbolos definidos pelo autor).
- *Elementos de construção* – são elementos que constroem expressões matemáticas combinando dados com operadores ou funções. Por exemplo: `apply`.

- *Operadores e Funções* – tal como o nome indica, estes elementos correspondem a operadores e funções. Por exemplo: `int` (integral), `sin` (seno de) e `plus` (adição).
- *Qualifier elements* – são elementos que contribuem para fornecer informação adicional ao significado de outros elementos. Por exemplo: `uplimit`, `lowlimit` e `bvar` são elementos usados para especificar o limite superior, o limite inferior e a variável de integração, de um integral definido pelo elemento `int`.
- *Constantes e Símbolos* – tal como o nome indica, estes elementos correspondem a constantes e símbolos. Por exemplo: `pi` (número pi), `exponentiale` (número de neper) e `infinity` (símbolo do infinito).
- *Elementos de mapeamento semântico* – são elementos utilizados para fornecer informação adicional sob a forma de anotações ou comentários. Exemplo: `annotation`.

Tabela 4.6 - Elementos do conjunto de etiquetas de marcação *Content Markup*.

Categoria	**Elementos**
Token elements	`ci, cn, csymbol`
Elementos de construção básicos	`apply, lambda, declare, reln, fn, set, list, vector, matrix, matrixrow, interval, piecewise, piece, otherwise, domaIn codomaIn image, inverse, ident`
Aritmética, Álgebra e Lógica	`plus, minus, times, divide, power, root, quotient, rem, exp, factorial, Max, mIn gcd, abs, conjugate, arg, real, imaginary, lcm, floor, ceiling, and, or, xor, not, implies, forall, exists`
Relações	`eq, neq, gt, lt, geq, leq, equivalent, approx, factorof, tendsto, In notIn subset, prsubset, notsubset, notprsubset`
Funções elementares	`exp, ln, log, sin cos, tan, sec, cosec, cot, sinh, cosh, tanh,`

	sech, cosech, coth, arcsin arccos, arctan, arcsec, arccosec, arccot, arcsinh, arccosh, arctanh, arcsech, arccosech, arccoth
Teoria de Conjuntos	set, list, union, intersect, setdiff, card, cartesianproduct
Sequências e Séries	sum, product, limit
Álgebra Linear	determinant, transpose, selector, vectorproduct, scalarproduct, outerproduct
Calculus	int, diff, partialdiff, grad, divergence, curl, laplacian
Estatística	mean, median, mode, var, sdev
Qualifier elements	bvar, lowlimit, uplimit, degree, logbase, domainofapplication, momentabout
Constantes e Símbolos	pi, exponentiale, eulergamma, infinity, imaginaryi, true, false, emptyset, notanumber, integers, reals, rationals, complexes, primes, naturalnumbers
Elementos de mapeamento semântico	semantics, annotation, annotation-xml

Como se pode observar pela Tabela 4.6, o conjunto de etiquetas de marcação de *Content Markup* cobre poucas áreas da matemática (ensino básico, secundário e alguns casos do ensino superior).

Cada um dos elementos de marcação de *Content Markup* possui um significado semântico, no entanto existem dois atributos importantes que permitem mudar o conceito de um elemento:

- `definitionURL` – indica um endereço web onde se encontra a informação acerca do significado do elemento;
- `encoding` – especifica em que formato se encontra a informação.

Por exemplo, o elemento `times` representa, por defeito, a operação multiplicação entre dois números. No entanto, é possível modificar a semântica deste elemento para que ele represente o produto interno de dois vectores:

$A \times B$

```
<apply>
  <times encoding="text"
definitionURL="http://www.exemplo.com/produtointerno.html"/>
  <ci type="vector">A</ci>
  <ci type="vector">B</ci>
</apply>
```

Figura 4.6 - Exemplo do uso de definitionURL em *Content Markup*.

Contudo, o uso do atributo `definitionURL` é problemático, uma vez que não é conhecido o que se encontra no endereço especificado, nem existe uma normalização sobre a forma como se deve apresentar essa informação adicional.

No Anexo B são apresentados exemplos de *Content Markup*.

4.2 Unicode & MathML

Todos os computadores lidam com números, e gravam letras e outros caracteres na memória designando um número para cada um deles.

Antes de o Unicode ser inventado, havia centenas de sistemas diferentes de codificação. No entanto, nenhum desses sistemas de codificação poderia conter caracteres suficientes. Por exemplo, a União Europeia por si só requer vários sistemas de codificação diferentes para cobrir todas as línguas. Mesmo para uma única língua como o inglês não havia sistema de codificação adequado para todas as letras, pontuação e símbolos técnicos em uso corrente.

No entanto, estes sistemas também podem entrar em conflito entre si, uma vez que dois codificadores podem usar o mesmo número para dois caracteres diferentes, ou usar números diferentes para o mesmo caracter. Além disso, cada computador precisa

de suportar diversos sistemas de codificação diferentes, o que pode fazer com que alguns dados partilhados entre eles, possam ser corrompidos.

O Unicode veio mudar tudo isso fornecendo um número único para cada caracter, independentemente da plataforma, programa ou língua. Isto possibilita que um único programa ou sítio web seja utilizado em qualquer computador sem necessidade de reengenharia, permitindo que os dados sejam partilhados sem serem corrompidos. (78)

Sendo assim, fazia todo o sentido para a W3C usar a codificação Unicode para a notação de funções e símbolos matemáticos, assim como para todos os caracteres de MathML.

4.2.1 Unicode

Apesar de ser modelado a partir do conjunto de caracteres ASCII[26], o Unicode vai mais longe permitindo a codificação de todos os caracteres usados em todas as línguas do mundo. Tem capacidade de codificação para mais de 1 milhão de caracteres. O Unicode trata-os de modo idêntico, o que permite uma enorme flexibilidade na mistura de todo o tipo de caracteres diferentes, sejam eles alfabéticos, símbolos ou ideográficos.

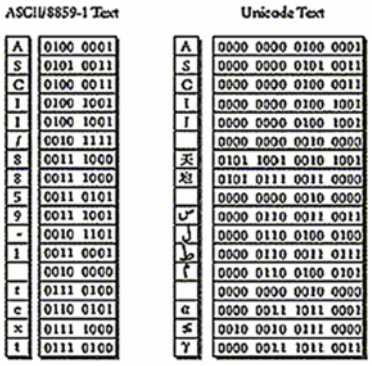


Figura 4.7 - Comparação entre tabela de ASCII e tabela de Unicode. (71)

O *standard* de Unicode (71) especifica um valor numérico e um nome para cada um dos caracteres. Juntamente com o código e nome é fornecida informação adicional, como por exemplo, se é maiúsculo ou minúsculo, a direccionalidade do caracter e outras propriedades alfabéticas.

[26] ASCII - *American Standard Code for Information Interchange* – http://www.asciitable.com/

Existem três formas principais de codificação Unicode (*UTF – Unicode Transformation Format*):

- UTF-32 – codificação de 32 bits.
- UTF-16 – codificação de 16 bits usada por defeito
- UTF-8 – codificação de 8 bits usada para permitir compatibilidade com o ASCII.

Para além de permitir compatibilidade com os sistemas baseados em ASCII, o *Unicode Standard 4.0* (71) é equivalente ao ISO/IEC 10646, bastando por isso estar em conformidade com o standard Unicode, para estar também em conformidade com a norma ISO. Actualmente o standard de Unicode 4.0 suporta 96.382 caracteres de todo o mundo, e possui ainda reservadas 131.068 posições para novos caracteres.

O Unicode foi desenhado e projectado para ser:

- Universal – possui um repositório de caracteres de todo o mundo, em todas as línguas e representações possíveis.
- Eficiente – actualmente é simples processar um texto. A sincronização entre caracteres deve ser rápida e não ambígua.
- Uniforme – um carácter deve ter sempre o mesmo código para que seja facilmente identificado, visualizado e editado.
- Não ambíguo - um determinado código de 16-bits representa sempre o mesmo símbolo.

Exemplo de que como o Unicode é aplicado no processamento de texto:

Quando um utilizador usa um processador de texto e pressiona a letra ‘T’, o computador recebe a informação de que uma tecla correspondente ao símbolo ‘T’ foi pressionada, e codifica o sinal com o código U+0054 (Unicode). De seguida, o processador de texto guarda esse código em memória e passa uma referência para a aplicação que se encarrega da visualização da letra. Essa aplicação usa o número como um índice para encontrar uma imagem de ‘T’, que posteriormente desenha no visor do monitor. Este processo é repetido à medida que o utilizador vai pressionando teclas.

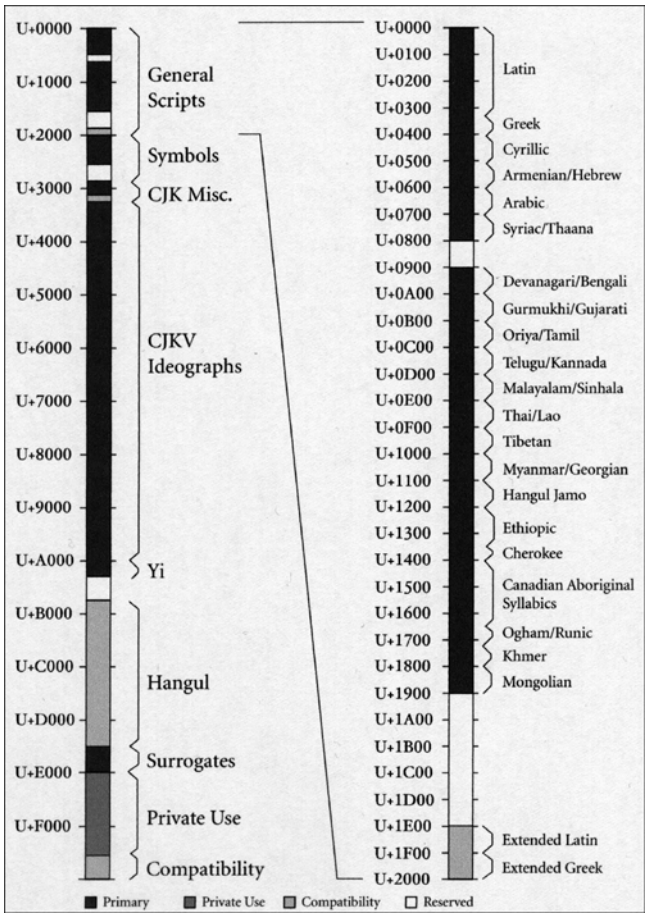


Figura 4.8 – Sistema de alocação de códigos Unicode. (71)

4.2.2 Unicode no MathML

As notações e símbolos matemáticos são muito importantes em todas as ciências, e têm evoluído bastante ao longo da existência do ser humano; actualmente existem enormes colecções de notações e símbolos.

No entanto, se estes símbolos ou glifos não estiverem disponíveis para codificação ou uso num documento, torna-se difícil a produção de conteúdos matemáticos portáveis e facilmente distribuídos, ou apresentados visualmente.

Sendo assim, o MathML faz uso do *standard* Unicode para representar a maior parte dos seus caracteres matemáticos. Todos os caracteres de MathML podem ser definidos como caracteres Unicode legais em documentos XML ou como elementos `mglyph`. Este último tipo só é usado para representar caracteres que ainda não tenham sido aprovados pelo *standard* Unicode, e que como tal ainda não possuem uma codificação Unicode. O uso deste tipo é raro, uma vez que a maior parte dos símbolos e glifos são suportados.

Os caracteres que o MathML considera correctos e válidos são os seguintes (79) (em hexadecimal e Unicode):

- *09* (tab = U+0009);
- *0A* (line feed = U+000A);
- *0D* (carriage return = U+000D);
- 00*20-D7FF* (U+0020..U+D7FF);
- *E000-FFFD* (U+E000..U+FFFD);
- *10000-10FFFF* (U+010000..U+10FFFF).

O MathML permite três formas de codificar os seus caracteres:

- *Usando os caracteres directamente*: Por exemplo, a letra "A" pode ser simplesmente codificada pressionando a tecla da letra (símbolo U+0065). Infelizmente, este método não resulta para a maioria dos símbolos de conotação matemática.
- *Usando referências numéricas de XML*: A letra "A" seria representada como A (decimal) ou A (hexadecimal). Repare-se que os números referem-se sempre à codificação Unicode.
- *Usando referências de entidades*: O MathML DTD define entidades que expandem a codificação dos caracteres, permitindo, por exemplo, usar a referência é, em vez da referência &#xE9 que codifica o caracter "é".

Quando a especificação do MathML 2.0 foi aprovada, a versão de Unicode existente era a 3.0. Nessa altura, diversos símbolos do MathML não eram suportados, e tiveram de ser acrescentados ao Unicode 3.1 e Unicode 3.2. A versão actual do MathML utiliza a especificação Unicode 3.2. Tendo isto em conta, verifica-se a existência de dois planos de alocação de caracteres Unicode distintos:

- Plano 0 – *Basic Multilingual Plane* (BMP) – é o esquema de alocação utilizado pelo Unicode 3.0, e que contém todos os valores abaixo de 2^{16}.
- Plano 1 – *Secondary Multilingual Plane* (SMP) – é o esquema de alocação de valores usado pelo Unicode 3.1 e Unicode 3.2, e que contém todos os valores acima de 2^{16}.

Os símbolos matemáticos alfanuméricos suportados pelo Unicode 3.1/3.2 e actualmente pelo MathML, são codificados no Plano 1. Sendo assim, o MathML criou um elemento `mathvariant` que permite uma codificação alternativa do caracter sem usar o plano 1 do Unicode. No entanto, esta abordagem que permite uma visualização correcta do caracter não deve ser utilizada como ferramenta de estilo de texto, e é apenas provisória.

Por exemplo: A letra "A" do alfabeto Fraktur é codificada no plano 1 do Unicode como possuindo o valor: `U1D504`, e poderia ser marcada em MathML da seguinte forma: `<mi>&#x1D504</mi>`. No entanto, é possível evitar o uso de um valor Unicode usando o elemento `mathvariant`: `<mi mathvariant="fraktur">A</mi>`.

Existe ainda uma outra forma de mapear os valores do plano 1 no plano 0: usando o *DTD modificado do MathML* e referências a entidades.

Por exemplo, a mesma letra "A" no alfabeto Fraktur pode ser representada por `&Afr;`, que é substituída pelo valor `&x1D504;` no DTD do MathML (que fica no plano 1), e que é substituída pelo valor `&xE504;` no DTD modificado do MathML (que fica no plano 0).

Este *DTD modificado do MathML* foi criado pela W3C como ferramenta intermédia, e é usada pelo actual *DTD do MathML*.

Tabela 4.7 - Exemplo de códigos Unicode para notações e símbolos matemáticos. (79)

Unicode	Referência no DTD	Símbolo	Descrição
U0003D	equals	=	sinal de igualdade
U003A0	Pi	Π	letra grega pi maiúsculo
U02211	Sum	∑	símbolo de somatório
U0222B	Integral	∫	símbolo de integral
U0221A	Sqrt	√	símbolo de raiz quadrada

4.3 Integração do MathML com outras Tecnologias de Internet

Actualmente é possível publicar documentos com MathML nos navegadores de páginas web (*browsers*) mais conhecidos, tais como Mozilla, Netscape, Internet Explorer e Opera, directamente ou usando aplicações externas (*plug-ins*). Para que o MathML seja interpretado e visualizado de modo correcto, existem duas opções:

- O *browser* utilizado é capaz de interpretar MathML nativamente, como é o caso do Mozilla, Netscape 7.0 e Amaya.
- É utilizado um *plug-in* de controlo tipo *ActiveX* (exemplo: MathPlayer) ou um Java *applet* (exemplo: IBM Techexplorer e WebEQ).

No entanto, a forma como um documento com MathML se integra com um documento *web*, depende muito do *browser* que se vai utilizar e do tipo de *plug-in* que este exige.

Tabela 4.8 - Browsers que suportam MathML (76).

Plataforma	Browser
Windows	IE 5.0 com Techexplorer IE 5.5 com MathPlayer ou Techexplorer IE 6.0 com MathPlayer ou Techexplorer Netscape 6.1 ou superior, com Techexplorer Netscape 7.0 directamente ou com Techexplorer Mozilla 1.0 ou superior Amaya Opera 8.0 ou superior através de folha de estilos CSS (pmathmlcss.css)
Macintosh	IE 5.0 ou superior com Techexplorer Netscape 6.1 ou superior com Techexplorer Mozilla 1.1 ou superior
Linux/Unix	Netscape 6.1 ou superior com Techexplorer Netscape 7.0 directamente ou com Techexplorer Mozilla 1.0 ou superior Amaya

Nalguns casos, é mesmo necessário a instalação de fontes próprias para uma melhor visualização das expressões matemáticas. O MathPlayer e o Techexplorer já incluem fontes na instalação. O mesmo não acontece com o Mozilla e o Netscape, onde é necessário instalar fontes especiais[27].

Exemplo de documento HTML com MathML para ser visualizado pelo Internet Explorer com MathPlayer:

```
<html xmlns:m="http://www.w3.org/1998/Math/MathML">
  <head>
    <object  id="MathPlayer"  classid="clsid:32F66A20-7614-11D4-
BD11-00104BD3F987">
    </object>
    <?import namespace="m" implementation="#MathPlayer">
  </head>
  <body>
    <p>Aqui fica uma expressão matemática:
    <m:math>
      <m:mrow>
         <m:mi>a</m:mi><m:mo>+</m:mo><m:mi>b</m:mi>
      </m:mrow>
    </m:math>
    </p>
  </body>
</html>
```

Figura 4.9 - HTML+MathML no Internet Explorer com MathPlayer.

Exemplo de documento HTML com MathML para ser visualizado pelo Netscape com Techexplorer:

```
<html>
  <head></head>
  <body>
    <p>Aqui fica uma expressão matemática:</p>
    <p><embed type="application/x-techexplorer" mmldata="
    <math xmlns='http://www.w3.org/1998/Math/MathML'>
       <mrow><mi>a</mi><mo>+</mo><mi>b</mi></mrow>
    </math>"
pluginspage='http://www.software.ibm.com/techexplorer/'
height=100 width=200 name="equation">
    </embed></p>
  </body>
</html>
```

Figura 4.10 - HTML+MathML no Netscape com Techexplorer.

[27] Fontes usadas em MathML para Mozilla e Netscape - http://www.mozilla.org/projects/mathml/fonts/

Exemplo de documento HTML com MathML para ser visualizado pelo Internet Explorer com Techexplorer:

```
<html>
  <head></head>
  <body>
    <p>Aqui fica uma expressão matemática:</p>
    <p>
    <object   ID=”eqn”   classid=”clsid:5AFAB315-AD87-11D3-98BB-
002035EFB1A4”
    height=100 width=200>
       <param name=”autosize” value=”true”>
       <param name=”datatype” value=”1”>
       <param name=”data” value=
       “<math xmlns=’http://www.w3.org/1998/Math/MathML’>
           <mrow><mi>a</mi><mo>+</mo><mi>b</mi></mrow>
       </math>”>
    </object>
    </p>
  </body>
</html>
```

Figura 4.11 - HTML+MathML no Internet Explorer com Techexplorer.

Exemplo de documento HTML com MathML para ser visualizado pelo Internet Explorer e pelo Netscape com Techexplorer:

```
<html>
  <head></head>
  <body>
    <p>Aqui fica uma expressão matemática:</p>
    <p><object  ID=”eqn” classid=”clsid:5AFAB315-AD87-11D3-98BB-
002035EFB1A4”
    height=100 width=200>
       <param name=”autosize” value=”true”>
       <param name=”datatype” value=”1”>
       <param name=”data” value=
       “<math xmlns=’http://www.w3.org/1998/Math/MathML’>
          <mrow><mi>a</mi><mo>+</mo><mi>b</mi></mrow>
       </math>”>
       <embed type=”application/x-techexplorer” mmldata=”
       <math xmlns=’http://www.w3.org/1998/Math/MathML’>
          <mrow><mi>a</mi><mo>+</mo><mi>b</mi></mrow>
       </math>”
       pluginspage=’http://www.software.ibm.com/techexplorer/’
height=100 width=200
       name=”eqn”></embed></object></p>
  </body>
</html>
```

Figura 4.12 - HTML+MathML no IE e Netscape com Techexplorer.

Exemplo de documento HTML com MathML para ser visualizado com WebEQ:

```
<html>
  <head></head>
  <body>
    <p>Aqui fica uma expressão matemática:</p>
    <p>
    <applet                                codebase=”../../classes”
code=”webeq3.ViewerControl”
      height=100 width=200>
       <param name=”size” value=”12”>
       <param name=”eq” value=
       “<math xmlns=’http://www.w3.org/1998/Math/MathML’>
           <mrow><mi>a</mi><mo>+</mo><mi>b</mi></mrow>
       </math>”>
    </applet>
    </p>
  </body>
</html>
```

Figura 4.13 - HTML+MathML usando WebEQ.

Para resolver o problema de dependência de *browser* e *plug-in* um dos elementos da W3C, *David Carlisle*, inventou uma folha de estilos XSLT denominada *Universal MathML Stylesheet* (80). Esta folha de estilos tem como objectivo transformar o documento original, adicionando as marcas necessárias no documento, para que este seja correctamente visualizado no *browser*. No entanto, esta folha de estilos XSLT só é suportada no Internet Explorer 5.0 ou superior, no Netscape 6.1 ou superior e no Mozilla 1.1 ou superior.

Exemplo de documento HTML com MathML que usa a *Universal MathML Stylesheet*:

```
<?xml version=”1.0”?>
<?xml-stylesheet type=”text/xsl”
href="http://www.w3.org/Math/XSL/mathml.xsl"?>
<html xmlns=”http://www.w3.org/1999/xhtml”>
  <head><title>...</title></head>
  <body>
     <math xmlns=”http://www.w3.org/1998/Math/MathML”>
        <mrow><mi>a</mi><mo>+</mo><mi>b</mi></mrow>
     </math>
  </body>
</html>
```

Figura 4.14 - HTML+MathML usando a Universal MathML Stylesheet.

4.4 Interpretação do MathML para leitura oral de expressões matemáticas

Tanto quanto se conhece, o trabalho desenvolvido em (64), e durante esta dissertação, foi o primeiro a fazer um estudo que visava a determinação de elementos e atributos que deveriam de ser usados na conversão áudio de expressões matemáticas em MathML.

Nem todos os elementos e atributos do MathML necessitam de ser interpretados e processados na conversão áudio. Por exemplo, atributos visuais ou de estilo nem sempre fornecem informações úteis em relação à expressão e raramente contribuem para um maior enriquecimento da conversão áudio. No entanto, existem casos em que sem os atributos visuais não seria possível compreender a semântica da expressão:

$\left(\frac{3}{2}\right)$ Elemento: `mfrac` Atributo: `linethickness` <> 0 É uma fracção.	$\binom{3}{2}$ Elemento: `mfrac` Atributo: `linethickness` = 0 É um número combinatório.

Figura 4.15 - Influência de um atributo visual no significado da fórmula.

Nesta secção serão exploradas as hipóteses de escolha de um conjunto de marcação para conversão áudio; e serão apresentados comentários relativos aos elementos e atributos a serem interpretados.

4.4.1 Escolha do conjunto de etiquetas de marcação para conversão áudio

O primeiro desafio que se coloca na interpretação oral do MathML é a escolha do conjunto de etiquetas de marcação que se pretende converter.

Foi referido na secção 4.1.1 que o MathML se dividia em dois subconjuntos de marcação: *Presentation Markup* e *Content Markup*.

Embora este último pareça mais adequado à conversão áudio de MathML (uma vez que exprime a semântica de uma expressão matemática), possui dois grandes problemas:

- Apenas cobre os níveis básicos da matemática. O dicionário de operadores não é vasto, empobrecendo o seu suporte à matemática.[28]
- A maioria dos documentos on-line publicados estão codificados com *Presentation Markup*.

Assim sendo, actualmente o uso de *MathML Presentation Markup* apresenta-se como a melhor escolha para a interpretação oral de expressões matemáticas em MathML. No entanto o uso de *Presentation Markup* possui um problema de ambiguidade na semântica das fórmulas matemáticas. Por exemplo, a expressão A_2^3 é codificada em *Presentation Markup* como usando "3" e "2" como índices. Isto torna difícil a interpretação da expressão, uma vez que esta pode ser lida como: "*á de índice dois ao cubo*" ou como "*arranjos de três elementos dois a dois*". Este problema só pode ser resolvido com algoritmos de análise e interpretação mais poderosos e eficazes, o que implica mais esforço de interpretação e conversão.

O ideal seria a implementação de uma aplicação capaz de interpretar *Parallel Markup*, sendo que o significado semântico da fórmula estaria disponível para quebrar a ambiguidade na interpretação. No entanto, são raros os casos em que se encontram documentos codificados com *Parallel Markup*.

4.4.2 Interpretação de elementos e atributos de Presentation Markup

Nesta secção é abordada a necessidade de interpretação e processamento de certos elementos ou atributos do conjunto de etiquetas de marcação de *Presentation Markup*, para a conversão áudio.

Os atributos especiais: `class`, `style`, `id`, `xref`, `xlink:href` e `other`, não são tipicamente processados uma vez que estão maioritariamente associados a propriedades de estilos visuais.

[28] O grupo *OpenMath* referido na secção 4.1.1 está a realizar um trabalho mais extenso e possui um dicionário bastante vasto.

Também os atributos dos *token elements*: `mathbackground`, `mathsize`, `fontsize`, `fontweight` e `fontfamily` não são geralmente processados pelos mesmos motivos anteriormente referidos.

O processamento do atributo: `mathcolor` pode ser interessante se tivermos em conta que algumas informações poderão estar codificadas na cor que é utilizada na fórmula matemática. Por exemplo: "a expressão a azul comprime a informação da expressão vermelha". Na leitura de ambas as expressões, a aplicação precisa de referir a cor das mesmas (caso esteja presente o atributo `mathcolor`), para que o utilizador seja capaz de co-relacionar as informações da expressão com o texto anterior.

O atributo `mathvariant` é relevante no processamento de MathML. Se for usado com o valor "`italic`" estará a representar, muito provavelmente, uma função. Se for usado com o valor "`bold`" estará a representar um vector. Se "`fraktur`" for usado, então poderá estar a representar álgebra linear.

Em relação aos elementos de *MathML Presentation Markup*, em (64) foi possível reunir as seguintes conclusões:

- `maction` – não é processado em geral. No entanto pode ser interessante se for pretendido que o utilizador obtenha conhecimento acerca das possíveis acções com aquela expressão matemática.
- `maligngroup`, `malignmark` – não são processados (expressão de informação visual).
- `menclose` – processado, pois identifica um operador.
- `merror` – não é processado. Antes de uma fórmula ser processada, deve ser testada a sua validade.
- `mfenced` – é importante para que se conheçam os delimitadores da expressão matemática.
- `mfrac` – processado, pois identifica um operador.
    - possui atributo: `linethickness` – importante porque identifica se expressão é fracção ou número combinatório.
- `mglyph` – processado pois simboliza um ou mais caracteres.
    - possui atributo: `alt` – contém a descrição do símbolo.
- `mi` e `mn` – são processados pois identificam dados.

- `mlabeledtr` – é usado para exprimir o titulo da tabela.
- `multiscripts` – é processado pois identifica operadores. No entanto os seus atributos podem ser descartados.
- `mo` – é processado pois identifica operadores.
    - possui atributos: `moveablelimits`, `fence`, `separator` e `accent` – são processados pois implicam mudanças de comportamentos no `mo`.
- `mover` – é processado pois identifica a posição de limites.
- `mpadded` – não é usado, apenas exprime estilo visual.
- `mphantom` – todos os conteúdos dentro deste elemento devem ser ignorados.
- `mprescripts` – processado uma vez que identifica posições de limites.
- `mroot` – identifica um operador, logo é processado.
- `mrow` – é um dos elementos mais importantes, uma vez que fornece pistas em relação às sub expressões que compõem uma fórmula matemática (útil na colocação de pausas, por exemplo).
- `ms` – identifica um conjunto de caracteres e como tal é processado.
- `mspace` – não é processado.
- `msqrt` – identifica um operador, logo é processado.
- `mstyle` – usado apenas pelo seu atributo `displaystyle`, uma vez que este atributo pode modificar o comportamento de `mo`.
- `msub`, `msubsup` e `msup` – identificam operadores e são processados.
- `mtable`, `mtr` e `mtd` – identificam operadores e são processados. No entanto os seus atributos podem ser descartados.
- `mtext` - identifica um conjunto de caracteres e como tal é processado.
- `munder` e `munderover` – identificam operadores que posicionam limites, logo são processados.
- `none` – não é processado.

4.4.3 Interpretação de elementos e atributos de Content Markup

Nesta secção é abordada a necessidade de interpretação e processamento de certos elementos ou atributos do conjunto de etiquetas de marcação de *Content Markup*, para a conversão áudio.

À semelhança do *Presentation Markup*, os atributos especiais: `class`, `style`, `id`, `xref`, `xlink:href` e `other`, não são tipicamente processados.

O atributo `encoding` – usado para identificar o tipo de codificação de um determinado elemento – é processado tendo em conta que pode ter influência na interpretação do elemento.

O atributo `definitionURL` é uma incógnita. Apesar de ter sido implementado como uma solução que permite a expansibilidade do dicionário de elementos do *Content Markup*, nunca se sabe que tipo de informação se pode encontrar no endereço em causa. A forma como é interpretado no formato oral é imprevisível. Por isso não é recomendado.

Em relação aos elementos de *MathML Content Markup*, em (64) foi possível reunir as seguintes conclusões:

- `annotation`, `annotation-xml` e `semantics` – podem ser processados na conversão áudio, desde que o analisador possua capacidades de interpretação de outras linguagens que não o MathML.
- `apply` – é processado, uma vez que é importante para definir os limites de actuação de um operador.
- Todos os elementos que representam operadores devem ser processados e interpretados.

4.4.4 Interpretação de caracteres especiais de MathML

Existem algumas entidades ou caracteres especiais em MathML que contribuem positivamente na conversão áudio das expressões matemáticas. Por exemplo:

- `&ApplyFunction;`
- `&InvisibleTimes;`
- `&InvisibleComma;`

A entidade `&ApplyFunction;` tem como objectivo clarificar a existência de uma função. Por exemplo, na expressão $f\left(x\right)$, se for utilizada a entidade `&ApplyFunction;` a conversão áudio será: "*éfe de chis*". Sem a entidade, será lida como: "*éfe chis*".

A entidade `&InvisibleTimes;` tem uma função semelhante. Por exemplo, na expressão xy, se for utilizada a entidade `&InvisibleTimes;` a conversão será: "*chis vezes ípsilon*". Sem a entidade, será lida como: "*chis ípsilon*".

A entidade `&InvisibleComma;` tem, à semelhança das anteriores, o objectivo de separar logicamente os conteúdos. Por exemplo, na expressão m_{12}, se for utilizada a entidade `&InvisibleComma;` a conversão será lida: "*éme de índice um virgula dois*". Sem a entidade, será lida como: "*éme de índice doze*".

Isto é, verifica-se que a aplicação de cada uma destas entidades confere mais significado ou semântica à expressão matemática em causa, e torna-a menos ambígua.

Página em branco.

5 Conversores Texto-Fala e Linguagens de Marcação

Neste capítulo pretende-se apresentar sumariamente alguns conceitos básicos sobre síntese de fala e conversores texto-fala; assim como uma breve introdução a diversas linguagens de marcação usadas em tecnologias de fala, tais como: conversores texto--fala ou aplicações de reconhecimento de fala.

5.1 Síntese da Fala e Conversores Texto-Fala

As primeiras máquinas de produção de fala, muito primitivas, remontam a 1779, por C.G. Kratezenstein. Alguns anos mais tarde, em 1791, W.R. von Kempelen demonstrou uma máquina mais sofisticada, capaz de reproduzir fala contínua. Em 1835, Wheatstone melhorou a máquina de von Kempelen e construiu um dos primeiros *vocoders*, ainda manual e analógico.

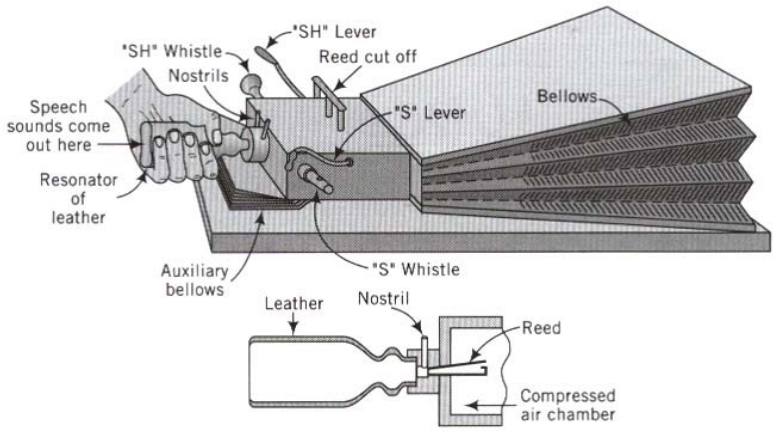


Figura 5.1 - Máquina Falante de Wheatstone (81).

Em 1939, Homer Dudley inventa o primeiro *vocoder* eléctrico e demonstra-o na feira mundial em Nova York. Ainda em 1939, Dudley propõe que os *vocoders* passem a representar os sinais de fala no domínio da frequência modelando-os através de um conjunto de filtros passa-banda. Este conjunto de filtros é excitado por ruído nas zonas não vozeadas (zonas em que as cordas vocais não vibram) e por pulsos periódicos nas zonas vozeadas (zonas em que o ar liberto pelos pulmões é colocado em vibração pela acção das cordas vocais). Actualmente, ainda existem muitos sistemas que fazem uso deste princípio básico.

Com o aparecimento dos computadores, o processamento da fala desenvolveu-se rapidamente e surgiram diversas tecnologias ao serviço da engenharia da fala.

Actualmente, existem três grandes técnicas de síntese da fala, enunciadas por ordem crescente de qualidade:

- Síntese por *Formantes*
- Síntese por *Predição Linear* (82)(83)
- Síntese por *Concatenação* (exemplo: TD-PSOLA, PSOLA (81)(84))
- Síntese por *HTS*[29] (85)

O desenvolvimento destas técnicas de síntese de fala, conjugadas com a evolução dos sistemas informáticos, conduziu ao aparecimento dos conversores texto-fala[30]. Poder-se--ia definir um conversor texto-fala como um sistema baseado em computador capaz de processar um texto e reproduzi-lo auditivamente. Um modelo arquitectural genérico para um conversor texto-fala é apresentado de seguida:

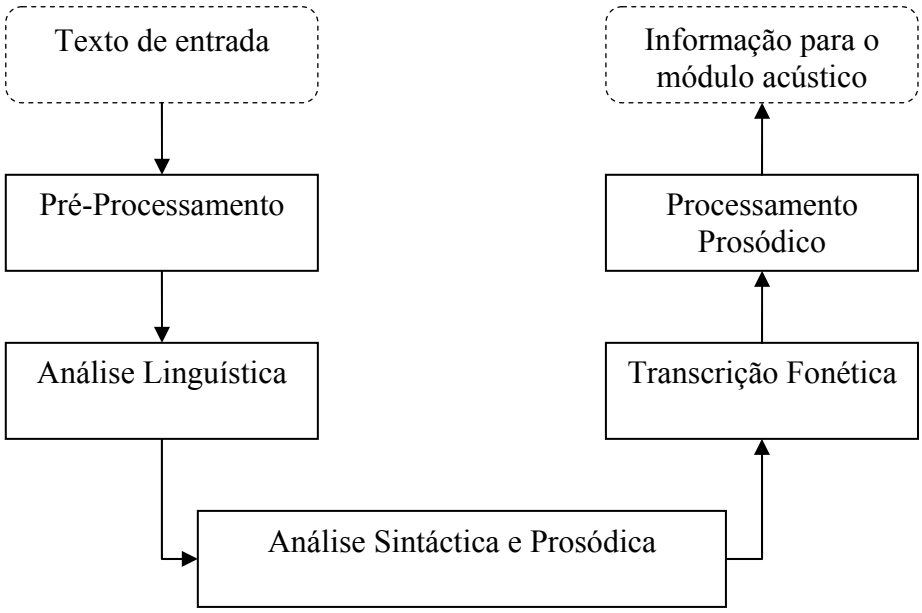


Figura 5.2 - Arquitectura básica de um conversor texto-fala.

O módulo de Pré-Processamento tem como objectivo aplicar filtros de tratamento do texto capazes de transformar os símbolos[31], que não se encontram explicitamente por extenso, na sua forma extensível. É o caso dos numerais, abreviações, acrónimos e referências de rede. Por exemplo: o numeral "3,45%" deveria de ser convertido em

[29] *Hidden Markov Model Speech Synthesis System*
[30] Na literatura inglesa, um sistema de conversão texto-fala é referenciado como um sistema TTS (*Text-to-Speech*).
[31] Entenda-se símbolo como um elemento constituinte de um texto, por exemplo: uma palavra ou um conjunto de caracteres alfanuméricos.

"*três vírgula quarenta e cinco por cento*"; a abreviação "*Prof.*" deveria de ser convertida em "*Professor*"; o acrónimo "*FEUP*" deveria de ser transformado em "*Faculdade de Engenharia Universidade do Porto*"; e o e-mail "*hfilipe@fe.up.pt*" deveria de ser transformado em "*agá efe í éle í pê é arroba éfe é ponto ú pê ponto pê tê*". O resultado final será um texto formatado e tratado.

O módulo de Análise Linguística possui uma função gramatical. Consiste num conjunto de algoritmos capazes de, para cada elemento do texto, identificarem os artigos, verbos, substantivos, adjectivos e outros elementos gramaticais. Por exemplo, na frase "*A Ana estuda matemática.*", o módulo de análise linguística identificaria e marcava "*A*" como artigo definido feminino singular; "*Ana*" como substantivo próprio feminino; "*estuda*" como verbo no indicativo presente; e "*matemática*" como substantivo comum feminino singular. As ambiguidades que restarem serão resolvidas no módulo seguinte.

O módulo de Análise Sintáctica e Prosódica consiste na aplicação de um conjunto de algoritmos que visam a marcação do texto com marcas prosódicas, nomeadamente, pausas, formantes e frequência fundamental. Exemplo: "*A Ana [pausa] estuda matemática*". Trata-se de dividir o texto em frases e definir uma forma de o ler.

O conjunto destes três módulos constitui o processamento linguístico de um conversor texto-fala.

O módulo de Transcrição Fonética tem como objectivo a transformação dos caracteres alfabéticos em símbolos fonéticos. Trata-se de uma conversão grafema-fonema, onde são aplicados símbolos fonéticos IPA[32] ou SAMPA[33]. Por exemplo, em SAMPA, a frase "*A Ana estuda matemática*" seria transcrita na seguinte frase: "*6 6n6 iStud6 m6t@matik6*".

O módulo de Processamento Prosódico consiste na consulta das marcações prosódicas, e sua utilização e ajuste na frase já transcrita fonéticamente. Posteriormente a frase marcada com variáveis de controlo prosódico é enviada para o módulo acústico que a reproduzirá sob a forma de áudio.

O conjunto destes dois últimos módulos constitui a fase de processamento acústico.

---

[32] IPA – *International Phonetic Alphabet* - http://www.arts.gla.ac.uk/ipa
[33] SAMPA – *Speech Assessment Methods Phonetic Alphabet* - http://www.phon.ucl.ac.uk/home/sampa

As aplicações de um conversor texto-fala são diversas, sendo alguns exemplos:

- Serviços de Telecomunicações – muitas empresas de telecomunicações fazem uso de conversores texto-fala nos centros de atendimento automático (*call center*), ou para serviços especializados.
- Formação e Ensino – alguns conversores texto-fala são utilizados na aprendizagem de novas línguas.
- Acessibilidade, Inclusão e Terapêutica da Fala – pessoas com necessidades especiais, nomeadamente cegos e amblíopes, deficientes motores, entre outros, fazem uso de conversores texto-fala. A síntese da fala serve como ferramenta para a terapia vocal, e como ferramenta de acessibilidade.
- Diversão e Comércio – existem produtos no mercado de brinquedos e livros que "falam".
- Multimédia – o uso de conversores texto-fala tem aberto novas fronteiras na interface pessoa-máquina.
- Investigação laboratorial – um conversor texto-fala é uma ferramenta linguística que pode ser controlada e modificada para a investigação de novas teorias e conceitos nas tecnologias de fala.

Tabela 5.1 – Lista de alguns conversores texto-fala disponíveis no mercado.

Empresa	**Nome do Produto**	**Suporta PE[34]?**	**Linguagem de marcação**	**Gratuito?**
AT&T Natural Voices	AT&T Natural Voices TTS Engines[35]	Não	SAPI4, SAPI5, SSML, JSAPI	Não
Acapela Group	InfoVox Desktop[36]	Sim	SAPI4, SAPI5	Não
Aculab	Aculab TTS[37]	Sim	SAPI, SSML	Sim
FreeTTS	FreeTTS[38]	Sim	JSAPI	Sim
Loquendo	Loquendo TTS[39]	Sim	SSML, SAPI4, SAPI5, JSML	Não
L&H TrueVoice	TrueVoice TTS[40]	Sim	SAPI4	Sim

[34] PE – Português Europeu
[35] http://www.naturalvoices.att.com/products/tts_data.html
[36] http://www.acapela-group.com/products/products.asp
[37] http://www.aculab.com/products/media_processing_resources/aculab_tts.htm
[38] http://freetts.sourceforge.net/docs/index.php
[39] http://www.loquendo.com/en/technology/TTS.htm
[40] http://www.bytecool.com/voices.htm

5.2 Linguagens de Marcação para Tecnologias de Fala

Um dos maiores desafios de um conversor texto-fala prende-se com a necessidade de interpretar correctamente as palavras e o seu contexto, de modo a que lhe seja permitido concretizar um pré-processamento eficaz e livre de erros, assim como a aplicação de padrões prosódicos adequados. Toda esta lógica contida num conversor texto-fala implica custos adicionais de processamento, e torna-se mais falível à medida que se necessita de mais inteligência por parte do conversor.

Por este motivo, e com o objectivo de desambiguar o significado das palavras e contextos, foram desenvolvidas linguagens de marcação que são aplicadas previamente no texto antes deste ser processado. Nesta secção são brevemente apresentadas algumas destas linguagens e suas principais características.

5.2.1 Microsoft SAPI

A Microsoft fundou um grupo de desenvolvimento de aplicações de fala em 1993. Em 1995, este grupo publica o SAPI[41] 1 como uma plataforma de desenvolvimento de aplicações baseada na fala para o Windows. Seguiu-se o SAPI 2, SAPI 3 e SAPI 4 em 1998. Nesta altura o grupo foi transferido para o núcleo de desenvolvimento de Speech.NET, onde publicou o SAPI 5 e o SAPI 5.1 em 2001. O grande objectivo destas aplicações é a função de interface com a plataforma do Windows, fornecendo o serviço de motores de conversão texto-fala e reconhecimento de fala. Actualmente o Windows 2000 e o Windows XP já integram nas suas distribuições a plataforma SAPI.

De entre todas as versões distribuídas pela Microsoft, o SAPI4 e o SAPI5 são os mais conhecidos e utilizados entre a comunidade empresarial e científica. Existem três grandes diferenças principais entre as duas distribuições:

- Arquitectura – No SAPI5 o módulo de conversão texto-fala encontra-se separado do módulo que guarda as propriedades e regras da voz. No SAPI4 não existe separação destes módulos.
- Linguagem de marcação – No SAPI5 a linguagem de marcação é baseada em XML, enquanto que no SAPI4 é baseada numa linguagem própria.
- Painel de controlo – No SAPI5 existe um painel de controlo centralizado com as propriedades genéricas do conversor texto-fala. O mesmo não acontece no SAPI4.

[41] Speech Application Programming Interface – SAPI - http://research.microsoft.com/srg/sapi.aspx

Tabela 5.2 - Exemplos da linguagem de marcação do SAPI 4. (86)

Propriedade	**Uso da Marca**
Volume	**\vol=65535** (valor máximo)
Velocidade	**\spd=10** , número de palavras por minuto
Frequência F0	**\pit=200** , valor em Hertz
Pausa	**\pau=1000** , número de milissegundos

Tabela 5.3 - Exemplos da linguagem de marcação do SAPI 5. (87)

Propriedade	**Uso da Marca**
Volume	**<volume level="100"> teste </volume>** 0 - mínimo 100 - máximo +/- é possível (volume relativo)
Velocidade	**<rate absspeed="-5"> teste </rate>** valor entre -10 (1/3 do valor por defeito) e 10 (3x do valor por defeito) 0 - coloca o valor por defeito
Frequência F0	**<pitch absmiddle="5" > teste </pitch>** valor entre -10 (1/3 do valor por defeito) e 10 (3x do valor por defeito) 0 - coloca o valor por defeito
Pausa	**<silence msec="3000" />** número de milissegundos desde 0 a 65535

5.2.2 SSML

SSML (88) – *Speech Synthesis Markup Language* – é uma linguagem de marcação para síntese da fala, baseada em XML, criada pela W3C[42] em 1992 e publicada em 2004, e que faz parte de um conjunto de especificações para aplicações áudio na Internet.

O objectivo do SSML é melhorar a qualidade dos conteúdos sintetizados, em que diferentes tipos de marcas afectam diferentes módulos de um conversor texto-fala. Trata-se de uma linguagem de utilização simples, em que apenas opções avançadas como o uso de fonemas e prosódia, exigem conhecimentos mais avançados.

[42] W3C – *World Wide Web Consortium* – http://www.w3.org/.

Os principais conceitos que conduziram a especificação da linguagem SSML foram:

- Consistência – possibilidade de prever o controlo da saída de voz em diversas plataformas e implementações de síntese da fala.
- Interoperabilidade – possibilidade de integração com outras tecnologias da W3C como VoiceXML[43], Aural CSS[44] e SMIL[45].
- Generalidade – suportar a síntese da fala para uma grande variedade de aplicações e contextos.
- Internacionalização – suportar o maior número de línguas possivel.
- Produção e Leitura – suportar a produção automática destes documentos SSML, e num formato de leitura facilitada.
- Implementação – permitir a implementação imediata com a tecnologia actualmente existente.

Tabela 5.4 - Exemplos da linguagem de marcação do SSML.

Propriedade	**Uso da Marca**
Volume	**<prosody volume="100"> teste </prosody>** 0 - mínimo 100 - máximo +/- é possível (volume relativo)
Velocidade	**<prosody rate="2"> teste </prosody>** valor indica o factor a multiplicar pelo valor por defeito 0 - coloca o valor por defeito
Frequência F0	**<prosody pitch="150Hz" > teste </prosody>** valor indica a frequência de base para as palavras marcadas
Pausa	**<break time="3s" />** valor em segundos ou milissegundos

5.2.3 STML

STML (89) – *Spoken Text Markup Language* – é uma linguagem de marcação, criada em conjunto pela Bell Labs[46] e a Universidade de Edinburgh[47], baseada no SGML (*Stardard Generalised Markup Language*) (90). Esta linguagem possui um conjunto de

[43] http://www.w3.org/TR/2004/REC-voicexml20-20040316/
[44] http://www.w3.org/TR/1998/REC-CSS2-19980512/aural.html
[45] http://www.w3.org/AudioVideo/
[46] http://www.bell-labs.com/
[47] http://www.cstr.ed.ac.uk/

marcas para duas finalidades distintas: descrição semântica do texto; e controlo de aplicações de texto-fala. A estrutura de um documento STML consiste no seguinte:

- Um documento STML pode ter mais do que uma linguagem.
- Cada linguagem pode consistir em mais do que um falante.
- Cada falante tem um género (exemplo: drama, leitura de prosa, leitura de poesia).

Usando esta estrutura permite-se que o estilo da síntese da fala seja mais flexível e possa alterar-se no decorrer da leitura do documento.

Tabela 5.5 - Exemplo de um documento STML.

```
<!DOCTYPE STML SYSTEM>
<STML>
 <LANGUAGE ID=ENGLISH>
  <SPEAKER ID=male1>
   <GENRE TYPE=plain>
      This is a demo text.
   </GENRE>
  </SPEAKER>
 </LANGUAGE>
</STML>
```

5.2.4 JSML & JSAPI

JSML (91) – *Java Speech API Markup Language* – é uma linguagem de marcação criada pela Sun Microsystems[48] em 1999. O seu objectivo era permitir a descrição da estrutura de um documento, condensar informação prosódica, ênfase, frequência fundamental, velocidade, volume e tantas outras características importantes na síntese da fala. Foi desenhado para ser fácil de usar e aprender, e portável entre diferentes conversores texto-fala, línguas e plataformas de tecnologias de fala. Os exemplos apresentados na Tabela 5.4 são os mesmos de JSML, uma vez que o SSML foi baseado nesta linguagem de marcação.

[48] http://www.sun.com/

Para além do JSML, existe uma plataforma de interface entre as tecnologias de fala JSML e as interfaces de utilizador ou aplicações em Java, criada pela Sun Microsystems em 1997. Esta plataforma denomina-se JSAPI (92) – *Java Speech API* – e inclui suporte para comandar sistemas de conversão texto-fala e sistemas de reconhecimento de fala.

5.2.5 SABLE

Em Setembro de 1997, três laboratórios: Sun Microsystems, Bell Labs e Universidade de Edinburgh juntam-se para criar uma linguagem de marcação para síntese de fala que compromete-se a unir três linguagens já existentes:

- SSML, *Speech Synthesis Markup Language*
- STML, *Spoken Text Markup Language*
- JSML, *Java Synthesis Markup Language*

Desta união nasce a linguagem de marcação SABLE (93)(94), desenvolvida com os seguintes objectivos:

- Permitir a marcação de texto para conversores texto-fala.
- Internacionalização: apropriada ao maior número de línguas possível.
- Facilidade de aprendizagem e uso.
- Portabilidade: possibilidade de controlar diferentes sistemas em diferentes plataformas.
- Ser extensível para permitir adaptações futuras.

Tabela 5.6 - Exemplo de um documento SABLE. (95)

```
<SABLE>
  Welcome to Sable.
  <SPEAKER gender=female age=younger> Switch between </SPEAKER>
  <LANGUAGE ID=FRA> français </LANGUAGE>
  <LANGUAGE ID=ESL-MEXICAN> español </LANGUAGE>.
  You can also <EMPH LEVEL=2.0> emphasize </EMPH> words.
  <PITCH RANGE=HIGHEST>
    Set pitch <RATE SPEED=fastest> or speech rate </RATE>
  </PITCH>
  Today is <SAYAS MODE=date MODETYPE=dmy>23/09/2005</SAYAS>
</SABLE>
```

5.2.6 ACSS & CSS 3

As *Cascading Style Sheets* (CSS)[49] surgiram pela primeira vez em 1996, CSS 1.0, através da W3C. O seu objectivo era o de criar um mecanismo simples para adicionar propriedades de estilo (fontes, cores, espaçamento, ...) a documentos na Internet. A grande vantagem destas folhas de estilo prende-se com o facto de permitirem uma separação total entre a apresentação e o conteúdo de um documento on-line.

O seu funcionamento é bastante simples e consiste num modelo de hierarquias. Por exemplo, se a um elemento que representa um parágrafo são atribuídas determinadas propriedades, todos os elementos dentro do parágrafo, que não estejam definidos na CSS, assumirão as propriedades do parágrafo como suas.

Em 1998 surge uma nova especificação, CSS 2.0, e mais tarde em 2005, CSS 2.1, que inclui um conjunto de propriedades que podem ser utilizadas como linguagem de marcação de tecnologias de fala, para controlar a informação contida no documento caso esta seja processada por um conversor texto-fala. Este conjunto de propriedades foi denominado de Aural CSS – ACSS (96).

Para a especificação destas propriedades foi desenvolvido um modelo empírico de uma caixa auditiva (Figura 5.3). Este modelo permite a especificação de propriedades 3D assim como temporais.

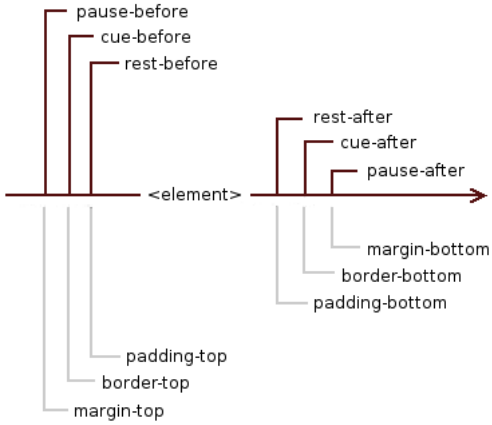


Figura 5.3 - Modelo auditivo de ACSS. (97)

É também possível variar a qualidade da fala sintetizada através de propriedades como: o tipo de voz a usar, a frequência, nível de stress, volume e velocidade.

[49] *W3C Cascading Style Sheets Homepage* - http://www.w3.org/Style/CSS/

Em Dezembro de 2004, a W3C deu inicio a um novo módulo da especificação CSS 3, denominado *CSS 3 Speech Module* (97). Este novo módulo pretende reunir diversas propriedades identificadas pela ACSS e permitir a integração com SSML.

Tabela 5.7 - Exemplos de ACSS.

```
h1, h2, h3, h4, h5, h6 {
    voice-family: paul;
    voice-stress: moderate;
    cue-before: url(ping.au)
    pause: 20ms /* pause-before: 20ms; pause-after: 20ms */
}
p.heidi { voice-balance: left; voice-family: female }
p.peter { voice-balance: right; voice-family: male }
p.goat  { voice-volume: soft }

a { cue-before: url(bell.aiff); cue-after: url(dong.wav) }
h1 { cue-before: url(pop.au) 80; cue-after: url(pop.au) 50% }
div.caution { cue-before: url(caution.wav) loud }

@phonetic-alphabet "ipa";
#tomato { phonemes: "t\0252 m\0251 to\028a " }

ul::before { content: "Start list: " }
ul::after  { content: "List end. " }
li::before { content: "List item: " }
```

5.2.7 SALT

SALT (98) – *Speech Application Language Tags* – é uma linguagem de marcação para tecnologias de fala desenvolvida por um fórum de diversas empresas e universidades, em 2001. Não é uma linguagem usada em conversores texto-fala, mas sim para aplicações baseadas em telefone, serviços de *Web*, *tablet PC's* e redes sem fios (*wireless*). É constituída por um conjunto de marcas que estende as funcionalidades de HTML[50], XHTML[51] e XML[52], e permite um acesso multimodal facilitando a interacção através de fala, teclado, rato, música, vídeo ou gráficos.

[50] W3C HTML 4.01 Specification - http://www.w3.org/TR/html4/
[51] W3C XHTML 1.1 Specification - http://www.w3.org/TR/xhtml11/
[52] W3C XML 1.1 Specification - http://www.w3.org/TR/2004/REC-xml11-20040204/

A SALT procura responder a três grandes desafios:

- Interacção com dispositivos sem fios.
- Desenvolvimento de cada vez mais aplicações com tecnologias de fala.
- Auxiliar o acesso a aplicações e serviços de Internet através do telefone.

Tabela 5.8 - Exemplo de SALT.

```
<prompt id=”bar” onbookmark=”f()” ...>
  Traveling to New York? <bookmark name=”imp_confirm” />
  There are <emph>3</emph> available flights...
</prompt>
```

5.2.8 VoiceXML

À semelhança da SALT, a W3C desenvolveu em 2000 uma linguagem de marcação – VoiceXML (99)(100) - para a criação de diálogos utilizando: síntese da fala, áudio digitalizado, reconhecimento de fala, reconhecimento de tons DTMF[53] e telefone. Não é uma linguagem para conversores texto-fala, mas contribui eficazmente para a construção de soluções baseadas em tecnologias de fala, onde a interacção multimodal é a regra, e onde um comando falado pode estar associado a uma acção através de um URI (*Universal Resource Identifier*).

Tabela 5.9 - Exemplo de um documento VoiceXML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0">
  <form>
  <field name="drink">
     <prompt>Deseja tomar café, leite, chá ou nada?</prompt>
     <grammar src="drink.grxml" type="application/srgs+xml"/>
  </field>
  <block>
     <submit next="http://www.drink.example.com/drink2.asp"/>
  </block>
 </form>
</vxml>
```

[53] DTMF – *Dual-Tone Multi-Frequency*

O exemplo da Tabela 5.9 poderia ter a seguinte interacção:

- C (computador): Deseja tomar café, leite, chá ou nada?
- H (humano): sumo de laranja
- C: Não compreendi o que disse. (mensagem por defeito)
- C: Deseja tomar café, leite, chá ou nada?
- H: Chá
- C: (continua no documento drink2.asp)

Página em branco.

6 Análise, Interpretação e Conversão de padrões num texto

Um documento técnico é geralmente composto por palavras, números, abreviaturas, siglas, expressões matemáticas, endereços de rede, esquemas, gráficos, tabelas e outros tipos de elementos. A análise e a interpretação de um texto deste género ou de outros, exige um elevado conhecimento lexical, uma análise da sua estrutura, a identificação de determinados tipos de elementos que possam ser interpretados e transformados, e uma análise semântica. A definição de padrões de pesquisa por estes elementos, é concretizada sob a forma de expressões regulares (uma das técnicas mais utilizadas na identificação de padrões num texto) (101)

Neste capítulo pretende-se abordar sumariamente alguns dos conceitos básicos sobre autómatos e transdutores de estados finitos; assim como os conceitos de expressões regulares, sendo que estas foram a base dos algoritmos desenvolvidos no âmbito desta dissertação.

6.1 Linguagens e Processadores de Linguagens

As expressões regulares são um dos mecanismos que permitem a definição de uma linguagem. Sendo assim, e para se conhecer o mundo das expressões regulares e dos autómatos é necessário primeiro compreender o contexto que os envolve, isto é, compreender o que é uma linguagem e para que servem os processadores de linguagens.

Entende-se por definição de uma linguagem: "Uma linguagem *L* sobre um alfabeto, também designado, frequentemente por vocabulário, *V*, é um conjunto de frases, em que cada frase é uma sequência de símbolos pertencentes a *V*." [54] (102), em que para se restringirem as combinações de símbolos nas frases aplicam-se regras de dois tipos:

- Sintácticas – regras focadas na estrutura da frase.
- Semânticas – regras focadas no significado da frase.

No entanto estas regras só podem ser aplicadas a linguagens formais, uma vez que as linguagens naturais não obedecem inteiramente a regras – tipicamente recorrem-se a excepções para as processar.

[54] CRESPO, Rui – Processadores de linguagens: da concepção à implementação, pág. 7.

As linguagens são analisadas por processadores de linguagens que se definem como: "Seja L_G a linguagem gerada por uma gramática G; um processador para essa linguagem, P_{LG}, é um programa que, tendo conhecimento da gramática G:

- Lê um texto (sequência de caracteres);
- Verifica se esse texto é uma frase válida de L_G;
- Executa uma acção qualquer em função do significado da frase reconhecida."[55]

Um processador de uma linguagem divide-se em dois grandes módulos:

- Módulo de Análise:
    - Análise léxical – reconhecimento e classificação de palavras e vocábulos
    - Análise sintáctica – verifica se uma frase está sintacticamente correcta
    - Análise semântica – determina o significado da frase
- Módulo de Síntese – em que se reage ao significado identificado, produzindo um determinado resultado.

São exemplos de processadores de linguagens:

- Compiladores – traduzem linguagens de programação de alto nível para código máquina.
- Interpretadores – executam os programas sem os compilar.
- Tradutores – transformam uma linguagem noutra linguagem.
- Filtros – retornam na saída o texto de entrada com alguns elementos transformados. É nesta categoria que encaixam a grande maioria dos algoritmos de análise, interpretação e conversão desenvolvidos nesta dissertação.

6.2 Expressões Regulares

Uma expressão regular é uma notação algébrica que caracteriza um conjunto de caracteres e, como tal, podem ser usadas para procurar e identificar elementos num texto, assim como definir uma linguagem de modo formal. (103) Na Figura 6.1 encontram-se resumidas as propriedades algébricas que permitem criar associações entre expressões regulares (assumem-se α, β e γ como expressões regulares).

[55] CRESPO, Rui – Processadores de linguagens: da concepção à implementação, pág. 10.

Ref.	Identidade algébrica	Designação
1	$\alpha\epsilon = \epsilon\alpha = \alpha$	*Elemento neutro da concatenação*
2	$\alpha + \emptyset = \emptyset + \alpha = \alpha$	*Elemento neutro da união*
3	$(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$	*Associatividade da união*
4	$(\alpha\beta)\gamma = \alpha(\beta\gamma)$	*Associatividade da concatenação*
5	$\alpha + \beta = \beta + \alpha$	*Comutatividade da união*
6	$\alpha + \alpha = \alpha$	*Idempotência*
7	$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$	*Distributividade da concatenação à direita*
8	$(\beta + \gamma)\alpha = \beta\alpha + \gamma\alpha$	*Distributividade da concatenação à esquerda*
9	$\alpha^+ = \alpha\alpha^* = \alpha^*\alpha$	*Relação entre operador de Kleene e fecho transitivo*
10	$\alpha^* = \epsilon + \alpha^+$	*Relação entre operador de Kleene e fecho transitivo*
11	$(\alpha + \epsilon)^+ = (\epsilon + \alpha)^+ = \alpha^*$	*Relação entre operador de Kleene e fecho transitivo*

Figura 6.1- Identidades algébricas entre expressões regulares.[56]

As expressões regulares são consideradas essenciais para que os algoritmos de processamento de texto sejam potentes, eficientes e flexíveis. A aplicação de expressões regulares permite a descrição e análise de texto, assim como, remoção, adição e manipulação de blocos de frases. (104)

Como já foi referido anteriormente, grande parte dos algoritmos de conversão desenvolvidos nesta dissertação fazem uso das expressões regulares, onde estas foram usadas sob a forma de linguagem de especificação de analisadores léxicais.

6.2.1 Breve introdução à sintaxe das expressões regulares

Para demonstrar algumas das regras e características das expressões regulares, adoptou-se a sintaxe da linguagem PERL[57] (105)(106)(107)(108), que foi a linguagem maioritariamente usada no desenvolvimento deste trabalho.

O uso de expressões regulares requer a existência de um padrão que queremos identificar, e um texto onde procurar.

Tabela 6.1 - Exemplos simples de identificação de padrões. (109)

Padrão a identificar	Aplicação prática	Comentários
/Helder/	Bom dia Helder!	encontrou 1 ocorrência
/e+/	Bom dia Helder!	encontrou 1 ocorrência
	Preencheu tudo?	encontrou 1 ocorrência
/E/	Bom dia Helder!	não existe E maiúsculo

[56] CRESPO, Rui – Processadores de linguagens: da concepção à implementação, pág. 33.
[57] PERL - *Practical Extraction and Report Language* – http://www.perl.org

Para aumentar a potencialidade das expressões regulares, estas foram complementadas com *meta-caracteres*. Entende-se um meta-caracter como um símbolo de uma expressão regular que possui um significado próprio, e que pretende definir conceitos estruturais da expressão regular.

Tabela 6.2 - Exemplos de uso de meta-caracteres em expressões regulares. (109)

Meta-caracter	Descrição	Exemplo	Aplicação prática
^	início de linha	/^Bom/	Bom dia Helder!
$	fim de linha	/Bom$/	Hoje é um dia bom
|	ou	/a|b/	Hoje é um dia bom.
			Hoje é um bom dia.
.	qualquer caracter	/Bo./	Bom dia Helder!
+	um ou mais	/ra.+/	Coisa rara não é?
*	zero ou mais	/carro*/	Bom dia Helder!
		/carro*/	Belo carro!
?	um opcional	/B(o)?m/	Bom carro.
		/B(o)?m/	É um Bmw?
(...)	captura padrão	/B(o|a)m?/	Bom dia. Captura: “o”.
		/B(o|a)m?/	Belo barco. Captura: “a”.

As expressões regulares são sensíveis ao uso de maiúsculas e minúsculas (*case--sensitive*). Caso se pretendesse encontrar todas as ocorrências de “e”, independentemente da forma como é escrito, ter-se-ia de usar um operador denominado *classe de caracteres* (“[ ]”).

Tabela 6.3 - Exemplo de *case-insensitive* numa expressão regular. (109)

Padrão a identificar	Aplicação prática	Comentários
/[E|e]+/	Bom dia Helder!	encontrou 1 ocorrência

O uso de operadores de *classe de caracteres* é muito importante nas expressões regulares, pois apresenta características muito próprias; por exemplo, os meta-caracteres apresentados anteriormente possuem comportamentos diferentes dentro de uma *classe de caracteres*. Por exemplo, o símbolo “.” significa “qualquer caracter” fora de uma classe de caracteres; dentro de uma classe de caracteres significa simplesmente “ponto”.

Existe também a possibilidade de se negar os conteúdos de uma classe de caracteres com recurso ao meta-caracter "^".

Tabela 6.4 - Exemplo da negação de uma classe de caracteres. (109)

Padrão a identificar	Aplicação prática	Comentários
/[^e]+/	Boneca!	ignora o "e" .

A Tabela 6.5 apresenta alguns exemplos em que se usam abreviaturas de padrões, nomeadamente "\d" e "\w", numa expressão regular. Note-se que o espaço é também um caracter.

Tabela 6.5 - Abreviaturas de padrões numa expressão regular. (109)

Padrão a identificar	Aplicação prática	Comentários
/[0-9]+/	Hoje é dia 25.	reconhece todos os dígitos.
/[a-z]+/	Bom dia Helder!	reconhece caracteres alfa minúsculos.
/[a-zA-Z]+/	Bom dia Helder!	reconhece caracteres alfa sejam minúsculos ou maiúsculos.
/[a-zA-Z0-9]+/	Bom dia Helder! Foram 5.	reconhece caracteres alfanuméricos.
/\d+/	Hoje é dia 25.	é equivalente a /[0-9]+/ .
/\w+/	Bom dia Helder!	é equivalente a /[a-zA-Z]+/ .

Existem ainda outros meta-caracteres, quantificadores e modificadores[58] que são usados na compilação de uma expressão regular em PERL, no entanto, não serão referidos neste documento[59].

[58] Os quantificadores e modificadores são utilizados por uma expressão regular como informação pré-compilação, para modificar o comportamento de alguns meta-caracteres ou mesmo o comportamento do autómato finito implementado pela expressão regular.
[59] Podem encontrar-se boas referências para o estudo de expressões regulares em (104)(105)(107)(108).

Tabela 6.6 – Aplicação de expressões regulares na prática: "Procura e Substitui".

Problema:
Um funcionário possui um modelo de questionário e quer preenchê-lo com os dados dos contribuintes.

Questionário:
```
$questionario=”Nome: =NOME=\n”;
$questionario.=”Morada: =MORADA=\n”;
$questionario.=”Telefone: =TELEFONE=\n”;
```

Dados do contribuinte:
```
$nome=”José Carlos”;
$morada=”Rua Miguel Silveira, 23 - Porto”;
$telefone=”22 928 16 69”;
```

Código em PERL:
```
$questionario = ~s/=NOME=/$nome/g;
$questionario = ~s/=MORADA=/$morada/g;
$questionario = ~s/=TELEFONE=/$telefone/g;
```

Resultado Final:
```
Nome: José Carlos
Morada: Rua Miguel Silveira, 23 - Porto
Telefone: 22 928 16 69
```

Existem duas formas de aplicar expressões regulares no PERL:

- Identificação de um padrão – uso da nomenclatura `m/<expressão regular>/` - implementação de um autómato de estados finitos.
- Substituição de padrão por outro – uso da nomenclatura `s/<expressão regular a ser substituída>/<expressão regular que a vai substituir>/` - implementação de um transdutor de estados finitos.

Na Tabela 6.6 exemplifica-se o uso da substituição de um padrão por outro, enquanto que no exemplo da Tabela 6.7 se exemplifica o uso de identificação de um padrão.

Tabela 6.7 - Aplicação de expressões regulares na prática: "Validação".

Problema:

Validar endereços de rede, Internet e correio electrónico num formulário web.

Dados a validar:

```
$email = "helder.filipe@sapo.pt";
$url = "http://lpf-esi.fe.up.pt/~audiomath";
$ip = "345.255.256.192";
```

Código PERL:

```
if ($email !~ m/^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-
9]{1,3})(\]?)$/igs)
{ return "inválido"; } else { return "válido"; }

if ($url !~ m/^(http|ftp|https):\/\/[\w]+(\.[\w]+)([\w\-
\.\,\@\?\^\=\%\&amp;\:\/\~\+\#]*[\w\-
\@\?\^\=\%\&amp;\/\~\+\#])?$|^[\w]+(\.[\w]+)([\w\-
\.\,\@\?\^\=\%\&amp;\:\/\~\+\#]*[\w\-
\@\?\^\=\%\&amp;\/\~\+\#])?$/igs)
{ return "inválido"; } else { return "válido"; }

if ($ip !~ m/^(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-
9]{1}|[1-9])\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-9]{1}[0-
9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-9]{2}|[1-
9]{1}[0-9]{1}|[1-9]|0)\.(25[0-5]|2[0-4][0-9]|[0-1]{1}[0-
9]{2}|[1-9]{1}[0-9]{1}|[0-9])$/igs)
{ return "inválido"; } else { return "válido"; }
```

Resultado Final:

```
$email é válido.
$url é válido.
$ip é inválido.
```

6.2.2 Funcionamento das expressões regulares

Foi já referido anteriormente que uma expressão regular especifica uma linguagem, especificação essa que dá origem na prática a autómatos de estados finitos (FSA[60]).[61] É através da representação de FSAs, em particular sob a forma de grafos, que nesta dissertação é explicado o conceito e algoritmia das expressões regulares.

Notação (110) utilizada nos grafos de autómatos finitos:

- Círculo – um estado do autómato; se tiver a indicação *start*, trata-se do estado inicial;
- Círculo duplo – estado final do autómato;
- Seta – transição entre estados;
- ε – transição *epsilone* – surge quando pode existir mais do que um destino numa transição, ou quando o símbolo num estado é clonado.

Considere a expressão regular `/ab/` e admita que a Figura 6.2 corresponde ao autómato finito que a implementa. De seguida é descrito o algoritmo (111) que o autómato realiza para identificar o padrão: `ab`.

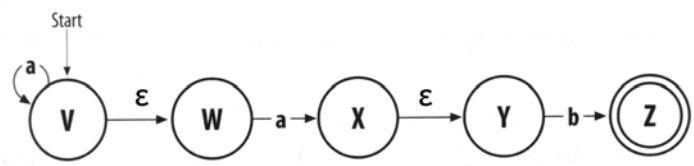


Figura 6.2 - Máquina de estados finita para expressão regular.

Considere que o estado inicial é representado por V e o estado final por Z; e que a máquina possui duas *transições-epsilone* e três transições de estados com condições. O autómato será "alimentado" por testemunhos (*tokens*), e as transições irão fazer os testemunhos mudarem de estado consoante o padrão que se está a analisar.

A colocação de um testemunho no estado inicial V provoca uma clonagem do testemunho e uma transição do clone do mesmo para W. Isto dá-se devido à transição--epsilone. Assim sendo, passam a existir dois testemunhos: um em V e outro em W. O testemunho em W faz com que o autómato leia o padrão de entrada, neste caso "`a`".

---

[60] FSA – *Finite State Automata*

[61] Existem ferramentas que permitem a criação automática de autómatos finitos a partir de especificações de expressões regulares, tais como: o Lex (http://dinosaur.compilertools.net/) e o Flex (http://www.gnu.org/software/flex/).

A leitura de "a" faz com que o testemunho em V transite para V novamente, e o testemunho em W transite para X.

Devido à transição-epsilone, o testemunho em X é clonado e transita do estado X para o estado Y; e o testemunho em V é clonado e transita do estado V para o W. Existem, neste momento, quatro testemunhos no sistema: um em V, um em W, um em X e outro em Y. O testemunho em Y faz com que o autómato leia o padrão de entrada, neste caso "b". A leitura de "b" faz com que o testemunho em Y transite para Z fazendo o autómato atingir o seu estado final. Neste ponto a máquina de estados finitos acabou de ler a entrada, e o que leu foi "ab", respondendo por isso com sucesso. Os testemunhos em excesso no autómato são descartados.

Resumo das regras para interpretação do funcionamento da máquina de estados:

1. Para começar é colocado um testemunho (*token*) no estado inicial.
2. Se existir uma transição-epsilone de X para Y, o testemunho é clonado ficando um testemunho em X e outro em Y.
3. Sempre que a máquina lê um caracter de entrada "c", todos os testemunhos são movidos. Se o testemunho está num estado que possui uma transição rotulada com "c", então esse testemunho transita e segue a regra 2 se apropriado. Se existir mais do que uma transição, o testemunho é clonado por cada uma das transições. Se o testemunho está num estado com uma transição cujo rótulo não é "c", então o testemunho é removido.
4. Quando toda a entrada é lida, a máquina de estados responde com sucesso se existir um testemunho no estado final, ou insucesso em caso contrário.

6.3 Autómatos de Estados Finitos (FSA)

O modelo lógico mais importante para o processamento computacional da fala e da linguagem é o Autómato.

A teoria de autómatos é o estudo dos dispositivos de computação abstractos, ou "máquinas". A *máquina de Turing* (112) é um exemplo disso, e foi o embrião em 1930, para o início do estudo, investigação e desenvolvimento da teoria destes modelos computacionais.

Os autómatos são essenciais para o estudo dos limites da computação: problemas que um computador pode resolver e os que não pode resolver ou tratar.

Os principais tipos de autómatos são:

- Autómato de estados finitos (FSA – *Finite State Automaton*)
- Transdutores de estados finitos (FST – *Finite State Transducer*)
- Transdutores pesados (WFST – *Weighted Finite State Transducer*)
- Cadeias escondidas de *Markov* (HMM – *Hidden Markov Models*)

Um autómato de estados finitos (FSA) descreve um sistema ou componente com estados e transições entre estados, provocadas pela resposta a entradas do sistema; em que objectivo de um estado é memorizar uma porção relevante da história do sistema, e com a vantagem destes estados serem de número reduzido (pode-se implementar o sistema com um conjunto fixo de recursos).

Um transdutor de estados finitos (FST) é uma extensão de um autómato de estados finitos. O seu comportamento é em tudo semelhante ao de um FSA, com a excepção de que os transdutores possuem a capacidade de actuar sobre a entrada transformando-a. Se o FST for usado no modo de funcionamento do tipo conversor ou interlocutor, isso significa que dois subconjuntos de caracteres são mapeados entre si.(103)

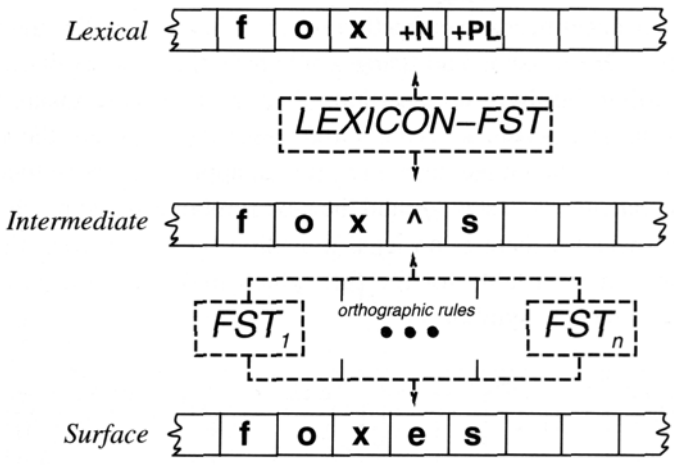


Figura 6.3 - Exemplo de mapeamento realizado por FSTs. (103)

Os transdutores de estados finitos pesados (WFSTs) (113)(114) são uma extensão dos FSTs, em que as transições encontram-se associadas a probabilidades de qual o caminho a seguir pelo autómato.

As cadeias escondidas de Markov (HMMs) (115) são um caso particular dos WFSTs, em que a sequência de caracteres de entrada identifica univocamente o estado para o qual o autómato transita. Assim sendo, são úteis principalmente quando se tratam de sequências não ambíguas. (103)

Neste documento apenas serão abordados com um pouco mais de detalhe os autómatos de estados finitos.

Existem duas formas de representação de um autómato de estados finitos: usando tabelas; ou usando grafos. Ambas as formas descrevem as transições e os estados de um autómato. Neste documento, adoptou-se pelo uso de grafos para a representação dos FSAs. Num FSA, existem duas grandes estratégias ou comportamentos de processamento:

- Determinísticos (DFA[62]): significa que o autómato não pode estar em mais de um estado em qualquer instante.
- Não-Determinísticos (NFA[63]): significa que o autómato pode estar ao mesmo tempo em vários estados. Possuem transições-epsilone[64].

Para se converter um NFA num DFA, usa-se uma metodologia que consiste em atribuir a um estado de um DFA, um subconjunto de estados de um NFA.

O trabalho realizado nesta dissertação foi baseado, em grande parte, na linguagem de codificação PERL, onde os autómatos finitos, traduzidos pelas expressões regulares, são do tipo NFA. Apesar de os autómatos do tipo NFA não serem adequados ao uso de modelos de reconhecimento de linguagens regulares, uma vez que podem possuir mais do que um estado em cada instante, o PERL resolve este problema implementando um conceito de "voltar atrás" – *backtracking* (104). Este conceito consiste num retrocesso de uma solução de identificação (*matching*) caso esta falhe, regressando aos estados anteriores à decisão que falhou e escolhendo outro caminho. A qualidade da expressão regular contribui bastante para evitar que este conceito seja aplicado diversas vezes durante a identificação de um padrão – quanto mais "voltar atrás" existir, mais lento é o processo.

---

[62] *Deterministic Finite Automaton*
[63] *Non-Deterministic Finite Automaton*
[64] Ver secção Funcionamento das expressões regulares6.2.2.

Tabela 6.8 - Comparação entre autómatos finitos dos tipos DFA e NFA.

DFA	NFA
Diferenças na pré-compilação	
• Antes de iniciar o processo de pesquisa, o algoritmo é compilado.	• Antes de iniciar o processo de pesquisa, o algoritmo é compilado.
• Mais lento que NFA, exige mais memória.	• A compilação é mais rápida que um DFA e requer menos memória.
Diferenças na rapidez de identificação (*match*)	
• Tão rápido como NFA em condições normais.	• Tão rápido como DFA em condições normais.
• A rapidez não depende da expressão regular usada.	• A rapidez depende bastante da expressão regular usada. É por isso aconselhável a construção de uma expressão regular eficiente.
Diferenças em como é feita a identificação (*match*)	
• Procura o maior caminho que seja sucesso para uma identificação.	• Procura o maior caminho que seja sucesso para uma identificação.
Diferenças de capacidades	
• Possui menos capacidades que um NFA.	• Suporta muitas capacidades que um DFA não possui, tais como: 1. Captura de texto para guardar como referência, assim como registar a posição de captura. 2. Possui capacidades de "observar o que está para trás e o que está para a frente". 3. Possui quantificadores de expressões regulares para implementar algoritmos não-gananciosos.
Diferenças na facilidade de implementação	
• Simples de implementar.	• Simples de implementar.

A Tabela 6.9 reúne um conjunto de programas e os respectivos tipos de autómatos que esses programas utilizam:

Tabela 6.9 - Alguns programas e seus motores FSA. (116)

Tipo de autómato	Programas
DFA	awk, egrep, flex, lex, MySQL, Procmail
NFA Tradicional	GNU Emacs, Java, grep, less, more, .NET, PCRE, PERL, PHP, Python, Ruby, sed, vi
POSIX NFA	mawk, GNU Emacs, Mortice Kern
Híbrido NFA/DFA	GNU awk, GNU grep/egrep, TCL

6.3.1 Conversão de uma expressão regular num autómato finito

O algoritmo de criação de um autómato finito do tipo NFA a partir de uma expressão regular foi publicado pela primeira vez por *Thompson* (111) em 1968.

A relação entre expressões regulares e autómatos finitos é possível através de uma classe de linguagens denominadas de regulares. O facto destas duas notações (expressões regulares e autómatos) representarem a mesma classe de linguagens, permite a conversão de expressões regulares em autómatos finitos. O inverso é mais complicado e nem sempre possível (116).

Vejamos alguns exemplos:

Tabela 6.10 - Autómato finito equivalente da expressão regular /a/ .

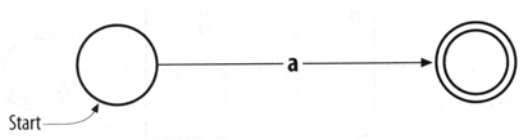
Expressão regular: `/a/`
Tem como objectivo identificar o caracter "a".
Autómato finito equivalente


Tabela 6.11 - Autómato finito equivalente da expressão regular // .

Expressão regular: `//`
Tem como objectivo identificar uma sequência de caracteres vazia.
Autómato finito equivalente


Tabela 6.12 - Autómato finito equivalente da expressão regular /P|Q/ .

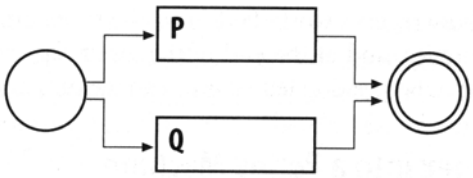
Expressão regular: `/P|Q/`
Tem como objectivo executar a expressão P ou a expressão Q.
Autómato finito equivalente


Tabela 6.13 - Autómato finito equivalente da expressão regular /PQ/ .

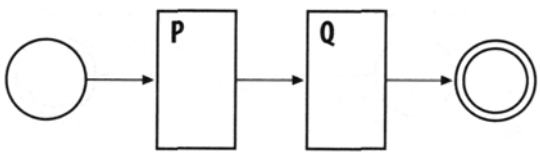
Expressão regular: `/PQ/`
Tem como objectivo executar a expressão P e Q.
Autómato finito equivalente


Tabela 6.14 - Autómato finito equivalente da expressão regular /perl/ .

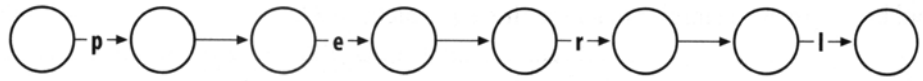
Expressão regular: `/perl/`
Tem como objectivo identificar a sequência de caracteres "`perl`".
Autómato finito equivalente


Tabela 6.15 - Autómato finito equivalente da expressão regular /P+/ .

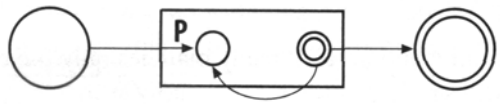
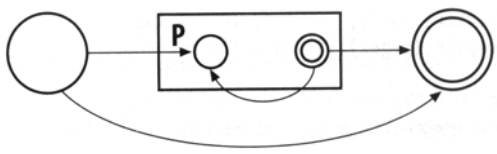
Expressão regular: `/P+/`
Tem como objectivo executar pelo menos uma vez a expressão P.
Autómato finito equivalente


Tabela 6.16 - Autómato finito equivalente da expressão regular /P*/ .

Expressão regular: `/P*/`
Tem como objectivo não executar P ou executá-lo uma vez ou mais do que uma vez.
Autómato finito equivalente


A partir destes modelos básicos torna-se possível construir autómatos finitos equivalentes a expressões regulares complexas.

Página em branco.

7 Motor de Análise, Interpretação e Conversão - AudioMathENGINE

Um conversor texto-fala em geral não está preparado para ler números em formatos além do básico, converte poucas abreviações e acrónimos, e não conhece quaisquer expressões matemáticas nem tão pouco a semântica destes elementos; isto é, limita-se a reproduzir em áudio os caracteres que reconhece, quer estes façam sentido ou não.

Foi já discutido, num capítulo anterior, a necessidade de existência de um módulo de pré-processamento capaz de tratar o texto a ser sintetizado, de modo a torná-lo "legível" para um conversor texto-fala. O seu comportamento consiste na aplicação de filtros de tratamento de texto capazes de transformar os símbolos que não se encontram explicitamente por extenso, na sua forma por extenso.

A frase:
"*O Prof. João dá aulas, às 9h00 na sala 123, na FEUP.*"
Possui uma abreviação social ("*Prof.*"), um numeral temporal ("*9h00*"), um numeral cardinal ("*123*") e uma sigla ("*FEUP*").
Estes elementos seriam tratados e convertidos, transformando a frase em:
"*O Professor João dá aulas, às nove horas na sala cento e vinte e três, na Faculdade de Engenharia Universidade do Porto*".

Figura 7.1 - Exemplo de pré-processamento.

No entanto, a maioria dos conversores texto-fala possui módulos de pré-processamento ineficazes ou de comportamento simples e básico. Também não existem, até à escrita desta dissertação, módulos de pré-processamento capazes de lidar com expressões matemáticas. Sendo assim, procurou-se exteriorizar o módulo de pré-processamento de um conversor texto-fala, e desenvolver uma aplicação capaz de fazer um tratamento eficaz num documento contendo diversos elementos, tais como: numerais, abreviações, acrónimos, referências de rede e expressões matemáticas. A esta aplicação deu-se o nome de AudioMathENGINE.

O AudioMathENGINE possui as seguintes características técnicas:

- Desenvolvido em PERL – módulos disponíveis para ambientes LINUX/UNIX e WINDOWS.
- Disponível sob quatro formas distintas:
  - *ActiveX DLL*[65]– para aplicações em *Visual Basic* ou *C/C++*.
  - *Componente .NET* – para aplicações .NET (usado durante o trabalho realizado no âmbito desta dissertação).
  - *Interface CGI* – para aplicações on-line, ou em ambientes LINUX/UNIX, através do uso de PERL.
  - *Executável EXE* – para aplicações de linha de comando em WINDOWS.
- Arquitectura modular, extensível e configurável – permite facilmente o uso das suas bibliotecas noutras aplicações ou sistemas, se tal for requerido; assim como a implementação de novas funcionalidades sem grandes custos de manutenção.
- Diversos módulos lógicos para:
  - Numerais, Abreviações, Acrónimos, Referências de Rede
  - Expressões Matemáticas e sua Navegação
  - Reconhecimento Automático
  - HTML e páginas on-line
  - Dicionários e Bibliotecas auxiliares
- Disponível para a língua: Português Europeu (espera-se de futuro a implementação de outras línguas).
- Especificação, configuração e modos de utilizador implementados parcialmente com recurso a uma linguagem de marcação – AKML (*AudioMath Knowledge Markup Language*)[66].
- Suporte nativo de *MathML Presentation Markup*
- Suporte de *MathML Content Markup*, através de uma folha de estilo de transformação XML (XSLT).
- Suporte parcial de Unicode.

[65] *Dynamic Link Library*
[66] Ver secção 7.2

As aplicações de uma ferramenta como o AudioMathENGINE são diversas, tais como:

- Integração em sistemas informáticos, promovendo a sua acessibilidade;
- Utilização como ferramenta didáctica da matemática e das ciências exactas;
- Aplicação isolada de pré-tratamento de documentos para conversores texto-fala;
- Desenvolvimento de serviços de acessibilidade.

Apesar deste trabalho ser a continuação de um projecto iniciado no estágio de licenciatura do autor, diversos melhoramentos e novas funcionalidades[67] foram introduzidas: a implementação de uma linguagem de marcação própria (*AudioMath Knowledge Markup Language – AKML*) para a definição e o tratamento de dados no motor de análise, interpretação e conversão de texto; a reestruturação da interface de ligação com aplicações externas; o melhoramento dos dicionários internos (abreviações e acrónimos); o suporte para tratamento de páginas on-line; o suporte para *Presentation Markup* e *Content Markup*; a implementação de algoritmos de navegação das fórmulas matemáticas; e a reestruturação completa dos algoritmos de reconhecimento automático, assim como do módulo de interpretação e conversão de expressões matemáticas.

7.1 Arquitectura e Desenho

Foi já referido na secção anterior que a aplicação AudioMathENGINE se encontra estruturada numa arquitectura modular, composta por camadas ou estruturas lógicas distintas. De facto, o estilo de arquitectura utilizado baseia-se numa hibridez entre o estilo em camadas (*layered*) e o estilo de função principal e sub rotinas (*main program & subroutines*) (117).

Uma redefinição da arquitectura utilizada para uma abordagem orientada a objectos (*object-oriented*) foi considerada, mas não implementada por restrições temporais e práticas (custos elevados na reestruturação), uma vez que o código se baseava num projecto antigo bastante extenso e estável. Apesar disso, a arquitectura actual do AudioMathENGINE apresenta características semelhantes, de reusabilidade, modularidade e flexibilidade, às de um sistema orientado por objectos.

[67] Foram implementadas mais de oito mil linhas de código em relação ao trabalho desenvolvido anteriormente.

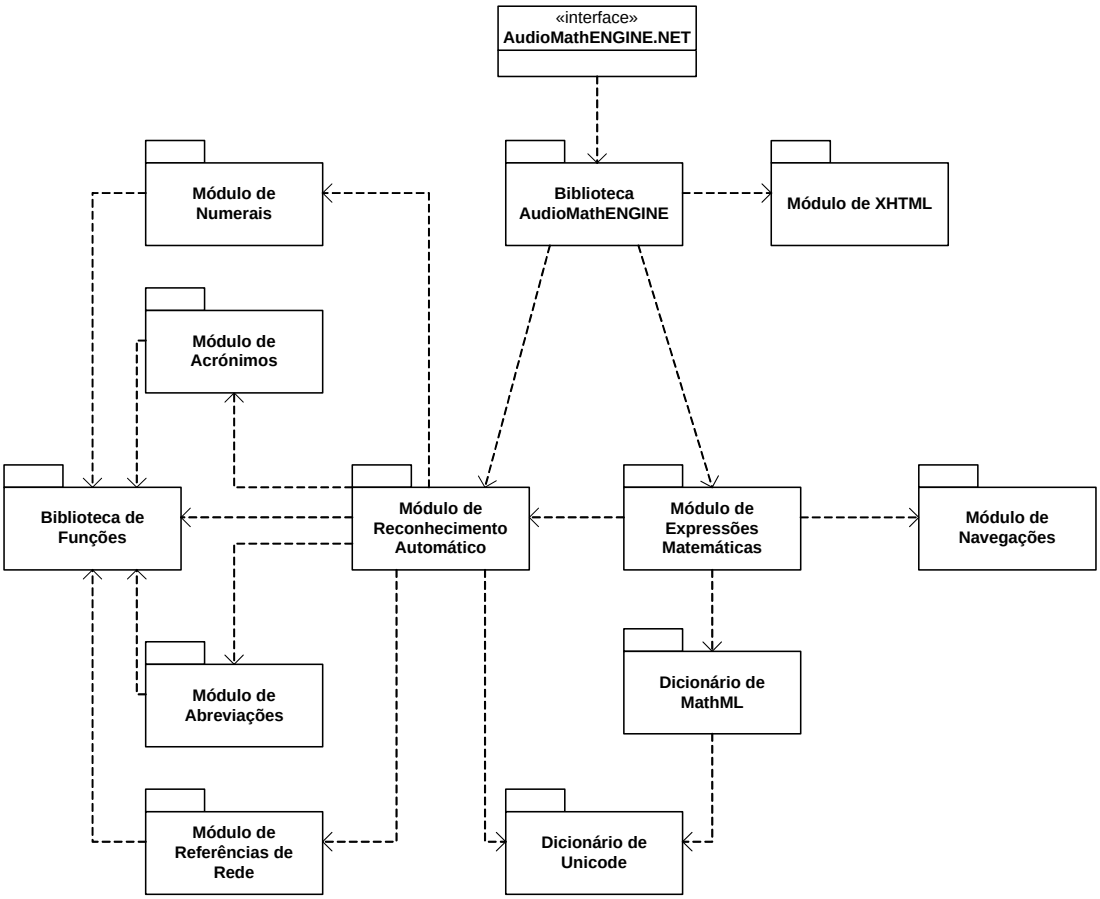


Figura 7.2 - Arquitectura de "alto nível" do AudioMathENGINE (componente .NET).

A Figura 7.2 mostra a arquitectura de "alto nível" do AudioMathENGINE, onde se podem verificar os diferentes módulos deste motor de análise, identificação e conversão; assim como as dependências entre eles. Estes módulos encontram-se compilados no componente .NET desenvolvido neste trabalho, sendo eles:

- *Módulo de Numerais* - `numerais.pm` – conjunto de algoritmos para identificação e conversão de numerais de diversos tipos.
- *Módulo de Abreviações* – `abreviacoes.pm` – conjunto de algoritmos para identificação e conversão de abreviaturas.
- *Módulo de Acrónimos* – `acronimos.pm` – conjunto de algoritmos para identificação e conversão de siglas ou acrónimos.
- *Módulo de Referências de Rede* – `referenciasrede.pm` – conjunto de algoritmos para identificação e conversão de diversos tipos de referências de rede.
- *Módulo de Expressões Matemáticas* – `pmathmlPT.pm` – análise, interpretação e conversão de MathML.

- *Módulo de Reconhecimento Automático* – `autorecon.pm` – conjunto de algoritmos de reconhecimento automático de diversos elementos constituintes de um texto.
- *Módulo de Navegação* – `navegacao.pm` – conjunto de algoritmos que identificam os pontos de navegação de uma expressão matemática.
- *Módulo de XHTML* – `xhtml.pm` – algoritmo que lida com textos de páginas on-line.
- *Dicionário de Unicode* – `mathml_to_unicode.pm` – conjunto de definições que converte entidades MathML em Unicode.
- *Dicionário de MathML* – `mathml_dictionary.pm` – conjunto de definições que converte referências de Unicode, operadores e funções MathML, em frases ou palavras, em Português Europeu.
- *Biblioteca AudioMathENGINE* – `audiomathlib.pm` – define a arquitectura principal do AudioMath.
- *Biblioteca de funções* – `tabela.pm` e `biblioteca.pm` – tabelas de conversão e conjuntos de funções auxiliares.
- *Interface AudioMathENGINE.NET* – `AudioMathEngineNET.pm` – definição da interface do componente .NET.
- *Interface AudioMathENGINEWEB* – `AudioMathEngineWeb.pm` - definição da interface para aplicações CGI.
- *Interface AudioMathENGINEDLL* – `AudioMathEngineDLL.pm` – definição da interface do componente ActiveX.
- *Interface AudioMathENGINE* – `AudioMathEngine.pl` – definição da interface para aplicações de linha de comando.

A interface .NET implementada no AudioMathENGINE foi definida segundo o padrão de desenho *Singleton* (apenas é permitida uma instância da classe) (118), de forma a salvaguardar as propriedades do motor enquanto instanciado por uma aplicação.

Além dos módulos que constituem a arquitectura do AudioMathENGINE, esta aplicação também interage com ficheiros no sistema operativo (AKML), e com a aplicação gráfica (AudioMathGUI[68]).

[68] Esta aplicação é descrita no capítulo seguinte.

A interacção com os ficheiros é apenas de leitura e acontece apenas na inicialização do componente .NET. Trata-se de uma leitura de configurações e modos de utilizador definidos pelo AudioMathGUI. A Figura 7.3 descreve esta interacção.

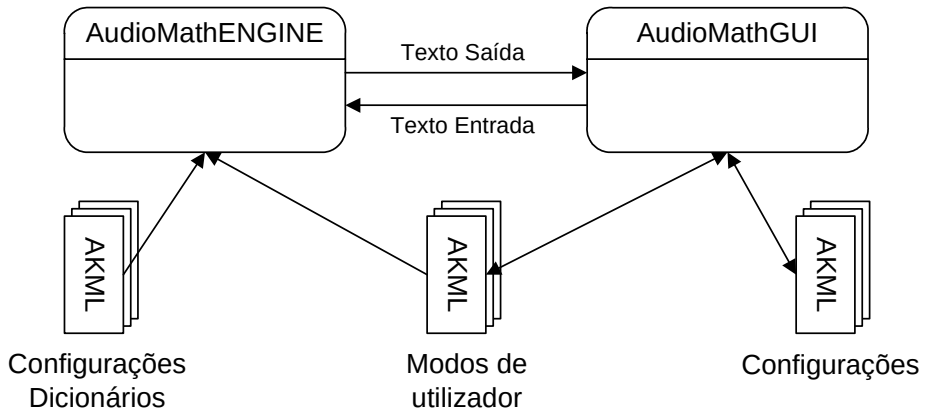


Figura 7.3 - Interacção entre AudioMathGUI, AudioMathENGINE e ficheiros AKML.

A Tabela 7.1 resume a API (*Application Programming Interface*) disponibilizada pelo AudioMathENGINE.NET (utilizado nesta dissertação) para ligação com outras aplicações.

Tabela 7.1 - API do Componente .NET.

API do Componente .NET
`int AudioMathEngineNET() - Construtor da classe. Inicializações.`
`str GetURL(str URL) – Recolhe o código fonte de uma página on-line.`
`int Convert(str InputText) – Aplica os diversos algoritmos de análise, interpretação e conversão. Transforma o texto inicial num texto "legível" por um conversor texto-fala.`
`str GetResult() – Devolve o texto convertido sob a forma de um documento AKML.`
`str CleanAKML(str InputText) – Remove todos os elementos AKML que se encontrem num determinado documento, e devolve o documento transformado.`

7.2 AudioMath Knowledge Markup Language - AKML

Tratando-se o AudioMathEngine e AudioMathGUI de duas aplicações com alguma complexidade assumida, que procuram interagir entre si, foi necessário implementar uma linguagem própria do tipo XML que facilitasse a comunicação e configuração de ambas as aplicações. A esta linguagem foi dado o nome de *AudioMath Knowledge Markup Language – AKML*.

A área de actuação do AKML cinge-se às seguintes funcionalidades:

- Configuração das aplicações AudioMathENGINE e AudioMathGUI.
- Suporte de navegação de fórmulas matemáticas.
- Marcação de textos pré e pós processados pelo AudioMathENGINE.
- Suporte de dicionários de abreviações e acrónimos.

Deste modo as aplicações tornam-se mais flexíveis e configuráveis, permitindo adaptações externas sem necessidade de recompilação do código. Por outro lado, a leitura dos ficheiros AKML implica acesso ao disco que por sua vez implica custos de processamento. No entanto, este acesso é feito apenas uma vez na inicialização das aplicações evitando assim custos desnecessários. Torna-se assim necessário possuir uma boa gestão de memória, sendo que os objectos, propriedades e configurações serão lidas uma vez do disco e guardadas em memória para posterior utilização e configuração dos algoritmos de análise, interpretação, conversão e navegação.

Procurou-se arquitectar uma linguagem XML que reaproveitasse ideias já concebidas pelo autor durante os anos de investigação no LPF-ESI[69]. O AKML procura ser uma linguagem simples e eficaz na marcação de informação, permitindo facilmente abstrair a semântica da mesma. Embora a finalidade do XML não seja ser manipulado por seres humanos, mas sim por máquinas, procurou-se desenvolver uma linguagem legível (*human-readable*) para facilitar a construção e a marcação de documentos.

O AKML foi definido como uma linguagem de marcação semântica e estrutural tanto para resultados AudioMathENGINE (conversões e navegação), como para fonte de dados do AudioMathENGINE e AudioMathGUI (configurações e dicionários).

Para a validação de documentos construídos com AKML foi desenvolvido um *XML Schema* (disponível no Anexo C) com a definição dos elementos e atributos em causa, e as regras de interacção entre os mesmos.

[69] Laboratório de Processamento da Fala, Electroacústica, Sinais e Instrumentação

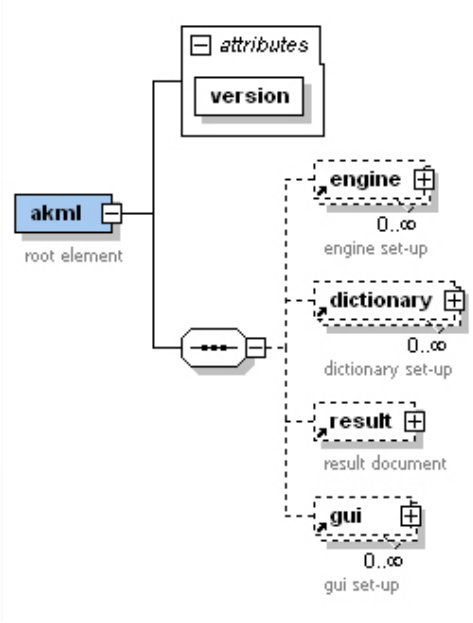


Figura 7.4 - Arquitectura base do AKML.

O elemento raiz de um documento AKML denomina-se `akml` e possui como atributos:

- `version` – versão da linguagem AKML utilizada;
- `xmlns:xsi` – atributo próprio de XML *Schemas* que define o *namespace* usado;
- `xsi:noNamespaceSchemaLocation` – atributo próprio de XML *Schemas* que define a localização do mesmo.

```
<akml version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="../schema/AKML.xsd">
  <dictionary name="acronym" lang="pt">
    <definition alias="Organização das Nações Unidas">ONU</definition>
    <definition alias="Imposto de Valor Acrescentado">IVA</definition>
    <definition alias="Sport Lisboa e Benfica">SLB</definition>
  </dictionary>
</akml>
```

Figura 7.5 - Exemplo de um documento AKML.

A linguagem AKML pode-se subdividir em quatro grandes conjuntos de etiquetas de marcação:

- *Configuração* – conjunto de marcas que descrevem as opções de configuração para o AudioMathENGINE e o AudioMathGUI.
- *Dicionário* – conjunto de marcas que descrevem um grupo de definições a serem usadas pelo AudioMathENGINE.

- *Conversão e Resultados* – conjunto de marcas que assinalam os resultados das conversões dos algoritmos do AudioMathENGINE.
- *Navegação* – conjunto de marcas (incluídas nas etiquetas de conversão e resultados) que definem a estrutura de navegação de uma expressão matemática, implementada pelo AudioMathENGINE.

As secções seguintes descrevem com mais detalhe os diferentes conjuntos de etiquetas de marcação do AKML relacionadas com o AudioMathENGINE (o uso de AKML no AudioMathGUI será abordado no capítulo seguinte).

7.2.1 Etiquetas de marcação para Configuração

Existem 2 tipos distintos de configuração no AudioMath:

- *Modos de Utilizador* – `usermodes.xml`: estas marcas dizem respeito a um conjunto de opções dos algoritmos de conversão. Na realidade permitem ao utilizador escolher como gostaria que determinados elementos fossem convertidos. Por exemplo, um número decimal pode ser lido na totalidade por extenso, mas também se pode ler a parte inteira por extenso e a parte decimal soletrada.
- *Inicializações* – `config.xml`: estas marcas dizem respeito a um conjunto de definições gerais para as aplicações AudioMathENGINE e AudioMathGUI.

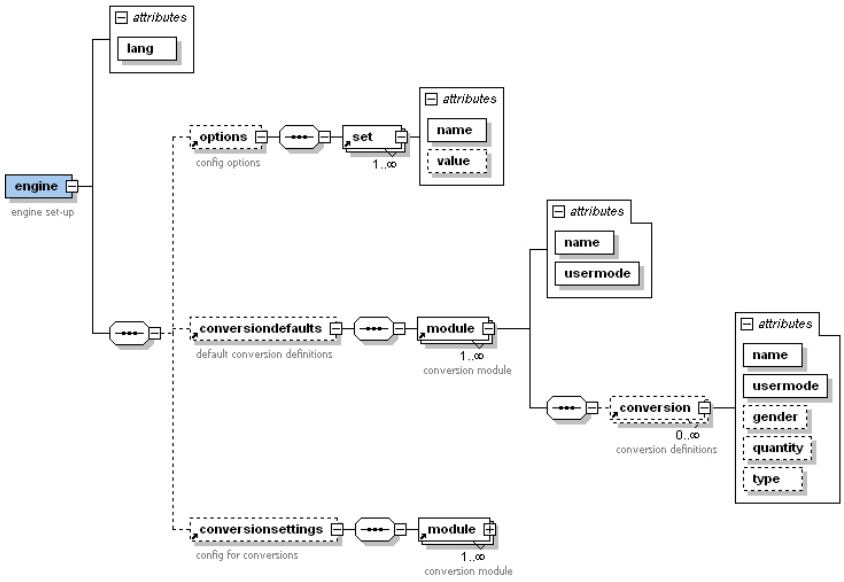


Figura 7.6 - Estrutura AKML para configurações do AudioMathENGINE.

Segue-se uma descrição dos elementos e atributos utilizados:

`<engine lang="pt|en|fr|de|...">...</engine>`

Indica um bloco de configurações para o AudioMathENGINE. Possui um único atributo `lang` cujo valor indica a língua usada pela aplicação. O valor de `lang` adoptado é o mesmo que para a identificação de país num endereço de rede, de acordo com o ISO639-1 e o ISO3166-1-alpha2. Actualmente as aplicações AudioMath apenas suportam a língua Português Europeu – `pt`.

`<options>...</options>`

Indica um bloco de opções para algo.

`<set name="" value="" />`

Define um conjunto de inicializações. Por exemplo, ficheiros de configuração a usar externamente – dicionários.

`<conversiondefaults>...</conversiondefaults>`

Define um bloco de configurações por defeito para os algoritmos de conversão do AudioMathENGINE.

`<conversionsettings>...</conversionsettings>`

Define um bloco de configurações correntes para os algoritmos de conversão do AudioMathENGINE.

`<module name="" usermode="">...</module>`

Define um módulo de conversão existente no AudioMathENGINE. O atributo `name` indica o módulo em causa. Pode conter os valores: `numeral`, `abbreviation`, `acronym` ou `network`. O atributo `usermode` é utilizado para indicar o modo de utilizador por defeito naquele módulo. Estes modos de utilizadores encontram-se implementados no AudioMathENGINE com códigos específicos[70].

[70] Estes códigos serão apresentados nas secções seguintes que descrevem os algoritmos de conversão.

`<conversion name="" usermode="" gender="" quantity="" type="" />`

Define um algoritmo de conversão no AudioMathENGINE. O atributo `name` especifica o algoritmo em causa. Pode conter os valores: `cardinal`, `ordinal`, `fractional`, `decimal`, `percentage`, `roman`, `date` e `ip`. Os atributos `gender` (com valores: `m` ou `f`), `quantity` (com valores: `s` ou `p`) e `type` (com valores: `c` ou `o`) destinam-se a caracterizar os algoritmos de conversão.

```
<akml version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="../schema/AKML.xsd">
  <engine lang="pt">
    <options>
      <set name="acronym" value="akml-engine/pt_acronyms.xml" />
      <set name="abbreviation" value="akml-
engine/pt_abbreviations.xml" />
      <set name="usermodes"  value="akml-gui/usermodes.xml" />
    </options>
    <conversiondefaults>
      <module name="numeral" usermode="0">
        <conversion name="cardinal" usermode="0" gender="m" />
        <conversion name="ordinal" usermode="0" gender="m"
quantity="s" />
        <conversion name="fractional" usermode="0" gender="m" />
        <conversion name="decimal" usermode="0" gender="m" />
        <conversion name="percentage" usermode="0" gender="m" />
        <conversion name="roman" usermode="0" gender="m" type="c" />
        <conversion name="date" usermode="0" />
      </module>
      <module name="abbreviation" usermode="0" />
      <module name="acronym" usermode="0" />
      <module name="network" usermode="0">
        <conversion name="ip" usermode="0" />
      </module>
    </conversiondefaults>
    <conversionsettings>
      <module name="numeral" usermode="0">
        <conversion name="cardinal" usermode="0" gender="m" />
        <conversion name="ordinal" usermode="0" gender="m"
quantity="s" />
        <conversion name="fractional" usermode="0" gender="m" />
        <conversion name="decimal" usermode="1" gender="m" />
        <conversion name="percentage" usermode="0" gender="m" />
        <conversion name="roman" usermode="0" gender="m" type="o" />
        <conversion name="date" usermode="0" />
      </module>
      <module name="abbreviation" usermode="0" />
      <module name="acronym" usermode="0" />
      <module name="network" usermode="0">
        <conversion name="ip" usermode="1" />
      </module>
    </conversionsettings>
  </engine>
</akml>
```

Figura 7.7 - Exemplo de AKML de Configuração para o AudioMathENGINE.

7.2.2 Etiquetas de marcação para Dicionário

A marcas de dicionário do AKML foram definidas de modo genérico, possibilitando assim o seu uso para diferentes tipos de dicionários.

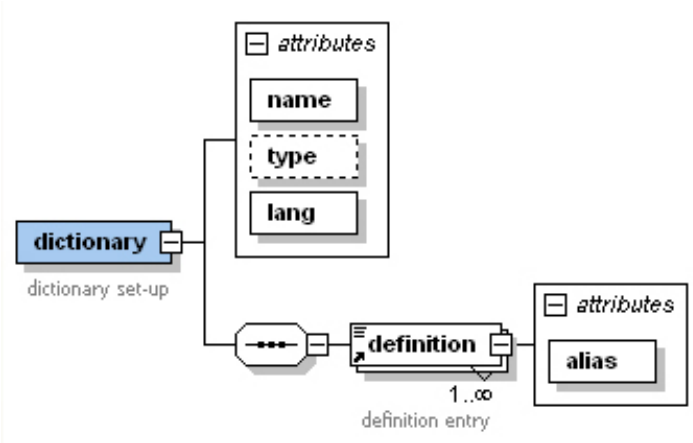


Figura 7.8 - Estrutura AKML para Dicionários do AudioMathENGINE.

Segue-se uma descrição dos elementos e atributos utilizados:

`<dictionary name="" lang="">...</dictionary>`

Define um bloco do tipo dicionário. O atributo `name` tem como objectivo identificar o tipo de dicionário que é definido e pode ter como valor: `acronym` ou `abbreviation`. O atributo `type` indica uma sub classificação do dicionário especificado. Por exemplo, um dicionário de abreviações faz com que `type` possa ter como valores: `social` (abreviações sociais), `money` (abreviações monetárias), `physics` (unidades físicas), `chemistry` (símbolos químicos) ou `others` (outros tipos de abreviações). O atributo `lang` permite identificar a língua na qual o dicionário se encontra definido.

`<definition alias="">...</definition>`

Definição de um determinado termo do dicionário. O atributo `alias` é usado para indicar a leitura por extenso que o AudioMathENGINE utiliza no algoritmo de conversão.

```
<akml version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="../schema/AKML.xsd">
      <dictionary name="abbreviation" type="social" lang="pt">
            <definition alias="Engenheiro">Eng.</definition>
            <definition alias="Engenheira">Eng.ª</definition>
      </dictionary>
      <dictionary name="abbreviation" type="money" lang="pt">
            <definition alias="euro">€</definition>
            <definition alias="libra">£</definition>
      </dictionary>
      <dictionary name="abbreviation" type="physics" lang="pt">
            <definition alias="quilate">ql</definition>
            <definition alias="ano-luz">a.l.</definition>
      </dictionary>
      <dictionary name="abbreviation" type="chemistry" lang="pt">
            <definition alias="Actínio">Ac</definition>
            <definition alias="Alumínio">Al</definition>
      </dictionary>
      <dictionary name="abbreviation" type="others" lang="pt">
            <definition alias="antes de Cristo">a.c.</definition>
            <definition alias="depois de Cristo">d.c.</definition>
      </dictionary>
</akml>
```

Figura 7.9 - Exemplo de Dicionário de Abreviações em AKML.

7.2.3 Etiquetas de marcação para Conversão e Resultados

O resultado da conversão de um documento, via AudioMathENGINE, é outro documento marcado com AKML. Neste documento de resultado é possível encontrar informação estatística sobre a conversão, assim como uma separação lógica entre as expressões matemáticas e o texto normal; e até informação sobre navegação.

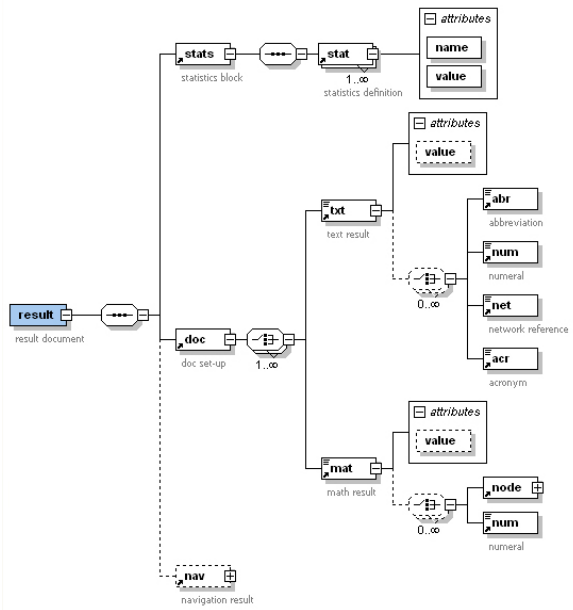


Figura 7.10 - Estrutura AKML para apresentação de Resultados.

Segue-se uma descrição dos elementos e atributos utilizados:

`<stats>...</stats>`

Define um bloco de estatísticas sobre a conversão realizada pelo AudioMathENGINE.

`<stat name="" value="" />`

Indica um resultado de uma estatística. O atributo `name` indica o nome da estatística que foi recolhida. Na versão actual do AudioMathENGINE, o atributo `name` apenas pode conter os seguintes valores: `numeral` (número de numerais identificados e convertidos), `abbreviation` (número de abreviações identificadas e convertidas), `acronym` (número de acrónimos identificados e convertidos), `network` (número de referências de rede identificadas e convertidas), `math` (número de expressões matemáticas identificadas e convertidas) e `convtime` (tempo de conversão dado em segundos e com precisão de três casas decimais).

`<doc>...</doc>`

Identifica um bloco de resultado final, sem pontos de navegação.

`<txt value="">...</txt>`

Identifica um bloco de texto normal convertido (sem expressões matemáticas). O atributo opcional `value` é apenas utilizado para efeitos de navegação e serve como texto introdutório a um nó de navegação.

`<abr>...</abr>,<acr>...</acr>,<num>...</num>,<net>...</net>`

Identificam respectivamente: abreviações, acrónimos, numerais e referências de rede no texto.

`<mat value="">...</mat>`

Identifica um bloco de expressões matemáticas. O atributo `value` tem o mesmo propósito que no elemento `<txt>...</txt>`. Dentro do elemento `mat` podem surgir elementos `node` (usados na navegação) e elementos `num` (numerais identificados na expressão matemática).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<akml version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="schema/AKML.xsd">
      <result>
            <stats>
                  <stat name="numeral" value="2"/>
                  <stat name="abbreviation" value="3"/>
                  <stat name="acronym" value="0"/>
                  <stat name="network" value="0"/>
                  <stat name="math" value="1"/>
                  <stat name="convtime" value="0.121"/>
            </stats>
            <doc>
                  <txt>O <abr>professor</abr> Helder e a
<abr>senhora</abr> Ana encontraram a <abr>página</abr> que mostrava a
expressão: </txt><mat> chis mais <num>cinco</num> igual a
<num>zero</num> </mat>
            </doc>
            <nav>
                  <txt value="Texto:">O professor Helder e a senhora
Ana encontraram a página que mostrava a expressão: </txt><mat
value="Fórmula:"><node value=" chis mais cinco igual a zero "><node
value=" chis "></node><node value=" mais "></node><node value=" cinco
"></node><node value=" igual a "></node><node value=" zero
"></node></node></mat>
            </nav>
      </result>
</akml>
```

Figura 7.11 - Exemplo do resultado de uma conversão em AKML.

7.2.4 Etiquetas de marcação para Navegação

O conjunto de etiquetas de marcação para navegação é um subconjunto das etiquetas de resultado apresentadas na secção anterior.

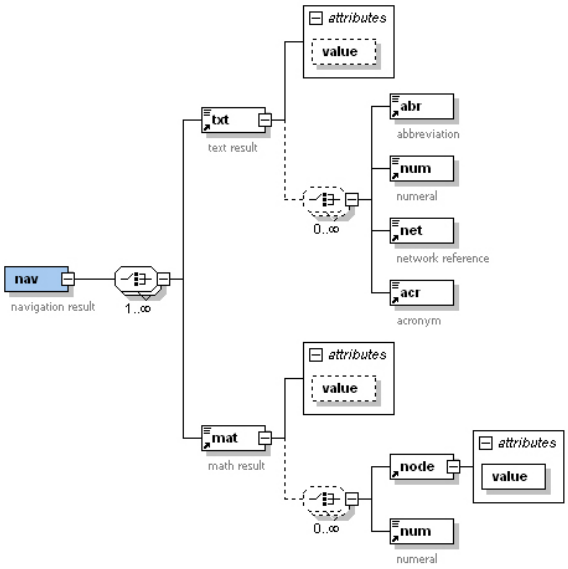


Figura 7.12 - Estrutura AKML para Navegação.

Segue-se uma descrição dos elementos e atributos utilizados:

`<nav>...</nav>`

Identifica um bloco de resultado final, preparado para navegação. Dentro deste bloco pode-se encontrar os elementos `txt` e `mat` que possuem as mesmas definições atribuídas na secção anterior.

`<node value=”>...</node>`

Indica um nó de navegação. É a hierarquia de nós que constrói a árvore de navegação. O atributo `value` contém o texto que cada nó tem atribuído quando se navega na árvore.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<akml version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="schema/AKML.xsd">
      <result>
            <stats>
                  <stat name="numeral" value="2"/>
                  <stat name="abbreviation" value="3"/>
                  <stat name="acronym" value="0"/>
                  <stat name="network" value="0"/>
                  <stat name="math" value="1"/>
                  <stat name="convtime" value="0.121"/>
            </stats>
            <doc>
               <txt>O <abr>professor</abr> Helder e a
<abr>senhora</abr> Ana encontraram a <abr>página</abr> que mostrava a
expressão: </txt>
                <mat> chis mais <num>cinco</num> igual a
<num>zero</num> </mat>
            </doc>
            <nav>
                  <txt value="Texto:">O professor Helder e a senhora
Ana encontraram a página que mostrava a expressão: </txt>
                  <mat value="Fórmula:">
                      <node value=" chis mais cinco igual a zero ">
                         <node value=" chis "></node>
                         <node value=" mais "></node>
                         <node value=" cinco "></node>
                         <node value=" igual a "></node>
                         <node value=" zero "></node>
                      </node>
                  </mat>
            </nav>
      </result>
</akml>
```

Figura 7.13 - Exemplo de uma navegação em AKML.

7.3 Algoritmos de Análise, Interpretação e Conversão

Como se pode observar da Figura 7.14, quando um documento é fornecido ao AudioMathENGINE, este realiza uma série de operações de pré-tratamento do texto, incluindo suporte para páginas on-line e Unicode. Após este tratamento, o documento é separado em blocos lógicos de texto normal ou expressões matemáticas. Cada um destes blocos é processado separadamente, onde são aplicados diversos algoritmos de análise, interpretação e conversão. À junção dos blocos processados inclui-se pontos de navegação, e constrói-se o resultado como um documento AKML.

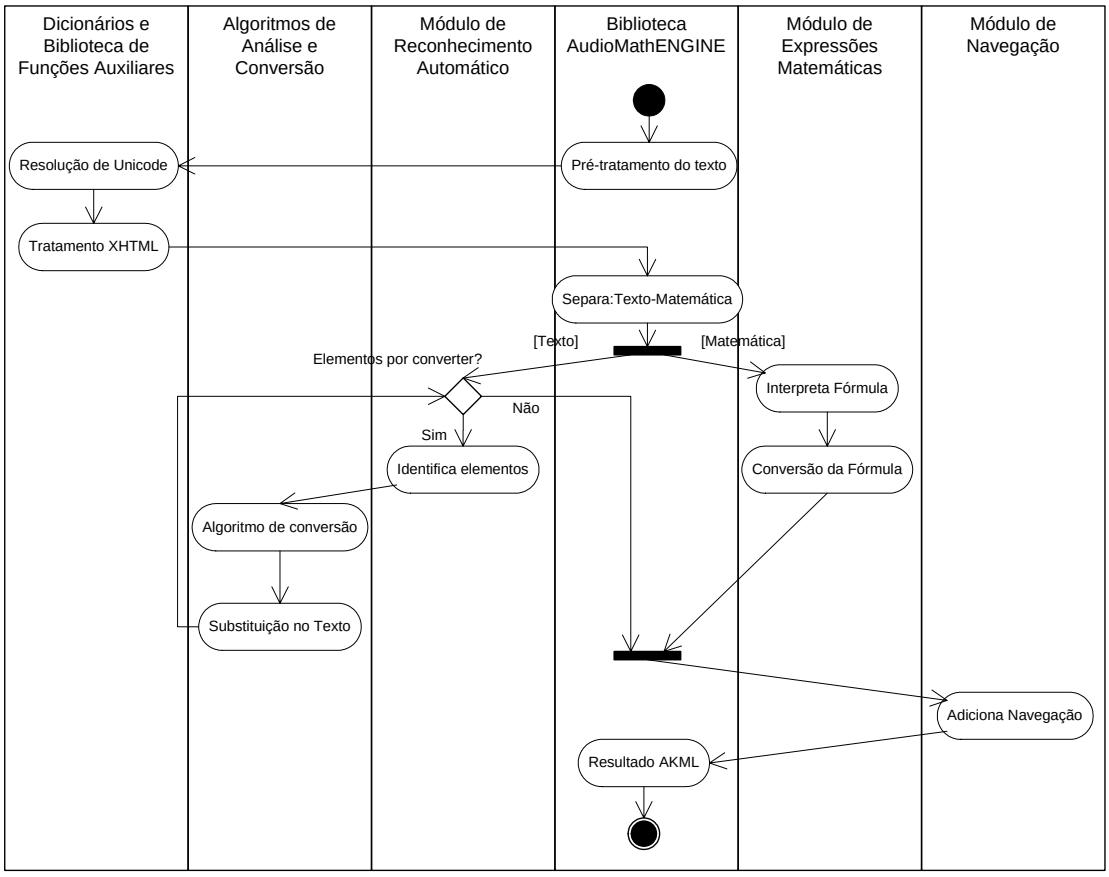


Figura 7.14 - Interacção (simplificada) dos módulos no AudioMathENGINE.

Nesta secção são apresentados cada um dos módulos que contêm algoritmos de análise, interpretação e conversão do AudioMathENGINE, com excepção dos algoritmos de reconhecimento automático e de conversão de expressões matemáticas que serão analisados em secções separadas dada a sua relevância no AudioMathENGINE.

7.3.1 Módulo de Numerais

Embora seja comum a confusão entre um numeral e um número, existe uma diferença ténue entre ambas as definições:

- *numeral* – "Referente a número, designativo de número. Palavra que designa o número, a ordem numa série ou a proporcionalidade numérica." (119)
- *número* – "Expressão de quantidade que permite enumerar e expressar grandezas. Unidade." (119)

O módulo de análise, interpretação e conversão de numerais é implementado por `numerais.pm`. Esta biblioteca suporta quinze classes diferentes de numerais, sendo elas: cardinais, ordinais, fraccionários, decimais, romanos, telefones, datas, horas, escalas, apostas, resultados desportivos, percentagens, valores monetários, números no formato de engenharia e potências.

Tabela 7.2 - Lista de funções de numerais.pm .

Funções exportadas pelo módulo:
&c_cardinais(numero, genero, usermode);
&c_ordinais(numero, genero, quantidade, usermode);
&c_fraccionarios(numero, genero, usermode);
&c_decimais(numero, genero, usermode);
&c_romanos(numero, genero, tipo, usermode);
&c_telefones(numero, usermode);
&c_datas(data, usermode);
&c_horas(horas, usermode);
&c_escalas(escala_ou_aposta, usermode);
&c_resultados_desportivos(resultado, usermode);
&c_percentagens(numero, genero, usermode);
&c_monetarios(numero, usermode);
&c_engenharia(numero, usermode);
&c_potencias(numero, usermode);
Funções internas do módulo:
Usadas para **conversão de cardinais**:
&c_dezenas(numero, genero); &c_centenas(numero, genero); &c_milhares4(numero, genero); &c_milhares5(numero, genero); &c_milhares6(numero, genero); &c_milhoes7(numero, genero); &c_milhoes8(numero, genero); &c_milhoes9(numero, genero); &c_milhoes10(numero, genero); &c_milhoes11(numero, genero); &c_milhoes12(numero, genero); &c_bignum(numero, genero);
Usadas para **conversão de ordinais**:
&c_ordinais2(numero, genero); &c_ordinais3(numero, genero); &c_ordinais4(numero, genero);
Usadas para **conversão de romanos**:
&isroman(numero); &arabic(numero); &Roman; &roman;
Usadas para **conversão de datas**:
&testa_data(data);

Conversão de cardinais

O algoritmo de conversão de cardinais implementado possui as seguintes características:

- Converte números até 10^{29} (30 algarismos). Números com mais de 12 algarismos são arredondados em biliões, triliões e quatriliões usando o termo "*cerca de...*". Acima dos 30 algarismos o número é soletrado.
- Permite o uso dos sinais "+", "-" ou combinação dos dois, precedendo o número a converter.
- É suportado o uso de pontos para separar as ordens de grandeza. Por exemplo: 1.234 é convertido em "mil duzentos e trinta e quatro".
- Possui como parâmetros de entrada auxiliares: `género` e `modo de utilizador`.
- O género pode ser: `masculino` (`m`) ou `feminino` (`f`). Por defeito, o algoritmo assume o masculino (m).
- O modo de utilizador pode ser: `0 (Leitura por extenso)` ou `1 (Leitura soletrada)`. Por defeito, o algoritmo assume o modo `0`.
- Se a entrada não for um número cardinal válido, então a saída será a leitura soletrada da entrada.

Este algoritmo faz uso de regras gramaticais (Tabela 7.3) para a concatenação de conversões parciais.

Tabela 7.3 – Números cardinais em português europeu (120).

Nº	Por extenso	Nº	Por extenso	Nº	Por extenso
0	zero	**70**	setenta	**xx00**	X mil e
1	um	**7x**	setenta e	**x1xx**	X mil cento e
2	dois	**80**	oitenta	**x2xx**	X mil duzentos e
3	três	**8x**	oitenta e	**1000000**	um milhão
4	quatro	**90**	noventa	**1000xxx**	um milhão e
5	cinco	**9x**	noventa e	**10xx000**	um milhão e
6	seis	**100**	cem	**1x00000**	um milhão e
7	sete	**1xx**	cento e	**1xxxxxx**	um milhão
8	oito	**200**	duzentos	**x000000**	X milhões
9	nove	**2xx**	duzentos e	**x000xxx**	X milhões e
10	dez	**300**	trezentos	**x0xx000**	X milhões e
11	onze	**3xx**	trezentos e	**xx00000**	X milhões e
12	doze	**400**	quatrocentos	**xxxxxxx**	X milhões
13	treze	**4xx**	quatrocentos e	**1000000000000**	um bilião
14	catorze	**500**	quinhentos	**xxxxxxxxxxxxx**	X biliões

15	quinze	**5xx**	quinhentos e	$\mathbf{10^{18}}$	um trilião
16	dezasseis	**600**	seiscentos	$\mathbf{10^{24}}$	um quatrilião
17	dezassete	**6xx**	seiscentos e		
18	dezoito	**700**	setecentos		
19	dezanove	**7xx**	setecentos e		
20	vinte	**800**	oitocentos		
2x	vinte e	**8xx**	oitocentos e		
25	vinte e cinco	**900**	novecentos		
30	trinta	**9xx**	novecentos e		
3x	trinta e	**1000**	mil		
40	quarenta	**10xx**	mil e		
4x	quarenta e	**1x00**	mil e		
50	cinquenta	**11xx**	mil cento e		
5x	cinquenta e	**12xx**	mil duzentos e		
60	sessenta	**x000**	X mil		
6x	sessenta e	**x0xx**	X mil e		

Tabela 7.4 - Exemplos de conversão de cardinais.

Sintaxe	**Conversão**
&c_cardinais(número, género, modo de utilizador);	
&c_cardinais(“122”,”m”,0);	cento e vinte e dois
&c_cardinais(“122”,”f”,0);	cento e vinte e duas
&c_cardinais(“+1.345”,”m”,0);	mais mil trezentos e quarenta e cinco
&c_cardinais(“-23”,”m”,0);	menos vinte e três
&c_cardinais(“-23”,”m”,1);	menos dois três
&c_cardinais(“+-1000”,”m”,1);	mais menos um zero zero zero
&c_cardinais(“63923659254945395”,”m”,0);	cerca de sessenta e três mil novecentos e vinte e três biliões

Conversão de ordinais

O algoritmo de conversão de ordinais implementado possui as seguintes características:

- Converte números até 1999[71], acima disso soletra.
- Suporta o uso dos símbolos: “º”, “ª”, “ºs” e “ªs”.
- Suporta números romanos para serem lidos como ordinais. Por exemplo: “D. Afonso II”, lê-se “Dom Afonso Segundo” e não “Dom Afonso Dois”.
- Possui como parâmetros de entrada auxiliares: `género`, `quantidade` e `modo de utilizador`.

[71] Em português europeu só existe forma escrita por extenso de ordinais até ao número 1999.

- O género pode ser: `masculino` (`m`) ou `feminino` (`f`). Por defeito, o algoritmo assume o masculino (`m`). Se o número vier acompanhado de um dos símbolos acima mencionados, o género do símbolo prevalece em relação ao género indicado na função.
- A quantidade pode ser: `singular` (`s`) ou `plural` (`p`). Por defeito, o algoritmo assume o singular (`s`).
- Só existe um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for um número ordinal válido, então a saída será a leitura soletrada da entrada.

Este algoritmo faz uso de regras gramaticais (Tabela 7.5) para a concatenação de conversões.

Tabela 7.5 - Números ordinais em português (120).

Nº	Por extenso	Nº	Por extenso
1	primeiro	**80**	octogésimo
2	segundo	**90**	nonagésimo
3	terceiro	**100**	centésimo
4	quarto	**200**	ducentésimo
5	quinto	**300**	tricentésimo
6	sexto	**400**	quadringentésimo
7	sétimo	**500**	quingentésimo
8	oitavo	**600**	sexcentésimo
9	nono	**700**	septingentésimo
10	décimo	**800**	octingentésimo
20	vigésimo	**900**	nongentésimo
30	trigésimo	**1000**	milésimo
40	quadragésimo		
50	quinquagésimo		
60	sexagésimo		
70	septuagésimo		

Tabela 7.6 - Exemplos de conversão de ordinais.

Sintaxe	Conversão
&c_ordinais(número, género, quantidade, modo de utilizador);	
&c_ordinais(“12”,”m”,”s”,0);	décimo segundo
&c_ordinais(“12”,”f”,”s”,0);	décima segunda
&c_ordinais(“2ª”,”m”,”p”,0);	segundas
&c_ordinais(“1023”,”m”,”s”,0);	milésimo vigésimo terceiro
&c_ordinais(“5º”,”f”,”p”,0);	quintos
&c_ordinais(“2013”,”m”,”s”,0);	dois zero um três

Conversão de fraccionários ou racionais

O algoritmo de conversão de fraccionários implementado possui as seguintes características:

- É limitado apenas pelas propriedades dos cardinais e ordinais.
- Permite o uso dos sinais "+", "-" ou combinação dos dois, precedendo o número a converter.
- Se a entrada for `X/1`, então esta é soletrada.
- Se a entrada for `X/X`, lê "chis sobre chis" e não "um".
- Possui como parâmetros de entrada auxiliares: `género` e `modo de utilizador`.
- O género pode ser: `masculino` (`m`) ou `feminino` (`f`). Por defeito, o algoritmo assume o masculino (`m`).
- Só existe um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for um número fraccionário válido, então a saída será a leitura soletrada da entrada. No entanto, realiza-se um teste para verificar se a entrada é um cardinal. Em caso afirmativo, converte o cardinal.

Este algoritmo faz uso de regras gramaticais (Tabela 7.5) para a concatenação de conversões.

Tabela 7.7 - Números fraccionários em português europeu.

Nº	Por extenso	Nº	Por extenso
1/2	um meio ; metade		
1/3	um terço ; uma terça-parte	**2/3**	dois terços ; duas terças-partes
1/4	um quarto ; uma quarta-parte	**2/4**	dois quartos ; duas quartas-partes
1/5	um quinto ; uma quinta-parte	**2/5**	dois quintos ; duas quintas-partes
1/6	um sexto ; uma sexta-parte	**2/6**	dois sextos ; duas sextas-partes
1/7	um sétimo ; uma sétima-parte	**2/7**	dois sétimos ; duas sétimas-partes
1/8	um oitavo ; uma oitava-parte	**2/8**	dois oitavos ; duas oitavas-partes
1/9	um nono ; uma nona-parte	**2/9**	dois nonos ; duas nonas-partes
1/10	um décimo ; uma décima-parte	**2/10**	dois décimos ; duas décimas-partes
1/11	um onze-avos ; uma décima primeira-parte	**2/11**	dois onze-avos ; duas décimas primeiras partes
1/12	um doze-avos ; uma décima segunda-parte	**2/12**	dois doze-avos ; duas décimas segundas partes

Tabela 7.8 – Exemplos de conversão de fraccionários.

Sintaxe	Conversão
&c_fraccionarios(número, género, modo de utilizador);	
&c_fraccionarios(“1/2”,”m”,0);	um meio
&c_fraccionarios(“1/2”,”f”,0);	metade
&c_fraccionarios(“23/23”,”m”,0);	vinte e três sobre vinte e três
&c_fraccionarios(“3/12”,”m”,0);	três, doze-avos
&c_fraccionarios(“1/3”,”m”,0);	um terço
&c_fraccionarios(“1/3”,”f”,0);	terça-parte

Conversão de decimais ou fraccionários não racionais

O algoritmo de conversão de decimais implementado possui as seguintes características:

- É limitado apenas pelas propriedades dos cardinais.
- Permite o uso dos sinais “+”, “-“ ou combinação dos dois, precedendo o número a converter.
- É suportado apenas o uso da vírgula como separador entre parte inteira e decimal.[72]
- Possui como parâmetros de entrada auxiliares: `género` e `modo de utilizador`.
- O género pode ser: `masculino` (`m`) ou `feminino` (`f`). Por defeito, o algoritmo assume o masculino (`m`).
- Os modos de utilizador podem ser: `0 (Leitura por extenso da parte inteira e decimal)` ou `1 (Leitura por extenso da parte inteira e soletrada da parte decimal)`.
- Se a entrada não for um número decimal válido, então a saída será a leitura soletrada da entrada. No entanto, realiza-se um teste para verificar se a entrada é um cardinal. Em caso afirmativo, converte o cardinal.

Tabela 7.9 - Exemplos de conversão de decimais.

Sintaxe	Conversão
&c_decimais(número, género, modo de utilizador);	
&c_decimais(“1.2”,”m”,0);	um ponto dois
&c_decimais(“+1,2”,”f”,0);	mais uma ponto duas
&c_decimais(“23,45”,”m”,1);	vinte e três vírgula quatro cinco
&c_decimais(“1.200,56”,”m”,0);	mil e duzentos vírgula cinquenta e seis

[72] Em português, a vírgula é o separador usado nos números decimais. O uso de ponto é uma prática de países como por exemplo, a Inglaterra.

Conversão de romanos

O algoritmo de conversão de romanos implementado possui as seguintes características:

- Possui como parâmetros de entrada auxiliares: `género`, `tipo` e `modo de utilizador`.
- O género pode ser: `masculino` (`m`) ou `feminino` (`f`). Por defeito, o algoritmo assume o masculino (`m`).
- O tipo pode ser: `cardinal` (`c`) ou `ordinal` (`o`). Por defeito, o algoritmo assume o `cardinal` (`c`). No entanto, o algoritmo suporta o uso de símbolos de ordinais. Neste caso, o género dos símbolos tem precedência sobre o género dado na função.
- Os modos de utilizador podem ser: `0 (Leitura por extenso)` ou `1 (Leitura soletrada)`.
- Se a entrada não for um número romano válido, então a saída será a leitura soletrada da entrada.

Tabela 7.10 - Exemplos de conversão de números romanos.

Sintaxe	**Conversão**
&c_romanos(número, género, tipo, modo de utilizador);	
&c_romanos(“II”,”m”,”c”,0);	dois
&c_romanos(“II”,”f”,”c”,0);	duas
&c_romanos(“VIII”,”m”,”o”,0);	oitavo
&c_romanos(“XVIII”,”m”,”c”,1);	chis vê i i i

Conversão de números telefónicos

O algoritmo de conversão de telefones implementado possui as seguintes características:

- Suporta os símbolos “-”, “/”, “()” no meio do número, e o símbolo “+” no início.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só existe um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for um número de telefone válido, então a saída será a leitura soletrada da entrada.

Tabela 7.11 - Exemplos de conversão de números de telefone.

Sintaxe	**Conversão**
&c_telefones(número, modo de utilizador);	
&c_telefones(“+351 229271569”,0);	mais três cinco um, dois dois nove dois sete um cinco seis nove
&c_telefones(“229271569/8”,0);	dois dois nove dois sete um cinco seis nove barra oito

Conversão de datas

O algoritmo de conversão de datas implementado possui as seguintes características:

- Suporta diversos formatos de datas: `dd/mm/aa`, `dd/mm/aaaa`, `aa/mm/dd` ou `aaaa/mm/dd`. Exemplos: “21-09-2005”, “2005/09/21” ou “21.Set.2005” .
- Suporta os separadores: “.”, “/” ou “-”.
- Utiliza um algoritmo que testa uma data, verificando se esta é possível. Devido a este sistema, só são aceites datas entre 1/1/1600 e 31/12/9999. Os anos bissextos também são considerados.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Os modos de utilizador podem ser: `0 (Leitura por extenso e conversão do mês)` ou `1 (Leitura por extenso sem converter o mês)`.
- Se a entrada não for uma data válida, então a saída será a leitura soletrada da entrada.

Tabela 7.12 - Exemplos de conversão de datas.

Sintaxe	**Conversão**
&c_datas(data, modo de utilizador);	
&c_datas(“30/02/2003”,0);	três zero barra zero dois barra dois zero zero três
&c_datas(“21.Janeiro.1999”,0);	vinte e um de Janeiro de mil novecentos e noventa e nove
&c_datas(“1977-05-10”,1);	dez do cinco de mil novecentos e setenta e sete
&c_datas(“05-01-20”,0);	cinco do um do vinte

Conversão de horas

O algoritmo de conversão de horas implementado possui as seguintes características:

- Suporta os formatos de 12 horas e 24 horas.
- Suporta os símbolos: `xHxM`, `x:x` e `x am/pm` .
- Suporta diversos formatos de horas, com diversas combinações entre os símbolos enunciados acima. Por exemplo: “23h”, “23:45”, “23H05”, “3 am” ou “23H05M”.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura por extenso)`.

- Se a entrada não for uma hora válida, então a saída será a leitura soletrada da entrada. No entanto, realiza-se um teste para verificar se a entrada é um cardinal. Em caso afirmativo, converte o cardinal.

Tabela 7.13 - Exemplos de conversão de horas.

Sintaxe	Conversão
&c_horas(hora, modo de utilizador);	
&c_horas("22h",0);	vinte e duas horas
&c_horas("15h30",0);	quinze horas e trinta minutos
&c_horas("3:45",0);	três horas e quarenta e cinco minutos
&c_horas("3 am",0);	três à éme

Conversão de escalas e apostas

O algoritmo de conversão de escalas e apostas implementado possui as seguintes características:

- Suporta o formato: `x:x`.
- É limitado apenas pelas propriedades dos cardinais.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for uma escala/aposta válida, então a saída será a leitura soletrada da entrada.

Tabela 7.14 - Exemplo de conversão de escalas ou apostas.

Sintaxe	Conversão
&c_escalas(escala_aposta, modo de utilizador);	
&c_escalas("3:2",0);	três para dois
&c_escalas("100:1",0);	cem para um

Conversão de resultados desportivos

O algoritmo de conversão de resultados desportivos implementado possui as seguintes características:

- Suporta o formato: `x:x`.
- É limitado apenas pelas propriedades dos cardinais.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for um resultado desportivo válido, então a saída será a leitura soletrada da entrada.

Tabela 7.15 - Exemplos de conversão de resultados desportivos.

Sintaxe	Conversão
&c_resultados_desportivos(resultado, modo de utilizador);	
&c_escalas(“3:2”,0);	três, dois
&c_escalas(“1:0”,0);	um, zero

Conversão de percentagens

O algoritmo de conversão de percentagens implementado possui as seguintes características:

- É limitado apenas pelas propriedades dos cardinais e dos decimais.
- Suporta o uso do símbolo: %.
- Possui como parâmetros de entrada auxiliares: `género` e `modo de utilizador`.
- O género pode ser: `masculino` (`m`) ou `feminino` (`f`). Por defeito, o algoritmo assume o masculino (`m`).
- Os modos de utilizador podem ser: `0 (Leitura por extenso)` ou `1 (Leitura soletrada da parte decimal)`.
- Se a entrada não for uma percentagem válida, então a saída será a leitura soletrada da entrada. No entanto, realiza-se um teste para verificar se a entrada é um cardinal. Em caso afirmativo, converte o cardinal.

Tabela 7.16 - Exemplos de conversão de percentagens.

Sintaxe	Conversão
&c_percentagens(número, género, modo de utilizador);	
&c_percentagens(“1.22%”,”m”,0);	um ponto vinte e dois por cento
&c_percentagens(“-0,45%”,”m”,1);	menos zero vírgula quatro cinco por cento

Conversão de valores monetários

O algoritmo de conversão de valores monetários implementado possui as seguintes características:

- É limitado apenas pelas propriedades dos cardinais e dos decimais.
- Apenas suporta *euros*, *escudos*, *libras* e *dólares*.

- Suporta singular e plural nas unidades monetárias[73].
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for um valor monetário válido, então a saída será a leitura soletrada da entrada. No entanto, realiza-se um teste para verificar se a entrada é um cardinal. Em caso afirmativo, converte o cardinal.

Tabela 7.17 - Exemplos de conversão de quantidades monetárias.

Sintaxe	**Conversão**
&c_monetarios(valor, modo de utilizador);	
&c_monetarios("3€",0);	três euros
&c_monetarios("2$00",0);	dois escudos
&c_monetarios("£4.45",0);	quatro ponto quarenta e cinco libras
&c_monetarios("$100,56",0);	cem vírgula cinquenta e seis dólares

Conversão de números no formato de engenharia

O algoritmo de conversão de números no formato de engenharia implementado possui as seguintes características:

- Formato: `+/-xE+/-y` .
- É limitado apenas pelas propriedades dos cardinais e dos decimais.
- Suporta o uso dos símbolos: "+" e "-".
- Suporta o símbolo: "E" ou "e" como "*vezes dez elevado a...*"
- Apenas possui como parâmetro de entrada auxiliar: modo de utilizador.
- Só tem um modo de utilizador: `0` (`Leitura por extenso`).
- Se a entrada não for um número no formato de engenharia válido, então a saída será a leitura soletrada da entrada. No entanto, ele realiza um teste para verificar se a entrada é um cardinal. Em caso afirmativo, ele converte o cardinal.

Tabela 7.18 - Exemplos de conversão no formato de engenharia.

Sintaxe	**Conversão**
&c_engenharia(número, modo de utilizador);	
&c_engenharia("2e3",0);	dois vezes dez elevado a três
&c_engenharia("-3,5E5",0);	menos três vírgula cinco vezes dez elevado a cinco

[73] Em português correcto, não se usa o plural nas unidades. Por exemplo: 2€ lê-se `dois euro`. No entanto, por razões de maior inteligibilidade e de habituação ao discurso oral, foi implementado o plural neste algoritmo. Por esse motivo, o exemplo anterior seria lido: `dois euros`.

Conversão de potências

O algoritmo de conversão de potências implementado possui as seguintes características:

- Formato: `+/-x^+/-y .`
- É limitado apenas pelas propriedades dos cardinais e dos decimais.
- Suporta o uso dos símbolos: "`+`" e "`-`".
- Suporta o símbolo: "`^`" como designação de "*elevado a...*"
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for uma potência válida, então a saída será a leitura soletrada da entrada. No entanto, realiza-se um teste para verificar se a entrada é um cardinal. Em caso afirmativo, converte o cardinal.

Tabela 7.19 - Exemplo de conversão de potências.

Sintaxe	**Conversão**
&c_potencias(número, modo de utilizador);	
&c_potencias("+3.5^2",0);	mais três ponto cinco elevado a dois
&c_potencias("3^34",0);	três elevado a trinta e quatro
&c_potencias("-1.2^3,67",0);	menos um ponto dois elevado a três vírgula sessenta e sete

7.3.2 Módulo de Acrónimos

Embora seja comum a confusão entre sigla e acrónimo, existe uma diferença bastante distinta entre ambas as definições:

- *sigla* – "(Gramática) expressão formada pelas letras iniciais de diversas palavras, sendo estas letras geralmente pronunciadas uma a uma e não com articulação silábica." (119)
- *acrónimo* – "(Gramática) palavra formada pelas letras ou sílabas iniciais de várias outras palavras, e que se pronuncia sílaba a sílaba e não letra a letra." (119)

No AudioMathENGINE não há distinção. O AKML de dicionário permite um uso genérico. Qualquer sigla ou acrónimo que se encontre definida será convertida no valor contido no atributo `alias`.

No entanto a conversão de siglas e acrónimos possui alguns desafios:

- A existência de siglas idênticas que denominam conceitos diferentes. Por exemplo: ACP pode significar "Automóvel Clube de Portugal", mas também pode significar "Associação de Comerciantes do Porto".
- A existência de siglas idênticas que denominam conceitos diferentes, em línguas diferentes. Por exemplo: AM pode significar "Assembleia Municipal", mas também pode significar "*Amplitude Modulation*".
- A decisão de leitura de uma sigla entre: fazer articulação silábica, ler letra a letra ou substituir por um conjunto de palavras.

O processo de desambiguação poderia ser implementado analisando o contexto da frase. No entanto isto não é possível ou suficiente para todos os casos. No AudioMathENGINE optou-se por incluir no dicionário um conjunto de definições que se julgam únicas. Por defeito, o algoritmo de conversão procura a definição neste dicionário e substitui o acrónimo encontrado pela definição. Os acrónimos não definidos são soletrados. No entanto, o utilizador pode alterar este comportamento modificando a configuração de `modo de utilizador` para os acrónimos.

Quanto à detecção de uma sigla ou acrónimo, esta não é relevante no processo descrito anteriormente. No entanto, em (121) desenvolveu-se um algoritmo de detecção de siglas ou acrónimos, baseado numa árvore de decisão em que os parâmetros levados em conta são: o número de letras envolvidas e as diferentes possibilidades de combinação das sequências de letras, no que diz respeito a vogais ou consoantes.

O módulo de conversão de siglas e acrónimos é implementado por `acronimos.pm`.

Tabela 7.20 - Lista de funções de acronimos.pm .

Funções exportadas pelo módulo:
`&c_acron(sigla, usermode);`
`&init_acr();`
Estrutura de dados exportada pelo módulo:
`%acr;`

Existe uma fase de inicialização antecedente ao algoritmo de análise, interpretação e conversão de acrónimos. Nesta fase, a função `init_acr` lê o dicionário AKML de acrónimos e preenche uma estrutura de dados interna do AudioMathENGINE. Este passo é realizado apenas uma vez, durante a inicialização do componente.

A função `c_acron` é uma função de pesquisa que procura um determinado acrónimo na estrutura interna do AudioMathENGINE.

O algoritmo de conversão de siglas e acrónimos implementado possui as seguintes características:

- A estrutura interna onde são guardadas as definições de acrónimos é desenvolvida sob uma `Hash`.
- Possui os seguintes modos de utilizador: `0 (Leitura soletrada)` e `1 (Leitura por extenso)`.
- Se a entrada não for uma sigla ou acrónimo válido, ou se não existir conversão para a entrada, então a saída será a leitura soletrada da entrada.
- Neste momento não é suportado o módulo de distinção entre sigla e acrónimo.

Considerando que o dicionário de acrónimos da Figura 7.15 foi lido pelo AudioMathENGINE, a Tabela 7.21 apresenta exemplos de possíveis conversões.

```
<akml    version="1.0"    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="../schema/AKML.xsd">
   <dictionary name="acronym" lang="pt">
      <definition  alias="Faculdade  de  Engenharia  Universidade  do
Porto">FEUP</definition>
      <definition alias=" Faculdade de Ciências Universidade do Porto
">FCUP</definition>
   </dictionary>
</akml>
```

Figura 7.15 - Exemplo de Dicionário de Acrónimos em AKML.

Tabela 7.21 - Exemplos de conversão de acrónimos.

Sintaxe	**Conversão**
&c_acron(sigla, modo de utilizador);	
&c_acron(“FEUP”, 1);	Faculdade de Engenharia Universidade do Porto
&c_acron(“FEUP” ,0);	éfe é u pê
&c_acron(“OVNI” ,1);	ó vê éne í
&c_acron(“OVNI” ,0);	ó vê éne í

7.3.3 Módulo de Abreviações

Entende-se por abreviatura:

- "Em geral, forma encurtada ou contraída de uma palavra. (Gramática) forma encurtada ou contraída de uma palavra, constituída por uma ou mais letras, seguidas de um ponto, e que se pronuncia como se estivesse por extenso." (119)

No AudioMathENGINE suportam-se diversos tipos de abreviações através de um dicionário em AKML baseado em (119). Para cada abreviatura é indicada uma forma por extenso, por exemplo, para a abreviatura "Prof." define-se "professor". No entanto existem certas abreviaturas que podem possuir mais do que uma definição dependendo do contexto, como por exemplo, a abreviatura "lat." pode definir-se como "latim" ou "latitude". Em AKML isto seria representado como: `<definition alias="latim, latitude">lat. </definition>.`

Até à escrita desta dissertação, o algoritmo de conversão de abreviações apenas suportava a conversão através da primeira definição e continha cerca de trezentas abreviaturas no dicionário.

O módulo de conversão de abreviações é implementado em `abreviacoes.pm`. Esta biblioteca suporta cinco classes diferentes de abreviações: abreviaturas sociais (`social`), abreviaturas de unidades físicas (`physics`), abreviaturas de unidades monetárias (`money`), abreviaturas químicas (`chemistry`) e outros tipos de abreviaturas (`others`).

Tabela 7.22 – Lista de funções de abreviacoes.pm .

Funções exportadas pelo módulo:
&c_abrev(abreviatura, tipo, usermode);
&init_abr_sociais();
&init_abr_uni_fis();
&init_abr_uni_mon();
&init_abr_qui();
&init_abr_outros();
Estruturas de dados exportadas pelo módulo:
%abr_sociais;
%abr_uni_fis;
%abr_uni_mon;
%abr_qui;
%abr_outros;

Existe uma fase de inicialização antecedente ao algoritmo de análise, interpretação e conversão de acrónimos. Nesta fase, as funções `init` lêem o dicionário AKML de acrónimos e preenchem diversas estruturas de dados interna do AudioMathENGINE. Este passo é realizado apenas uma vez, durante a inicialização do componente.

A função `c_abrev` tem como finalidade converter as abreviações detectadas onde o tipo de abreviação é identificado pelo parâmetro `tipo` que pode ter um dos seguintes valores:

- SOC – abreviatura social;
- FIS – abreviatura de unidade física;
- MON – abreviatura de unidade monetária;
- QUI – abreviatura de símbolos químicos;
- OUT – abreviatura de outros tipos;
- GEN – abreviatura genérica (procura nos dicionários de abreviaturas `social` e abreviaturas `others`);
- ALL – procura conversões em todos os dicionários de abreviaturas.

O algoritmo de conversão de abreviações implementado possui as seguintes características:

- Suporte de diversos tipos de abreviaturas, já enunciados.
- As estruturas internas onde são guardadas as definições de abreviaturas são desenvolvidas em `Hashes`.
- Os parâmetros de entrada auxiliares são: `tipo` e `modo de utilizador`.
- O `tipo` pode ser: `SOC`, `FIS`, `MON`, `QUI`, `OUT`, `GEN` ou `ALL`.
- Só tem um modo de utilizador: `0 (Leitura por extenso)`.
- Se a entrada não for uma abreviatura válida, então a saída será idêntica à entrada. Optou-se por não soletrar a abreviação uma vez que estas fazem mais vezes sentido quando lidas e não soletradas. Por exemplo, se a abreviação "`Prof.`" não fosse encontrada na base de dados, então seria preferível ler "`prófe`" do que ler "`pê érre ó éfe ponto`".

Tabela 7.23 - Exemplos de conversão de abreviaturas.

Sintaxe	Conversão
&c_abrev(abreviatura, tipo, modo de utilizador);	
&c_abrev("Sr.","SOC",0);	Senhor
&c_abrev("Engº","SOC",0);	Engenheiro
&c_abrev("Cu","QUI",0);	Cobre
&c_abrev("€","MON",0);	Euro
&c_abrev("m","FIS",0);	metro
&c_abrev("pag.","OUT",0);	página

7.3.4 Módulo de Referências de Rede

Os documentos técnicos contêm, cada vez mais, referências a endereços de Internet, endereços de correio electrónico ou mesmo endereços IP[74]. Por esse motivo foi desenvolvido um módulo de conversão dedicado à conversão destes tipos de elementos.

O módulo de conversão de referências de rede é implementado por `referenciasrede.pm`. Esta biblioteca suporta três classes diferentes de referências sendo elas: endereços de páginas on-line (URL/URI)[75], endereços de correio electrónico (e-mail) e endereços de máquinas ligadas na Internet (IP**)**.

Tabela 7.24 - Lista de funções de referenciasrede.pm .

Funções exportadas pelo módulo:
&c_url_uri(url_uri, usermode);
&c_emails(email, usermode);
&c_ips(ip, usermode);

Conversão de endereços de Internet

O algoritmo de conversão de URL/URI implementado possui as seguintes características:

- Possui uma rotina de validação que verifica se a entrada é mesmo um endereço de rede válido ou não. Em caso negativo, a saída será uma leitura soletrada da entrada.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura soletrada)`.

[74] *Internet Protocol*
[75] URL – *Universal Resource Locator*; URI – *Universal Resource Identifier*.

Tabela 7.25 - Exemplos de conversão de endereços de Internet.

Sintaxe	Conversão
&c_url_uri(url_uri, modo de utilizador);	
&c_url_uri(“http://www.fe.up.pt”,0);	agá tê tê pê dois pontos barra barra dabliu dabliu dabliu ponto éfe é ponto u pê ponto pê tê
&c_url_uri(“http://lpf-esi.fe.up.pt”,0);	agá tê tê pê dois pontos barra barra éle pê éfe hífen é ésse i ponto éfe é ponto u pê ponto pê tê

Conversão de endereços de correio electrónico

O algoritmo de conversão de endereços de correio electrónico implementado possui as seguintes características:

- Possui uma rotina de validação que verifica se a entrada é mesmo um endereço de correio electrónico válido ou não. Em caso negativo, a saída será uma leitura soletrada da entrada.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Só tem um modo de utilizador: `0 (Leitura soletrada)`.

Tabela 7.26 - Exemplos de conversão de endereços de correio electrónico.

Sintaxe	Conversão
&c_emails(email, modo de utilizador);	
&c_emails(“hfilipe@fe.up.pt”,0);	agá éfe i éle i pê é arroba éfe é ponto u pê ponto pê tê
&c_emails(“Ana.Mota@fe.up.pt”,0);	á éne á ponto éme ó tê á arroba éfe é ponto u pê ponto pê tê

Conversão de endereços do tipo IP

O algoritmo de conversão de endereços do tipo IP implementado possui as seguintes características:

- Possui uma rotina de validação que verifica se a entrada é mesmo um endereço de rede válido ou não. Em caso negativo, a saída será uma leitura soletrada da entrada.
- Apenas possui como parâmetro de entrada auxiliar: `modo de utilizador`.
- Os modos de utilizador podem ser: `0 (Leitura por extenso entre os pontos)` ou `1 (Leitura soletrada)`.

Tabela 7.27 - Exemplos de conversão de endereços de rede do tipo IP.

Sintaxe	Conversão
&c_ips(ip, modo de utilizador);	
&c_ips(“192.168.106.71”,0);	cento e noventa e dois ponto cento e sessenta e oito ponto cento e seis ponto setenta e um
&c_ips(“255.255.255.0”,1);	dois cinco cinco ponto dois cinco cinco ponto dois cinco cinco ponto zero

7.3.5 Bibliotecas Auxiliares

Os algoritmos de análise, interpretação e conversão anteriormente apresentados necessitam, por vezes, de funções extra que os auxiliem no desempenho das suas actividades. São as denominadas *bibliotecas de suporte* ou *bibliotecas auxiliares*. No AudioMathENGINE existem seis bibliotecas de suporte:

- `tabelas.pm` – conjunto de tabelas para conversões de numerais (por exemplo a definição de unidades ou a definição dos ordinais), assim como conversão dos meses do ano.
- `biblioteca.pm` – conjunto de funções genéricas e bastante reutilizadas, como por exemplo a soletração de uma palavra; leitura e escrita de modos de utilizador, validação de documentos MathML, e filtros de tratamento de texto (limpeza de entidades XML, Unicode e AKML).
- `xhtml.pm` – módulo cujas funções filtram os elementos XHTML de uma página on-line.
- `mathml_dictionary.pm` – módulo, com mais de 4200 entradas, cujas funções convertem todas as entidades XML do MathML em códigos Unicode.
- `mathml_to_unicode.pm` – módulo, com mais de 5200 entradas, que implementa um dicionário[76] de operadores e funções de MathML, assim como um dicionário de Unicode.
- `navegacao.pm` – módulo cujas funções implementam a navegação das fórmulas matemáticas.

Os módulos `mathml_to_unicode.pm`, `mathml_dictionary.pm` e `navegacao.pm` serão abordados com mais detalhe na secção 7.5.

[76] Este dicionário implementa toda a lista de caracteres especiais de MathML publicada na especificação do MathML 2.0, pela W3C.

7.4 Módulo de Reconhecimento Automático

Os módulos descritos anteriormente apenas fornecem funções capazes de converter diversos elementos de um documento, isto é, representam o papel de bibliotecas com rotinas de conversão. Contudo é necessária uma acção de análise e interpretação antes de uma conversão. Esta acção de análise e interpretação, ou de reconhecimento, é implementada pelo módulo de reconhecimento automático do AudioMathENGINE.

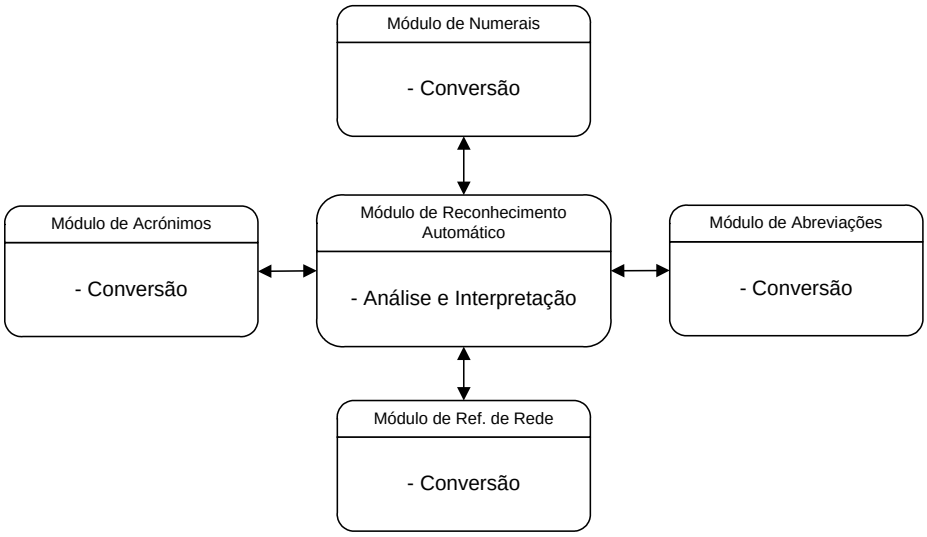


Figura 7.16 - Relação entre os módulos de reconhecimento e conversões.

O módulo de reconhecimento automático é implementado por `autorecon.pm` e suporta o reconhecimento automático de diversos elementos num documento. Este algoritmo é implementado num estilo de arquitectura em camadas (*layers*), onde existem funções de interface para os módulos de conversão de numerais, acrónimos, abreviações e referências de rede. Recorde-se que apenas os blocos de texto normal (sem expressões matemáticas) são processados pelo módulo de reconhecimento automático.

O processo de reconhecimento automático é suportado por um conjunto de expressões regulares que analisam blocos de texto e identificam potenciais candidatos à conversão. Estas expressões regulares encontram-se definidas numa ordem específica de identificação de elementos, necessária, em especial no caso dos numerais, uma vez que se aplica uma filosofia de converter os elementos partindo dos casos mais complexos para os casos mais simples.

Por exemplo, no texto “A falha sísmica aumenta 2,5 mm por ano. No mundo existem mais 2 casos assim. A hipótese de terramoto é de 25,6% .”, encontram-se três

tipos diferentes de numerais (decimal – “2,5”, percentagem – “25,6%” e cardinal – “2”). A aplicação de um identificador de decimais antes do identificador de uma percentagem poderia induzir o resultado em erro, uma vez que “25,6” também é um decimal. Verifica-se então que o reconhecimento de percentagens deve ser processado primeiro que os numerais cardinais e decimais.

Sendo assim, a ordem de identificação implementada no módulo de reconhecimento automático é a seguinte:

- Códigos em Unicode;
- Abreviaturas do tipo `social`, `others` e `money`;
- Siglas e Acrónimos;
- Endereços de correio electrónico;
- Endereços de rede (IP);
- Endereços de web (URL/URI);
- Ordinais;
- Percentagens;
- Decimais;
- Hexadecimais;
- Cardinais;
- Números desconhecidos
- Caracteres desconhecidos

O uso de expressões regulares para reconhecimento automático é aplicado no caso dos numerais e referências de rede; enquanto que a identificação de abreviações e acrónimos é implementada por expressões regulares em conjunto com uma pesquisa nos seus dicionários (neste ponto já em estruturas de dados internas).

A análise, identificação e conversão dos numerais apresenta ainda mais um desafio: a escolha de género. Por exemplo, no texto “Eram 2 maçãs e 2 maracujás.”, são identificados dois numerais cardinais – “2”, cuja conversão é “duas” na primeira ocorrência e “dois” na segunda ocorrência. Sendo assim, os algoritmos de numerais encontram-se preparados para pesquisar o contexto onde o numeral se encontra, procurando identificar o género a usar. O algoritmo utilizado consiste na procura de palavras que terminem em “a”, “as”, “ã” ou “ãs”. Caso estas palavras precedam o numeral, então o género feminino é aplicado.

No entanto, existem excepções à regra deste algoritmo, por exemplo, no texto “Eu vi 2 árvores.”, o numeral cardinal “2” deve ser convertido em “duas” e não “dois”. No entanto a palavra “árvores” termina em “es”. Estas palavras irregulares foram implementadas no algoritmo de detecção de género, sob a forma de uma lista de excepções que se encontra numa das bibliotecas auxiliares dos algoritmos de conversão.

Detecção de numerais cardinais
m/((?:\t+|^|\(|)((?:\+-|\+|-|- \+)?(?:(?:\d{1,3}(?:\.\d{3})+)|(?:\d+)))(?:\s|\.(?:\s)?|\, |\:(?:\s)?|$|\)|))/igos
Detecção de numerais decimais
m/(?:\t+|^|\(|)((?:\+-|\+|-|-\+)?(?:(?:\d{1,3}(?:\.\d{3})+,[0- 9]+)|(?:\d+,[0-9]+)))(?:\s|\.(?:\s)?|\, |\:(?:\s)?|$|\)|)/igos
Detecção de numerais ordinais
m/(?:\t+|^|\(|)((?:[0-9]+)(?:º(s)?|ª(s)?))(?:\s|\.(?:\s)?|\, |\:(?:\s)?|$|\)|)/igs
Detecção de percentagens
m/(?:\+\t+|^|\(|)((?:\+-|\+|-|-\+)?[0-9]+(?:[\.|,][0- 9]+)?[%])(?:\s|\.(?:\s)?|\, |\:(?:\s)?|$|\)|)/gs
Detecção de numerais hexadecimais
m/(0x([0-9|A-F]+))/gs
Detecção de códigos Unicode
m/(U\+(?:[0-9|A-F]+))/gs
Detecção de endereços web
m/(?:\t+|^|\(|)((?:http|ftp|https):\/\/[\w]+(?:\.[\w]+)(?:[\w\- \.\,\@\?\^\=\%\&\:\/\~\+\#]*[\w\- \@\?\^\=\%\&\/\~\+\#])?)(?:\s|\.(?:\s)?|\, |\:(?:\s)?|$|\)|)/igs)
Detecção de endereços de correio electrónico
m/(?:\t+|^|\(|)((?:[a-zA-Z0-9_\-\.]+)@(?:(?:\[[0-9]{1,3}\.[0- 9]{1,3}\.[0-9]{1,3}\.)|(?:(?:[a-zA-Z0-9\-]+\.)+))(?:[a-zA-Z]{2,4}|[0- 9]{1,3})(\]?))(?:\s|\.(?:\s)?|\, |\:(?:\s)?|$|\)|)/gs

Figura 7.17 – Exemplos de expressões regulares no reconhecimento automático.

Na Figura 7.17 estão representados alguns exemplos de expressões regulares que compõem os algoritmos de análise e identificação dos diversos elementos no texto.

Uma vez detectados os elementos e convertidos, estes são substituídos no texto original e marcados com os elementos AKML: `<num></num>`, `<abr></abr>`, `<acr></acr>` e `<net></net>`. São também recolhidas estatísticas acerca do número de ocorrências de um determinado elemento num texto. Estas estatísticas são apresentadas posteriormente no documento AKML de resultado final.

Apesar de a maioria de elementos num texto ser processada pelos analisadores, interpretadores e conversores referidos anteriormente, existem quase sempre elementos que não são detectados.

Por exemplo, no texto "A minha palavra passe é: 1234ABCD.", existem numerais que não são detectados como cardinais, uma vez que não se encontram isolados mas sim concatenados com letras. Estas letras também teriam de ser soletradas, uma vez que um conversor texto-fala não seria capaz de ler os numerais nem as letras, sem estes estarem nas suas formas extensíveis. Por esse motivo foram implementados dois algoritmos: um para números desconhecidos e outro para caracteres desconhecidos, com o objectivo de processarem este tipo de situações. Após a aplicação dos mesmos o texto convertido seria "A minha palavra passe é: um dois três quatro à bê cê dê.".

Tabela 7.28 - Lista de funções de autorecon.pm .

Funções exportadas pelo módulo:
&auto_reconhecimento(texto, usermodes);
&auto_cardinais(texto, género, usermode);
&auto_decimais(texto, género, usermode);
&auto_ordinais(texto, género, quantidade, usermode);
&auto_percentagens(texto, género, usermode);
&auto_url_uri_0(texto, usermode);
&auto_url_uri_1(texto, usermode);
&auto_ips(texto, usermode);
&auto_emails(texto, usermode);
&auto_num_desconhecidos(texto);
&auto_char_desconhecidos(texto);
&auto_abr(texto, usermode);
&auto_acr(texto, usermode);
&autorecon_rede_ref(texto);
&auto_num(texto, usermodes);
&auto_hex(texto, género);
&auto_unicode(texto);

A Tabela 7.28 apresenta a lista de funções do módulo de reconhecimento automático do AudioMathENGINE. As funções `auto_num` e `autorecon_rede_ref` funcionam como camadas de interface a outras funções de reconhecimento de numerais e de referências de rede, respectivamente.

A interface principal do módulo é a função `auto_reconhecimento`, onde se encontram as funções analisadores e identificadoras dos diversos elementos.

7.5 Módulo de Expressões Matemáticas

A análise, interpretação e conversão de expressões matemáticas é concretizada por este módulo. Apenas são suportadas expressões matemáticas representadas por MathML, quer sejam em *Presentation Markup* ou *Content Markup*.

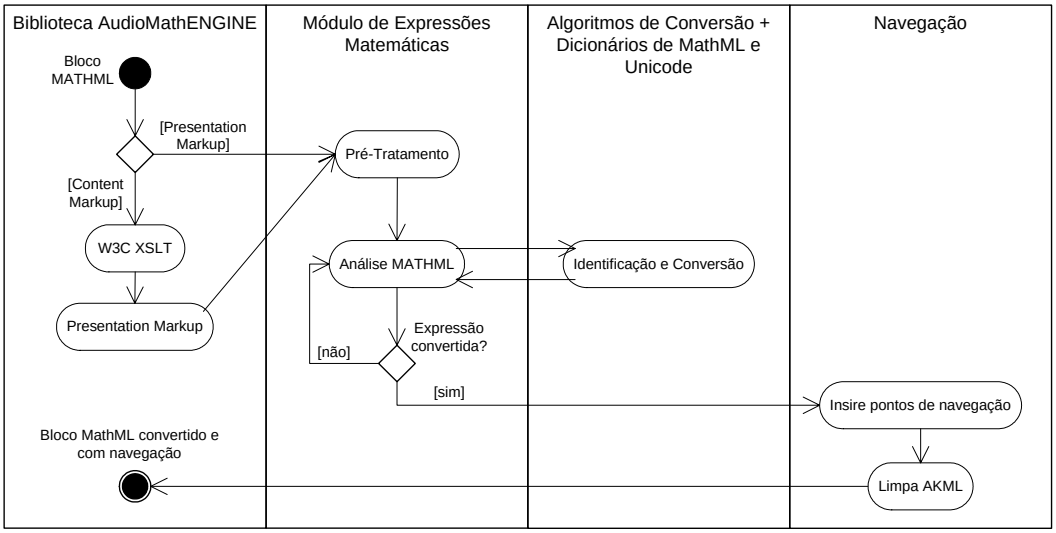


Figura 7.18 – Análise, Interpretação, Conversão e Navegação de MathML.

A Figura 7.18 representa de um modo simplificado as actividades de análise, interpretação, conversão e navegação de MathML. Como podemos observar, o módulo de expressões matemáticas interage com a biblioteca AudioMathENGINE, assim como com diversos algoritmos de conversão (através do módulo de reconhecimento automático), navegação, e dicionários de MathML e Unicode.

Nesta secção serão apresentados os diversos algoritmos de conversão, assim como a estruturação de dicionários e o processo de inclusão de pontos de navegação.

7.5.1 Dicionários de MathML e Unicode

Os documentos on-line com expressões matemáticas em MathML, geralmente, contêm entidades XML e entidades MathML nos seus textos e fórmulas. Por exemplo, no texto "A fórmula é <math><mn>4</mn><mo>+</mo><mn>3</mn>", encontram-se duas entidades XML ("ó" que é traduzido em "ó", e "é" que é traduzida em "é"), e uma entidade MathML ("+" que é traduzida em "mais").

Um conversor texto-fala não é capaz de interpretar os símbolos ou entidades contidos nos textos e reproduzi-los correctamente. Por esse motivo é necessário processar todas as entidades XML e MathML num documento.

Tal como já foi referido anteriormente (secção 7.3.5) existem dois tipos de dicionários auxiliares à conversão de expressões matemáticas:

- Dicionário de Unicode – `mathml_to_unicode.pm`
- Dicionário de MathML – `mathml_dictionary.pm`

O dicionário de Unicode do AudioMathENGINE contém apenas uma função (`mathml2unicode`) que converte todos os caracteres e entidades MathML definidas em (79), nos seus respectivos códigos Unicode. Este dicionário possuía, até ao momento da escrita da dissertação, mais de 4200 entradas de caracteres e entidades MathML. A Tabela 7.29 apresenta alguns exemplos dessas entradas neste dicionário.

Tabela 7.29 – Exemplos de Entidades MathML e respectivos códigos Unicode.

Símbolo	Entidade MathML	Código Unicode
∠	∠	U+02220
≅	≈	U+02248
∅	∅	U+02205-0FE00
=	=	U+0003D
e	ⅇ	U+02147
+	+	U+0002B
π	ϖ	U+003D6
θ	ϑ	U+003D1

O dicionário de MathML do AudioMathENGINE possui três funções:

- `operador_mathml` – traduz os operadores existentes na especificação MathML (23) para uma descrição em Português Europeu.
- `function_mathml` – traduz as funções de *Content Markup* existentes na especificação MathML (23) para uma descrição em Português Europeu.
- `unicode2pt` – traduz diversos códigos Unicode para uma descrição em Português Europeu.

Este dicionário possuía, até ao momento da escrita da dissertação, mais de 5200 entradas de operadores, funções e códigos Unicode. As tabelas seguintes apresentam alguns exemplos das entradas neste dicionário.

Tabela 7.30 – Exemplos de operadores MathML e respectivas descrições.

Operador MathML	Descrição
(	abrir parêntesis
{	abrir chaveta
lim	limite
max	máximo
exists	existe
forall	para todos
naturalnumbers	conjunto dos números naturais
prime	conjunto dos números primos

Tabela 7.31 – Exemplos de funções MathML e respectivas descrições.

Função MathML	Descrição
abs	módulo de
approx	aproximadamente igual a
integers	conjunto dos números inteiros
log	logaritmo de
sin	seno
sum	somatório
uplimit	limite superior
vectorproduct	produto vectorial

Tabela 7.32 – Exemplos de códigos Unicode e respectivas descrições.

Símbolo	Código Unicode	Descrição
!	U+00021	factorial
{	U+0007B	abrir chaveta
∞	U+0221E	infinito
α	U+003B1	alfa minúsculo
β	U+003B2	beta minúsculo
$\iint$	U+0222C	integral duplo
$\rightarrow$	U+02192	tende para
$\cap$	U+02229-0FE00	intersecção com

Sempre que o AudioMathENGINE recebe um documento com entidades MathML, estas são convertidas em Unicode e por fim em Português Europeu. Por exemplo, a fórmula matemática: "<math><mn>3</mn><mo>-</mo><mi>∞</mi></math>" possui uma entidade MathML ("∞"). Esta seria primeiro transformada em código Unicode ("U+0221E") e posteriormente traduzida para Português Europeu ("infinito"). A fórmula convertida seria lida como: "três menos infinito".

A conversão das entidades MathML em Unicode tem a vantagem de posteriormente se poder adicionar uma nova língua desenvolvendo apenas o dicionário de Unicode para a língua pretendida.

Durante o desenvolvimento foi considerada a hipótese de criar estes dicionários como documentos externos em AKML. Contudo, o facto de serem dicionários com milhares de entradas de definições obrigavam a custos de processamento desnecessários. Por esse motivo estes dicionários foram implementados internamente e encontram-se compilados no código do AudioMathENGINE. Perde-se em flexibilidade (uma vez que a modificação destes dicionários implica uma nova compilação), mas ganha-se em rapidez.

7.5.2 Conversão de Content Markup

Apesar de o AudioMathENGINE suportar ambos os tipos de marcação MathML, apenas o *MathML Presentation Markup* é suportado nativamente. No entanto algumas aplicações, como o *Maple* e *Mathematica*, trocam dados sobre a forma de *MathML Content Markup*. O suporte para este tipo de etiquetas de marcação do MathML poderia ser concretizado de duas formas:

- Implementação de um novo módulo de conversão para *Content Markup*.
- Transformação de *Content Markup* em *Presentation Markup* e reutilização do módulo de conversão de *Presentation Markup* já existente.

Optou-se pela segunda forma, sendo que assim se concentraria a lógica de análise, interpretação e conversão num único módulo. Embora a transformação eficaz de *Presentation Markup* em *Content Markup* seja algo ainda inatingível aquando da escrita desta dissertação, o mesmo não se pode dizer da transformação de *Content Markup* para *Presentation Markup*, onde a sua concretização é mais simples (apenas se descarta a informação semântica).

Existem três implementações distintas que concretizam esta transformação, todas elas através de folhas de estilo de transformação XML (XSLT):

- XSLT da W3C
- XSLT do Centro de Pesquisas de Ontário para Álgebra Computacional (ORCCA)[77]
- XSLT do navegador de páginas on-line Opera[78]

Foram realizados diversos testes de comparação entre as três folhas de estilo de transformação XML (XSLT) e que podem ser traduzidos na tabela seguinte:

Tabela 7.33 – Comparação entre XSLT para converter Content em Presentation.

XSLT	Eficácia	Direitos de Autor	Observações
W3C	satisfatória	pública	• entidades MathML criadas no plano 1 do Unicode.
ORCCA	muito eficaz	protegida	• entidades MathML criadas no plano 0 do Unicode. • maior suporte de etiquetas de marcação.
OPERA	pouco eficaz	protegida	• suporte de etiquetas de marcação bastante fraco.

Uma vez que a XSLT da W3C é de domínio público e a sua eficácia é satisfatória, foi a folha de estilo de transformação escolhida para ser incorporada no AudioMathENGINE. A única desvantagem desta XSLT encontra-se nas entidades MathML geradas no plano 1 do Unicode, isto é, são entidades não declaradas na especificação oficial do MathML, e como tal, não existentes nos dicionários Unicode e MathML do AudioMathENGINE. Por esse motivo, os dicionários foram complementados com as entidades MathML de plano 1, de uso mais frequente. A XSLT escolhida é processada através do utilitário *Microsoft XSLT Processor Version 3.0*, de uso livre e disponibilizado com a aplicação AudioMathENGINE. Após esta transformação, o código *Presentation Markup* é processado pelo fluxo normal do módulo de análise, interpretação e conversão.

[77] ORCCA – *Ontario Research Centre for Computer Álgebra* - http://www.orcca.on.ca/ .
[78] Opera *browser* - http://www.opera.com/ .

7.5.3 Conversão de Presentation Markup

Recorde-se que o conjunto de etiquetas de marcação *Presentation Markup* foi o conjunto escolhido para a interpretação oral das fórmulas matemáticas[79]. Por esse motivo, toda a lógica de análise, interpretação e conversão é implementada neste módulo – `pmathmlPT.pm`.

A interpretação de *Presentation Markup* é uma tarefa complexa. A expressão matemática encontra-se desprovida de informação semântica, pelo que é importante uma expressão ser decomposta nas várias sub expressões que a compõem e cada uma delas analisada no contexto onde se insere.

Sendo assim, o primeiro passo é o reconhecimento de etiquetas de marcação que possivelmente representem operações semânticas e assinalar as suas fronteiras de actuação ou delimitadoras com o elemento MathML `<mrow></mrow>`. Este elemento MathML representa, segundo (23) um elemento delimitador de um determinado contexto, pelo que a alteração da codificação MathML original não destrói a sua natureza, pelo contrário, evidencia a sua estrutura. A Figura 7.19 apresenta um exemplo desta técnica:

Fórmula:

$$\sqrt{\frac{a}{b}}$$

Código original:

```
<math>
  <msqrt>
    <mfrac>
      <mi>a</mi>
      <mi>b</mi>
    </mfrac>
  </msqrt>
</math>
```

Código pré-tratado:

```
<math>
  <mrow>
    <msqrt>
      <mrow>
        <mfrac>
          <mi>a</mi>
          <mi>b</mi>
        </mfrac>
      </mrow>
    </msqrt>
  </mrow>
</math>
```

Figura 7.19 – Exemplo de pré-processamento de código Presentation Markup.

[79] Ver capítulo de Introdução à Linguagem de Marcação Matemática.

Após o pré-processamento do código *Presentation Markup*, inicia-se um processo de validação, análise e interpretação, baseado num analisador de XML em PERL denominado `XML::Parser`. Este analisador é do tipo baseado em eventos (*SAX parser*) e configurado para:

- Receber texto com codificação ISO-8859-1 (onde se encontra inserida a codificação dos caracteres da língua portuguesa);
- Suportar o uso de XML *namespaces*;

O motor de interpretação e conversão de fórmulas matemáticas assenta sobre este analisador, onde são aproveitadas as capacidades de identificação dos elementos e atributos do código MathML à medida que este vai sendo processado. Se a expressão matemática tiver erros de sintaxe XML é retornada uma mensagem de erro e a conversão não é realizada.

Como este analisador XML é do tipo baseado em eventos, e não guarda histórico das marcas à medida que estas vão sendo processadas, foi desenvolvida uma estrutura de dados que guarda as seguintes informações:

- etiqueta de marcação detectada
- conteúdo original da etiqueta
- conteúdo convertido da etiqueta

É através desta estrutura de dados que vai sendo construída a hierarquia da fórmula matemática e suas sub expressões, de forma catalogada. Catalogação essa que permite analisar e interpretar o contexto da sub expressão e adoptar a conversão mais adequada.

Tomando o código da Figura 7.19 como exemplo, quando o elemento `</mrow>` antes de `</msqrt>` é detectado, o analisador já sabe que se encontra dentro de uma raiz quadrada e que dentro dela existe uma fracção. Na realidade, o analisador percepciona algo do género da Figura 7.20.

```
<math>
  <mrow>
    <msqrt>
      <SUB EXPRESSÃO = FRACÇÃO>
    </msqrt>
  </mrow>
</math>
```

Figura 7.20 – Exemplo de processamento de uma fórmula matemática – PARTE 1

Quando o analisador se encontra a processar o elemento `</mrow>` antes de `</math>`, este já processou a raiz quadrada com a fracção lá dentro (Figura 7.21). Note-se que a conversão dá-se quando o analisador encontra o elemento de saída de uma sub expressão e não quando encontra o inicio dela. Desta forma o analisador conhece o passado da expressão e o contexto onde esta se encontra inserida, processando a fórmula matemática com sucesso.

```
<math>
  <SUB EXPRESSÃO = RAIZ QUADRADA>
      <SUB EXPRESSÃO = FRACÇÃO>
  </SUB EXPRESSÃO>
</math>
```

Figura 7.21 – Exemplo de processamento de uma fórmula matemática – PARTE 2

Até ao momento da escrita desta dissertação, o AudioMathENGINE era capaz de reconhecer 29 das 33 etiquetas de marcação de *Presentation Markup*, sendo elas:

- `math` - elemento raiz da linguagem MathML;
- `mrow` - delimita o contexto das sub expressões matemáticas que compõem a fórmula;
- `mn`, `mi`, `mo` - identificam numerais, variáveis e operadores;
- `msqrt`, `mroot` - representam raízes;
- `msup`, `msub`, `msubsup`, `munderover`, `munder`, `none` – definem a posição de índices
- `mfrac` - elemento que representa uma fracção;
- `mtext`, `ms`, `merror` – definem texto;
- `mglyph` – representa símbolos;
- `mstyle` – define o estilo de elementos MathML;
- `menclose` – elemento que representa diversas notações;
- `annotation` – define anotações XML;
- `mtable`, `mtr`, `mtd` – representa matrizes ou sistemas de (in)equações;
- `maction` – define acções para elementos MathML;
- `maligngroup`, `malignmark`, `mphantom`, `mspace` – representam ajustes na notação da fórmula.

Para além da análise e interpretação das etiquetas de marcação, é necessário juntar palavras-chave na conversão de determinados operadores de forma a que a fórmula seja percepcionada não ambiguamente. De seguida são apresentadas considerações sobre as palavras-chave e as respectivas conversões realizadas pelo AudioMathENGINE.

Conversão de erros - `merror`

O elemento `merror` indica uma mensagem de erro na codificação da fórmula matemática, colocado intencionalmente no MathML. Sendo assim a sua conversão possui a seguinte estrutura: `"inicio de mensagem de erro:" + conteúdo convertido + "fim de mensagem de erro"`.

```
<math>
  <merror>
      <mtext> Operador desconhecido. </mtext>
  </merror>
</math>
```

```
inicio de mensagem de erro: Operador desconhecido. fim de mensagem de erro.
```

Figura 7.22 – Exemplo de conversão de `merror`.

Conversão de texto - `mtext` e `ms`

O elementos `mtext` e `ms` indicam apenas texto. Sendo assim, as suas conversões resultam apenas no texto que estas marcam, depois de passar pelo algoritmo de reconhecimento automático.

```
<math>
  <mtext>O Helder </mtext><ms>&eacute;</ms><mtext> o autor.</mtext>
</math>
```

```
O Helder é o autor.
```

Figura 7.23 – Exemplo de conversão de `mtext` e `ms`.

Conversão de glifos ou símbolos – `mglyph`

O elemento `mglyph` representa um símbolo ou glifo e contém um atributo `alt` com a descrição desse mesmo glifo. É essa descrição que é utilizada na conversão.

```
<math>
   <mglyph fontfamily="my-font" index="3" alt="3 da fonte do helder"/>
</math>
```

```
três da fonte do helder
```

Figura 7.24 – Exemplo de conversão de `mglyph`.

Conversão de numerais – `mn`

O elemento `mn` representa um numeral em MathML. Sendo assim, a sua conversão é dada pelo resultado de conversão do algoritmo de reconhecimento automático para numerais. Chama-se à atenção para o facto de que, segundo as regras de MathML, é permitido o uso de texto por extenso para representar um numeral.

```
<math>
   <mn>34</mn><mtext> ou </mtext><mn>trinta e cinco</mn>
</math>
```

```
trinta e quatro ou trinta e cinco
```

Figura 7.25 – Exemplo de conversão de `mn`.

Conversão de variáveis ou identificadores – `mi`

O elemento `mi` representa uma variável, um identificador ou uma unidade em MathML. O AudioMathENGINE apenas suportava até ao momento de escrita desta dissertação, o uso de variáveis ou identificadores. A conversão do conteúdo de um `mi` consiste num conjunto de testes para identificar se este é um operador MathML, um código Unicode, uma função MathML ou simplesmente texto que é soletrado.

```
<math>
   <mi>&pi;</mi><mtext> ou </mtext><mi>sin</mi>
</math>
```

```
pi minúsculo ou seno
```

Figura 7.26 – Exemplo de conversão de `mi`.

Para se suportar o uso de unidades no MathML, bastava implementar a lógica para a identificação do conteúdo do atributo `class='MathML-Unit'`, que identifica o valor de `mi` como uma unidade, e soletrar esse conteúdo.

Conversão de operadores ou funções - `mo`

O elemento `mo` representa um operador ou uma função MathML. A conversão deste elemento é idêntica à conversão do elemento `mi`.

```
<math>
   <mo>&int;</mo><mtext> ou </mtext><mo>&sum;</mo>
</math>
```

```
integral ou somatório
```

Figura 7.27 – Exemplo de conversão de `mo`.

Conversão de fracções ou números combinatórios - `mfrac`

O elemento `mfrac` é utilizado na representação de fracções ou números combinatórios, dependendo do valor do seu atributo `linethickness`. Se `linethickness=0` então é número combinatório.

No caso de um número combinatório, a conversão possui a seguinte estrutura: "número combinatório" + primeiro argumento convertido + "sobre" + segundo argumento convertido + "fim de combinatório".

No caso de uma fracção, a conversão possui a seguinte estrutura: "inicio de fracção" + "numerador:" + primeiro argumento convertido + "a dividir por denominador:" + segundo argumento convertido + "fim de fracção".

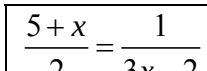
<table>
<tr><td>  </td><td><pre>
<math>
  <mrow>
    <mfrac>
      <mrow>
        <mn>5</mn>
        <mo>+</mo>
        <mi>x</mi>
      </mrow>
      <mn>2</mn>
    </mfrac>
    <mo>=</mo>
    <mfrac>
      <mn>1</mn>
      <mrow>
        <mn>3</mn>
        <mi>x</mi>
        <mo>−</mo>
        <mn>2</mn>
      </mrow>
    </mfrac>
  </mrow>
</math>
</pre></td></tr>
<tr><td colspan="2">inicio de fracção, numerador: cinco mais chis a dividir por denominador: dois fim de fracção igual a inicio de fracção, numerador: um a dividir por denominador: três chis menos dois fim de fracção</td></tr>
<tr><td>  </td><td><pre>
<math>
  <mo>(</mo>
  <mfrac linethickness="0">
    <mn>5</mn>
    <mn>2</mn>
  </mfrac>
  <mo>)</mo>
</math>
</pre></td></tr>
<tr><td colspan="2">abrir parêntesis número combinatório, cinco sobre dois fim de combinatório fechar parêntesis</td></tr>
</table>

Figura 7.28 – Exemplos de conversão de `mfrac`.

Conversão de potências – `msup`

O elemento msup é geralmente utilizado na identificação de potências. O AudioMathENGINE realiza testes que permitem identificar se o expoente tem valor 2, 3 ou outro; e se a base da potência é uma expressão complexa ou singular. São aplicadas as seguintes regras para os expoentes:

- Se expoente é 2, então aplica-se a estrutura: "elevado ao quadrado".
- Se expoente é 3, então aplica-se a estrutura: "elevado ao cubo".
- Se expoente é outro, então aplica-se a estrutura: "elevado ao expoente" + expressão convertida + "fim de expoente".

Para as bases do expoente são aplicadas as seguintes regras:

- Se base é uma função, então aplica-se a estrutura: "função convertida" + "elevado ao expoente" + expoente convertido + "fim de expoente" + "de" + argumento da função convertido.
- Se base é outro, então aplica-se a estrutura: "base convertida" + "elevado ao expoente" + expoente convertido.

$\left(x-3^2\right)^{\left(x-2^3\right)}-5=0$	`<math>`
	`<msup>`
	`<mrow>`
	`<mrow><mo>(</mo><mrow>`
	`<mi>x</mi>`
	`<mo>&minus;</mo>`
	`<msup><mn>3</mn>`
	`<mn>2</mn></msup>`
	`</mrow>`
	`<mo>)</mo>`
	`</mrow>`
	`</mrow>`
	`<mrow>`
	`<mrow><mo>(</mo><mrow>`
	`<mi>x</mi>`
	`<mo>&minus;</mo>`
	`<msup><mn>2</mn>`
	`<mn>3</mn></msup>`
	`</mrow>`
	`<mo>)</mo></mrow>`
	`</mrow>`
	`</msup>`
	`<mo>&minus;</mo>`
	`<mn>5</mn><mo>=</mo><mn>0</mn>`
	`</math>`

abrir parêntesis chis menos três ao quadrado fechar parêntesis elevado ao expoente: abrir parêntesis chis menos dois ao cubo fechar parêntesis fim de expoente menos cinco igual a zero	

Figura 7.29 – Exemplo de conversão de `msup`.

Conversão de raízes – `msqrt` e `mroot`

Uma raiz quadrada é representada pelo elemento `msqrt` em MathML, enquanto `mroot` representa uma raiz com um determinado índice.

No caso de uma raiz quadrada, a conversão possui a estrutura: `"raiz quadrada de" + expressão convertida + "fim de radicando"`.

No caso de uma raiz com um determinado índice, o AudioMathENGINE realiza testes para identificar se o índice é 2, 3 ou outro. São aplicadas as seguintes regras:

- Se índice é 2, então aplica-se a estrutura: `"raiz quadrada de" + expressão convertida + "fim de radicando"`.
- Se índice é 3, então aplica-se a estrutura: `"raiz cúbica de" + expressão convertida + "fim de radicando"`.
- Se índice é outro, então aplica-se a estrutura: `"raiz de índice:" + índice convertido + "fim de índice" + "e base" + base convertida + "fim de radicando`.

$\sqrt[x-1]{5+\sqrt{3}}$	<math> <mrow> <mroot> <mrow> <mn>5</mn> <mo>+</mo> <msqrt> <mn>3</mn> </msqrt> </mrow> <mrow> <mi>x</mi> <mo>−</mo> <mn>1</mn> </mrow> </mroot> </mrow> </math>
raíz de índice: chis menos um fim de índice e base: cinco mais raiz quadrada de três fim de radicando fim de radicando	

Figura 7.30 – Exemplo de conversão de `msqrt` e `mroot`.

Conversão de matrizes ou sistemas de (in)equações – `mtable`, `mtr` e `mtd`

Tanto as matrizes como os sistemas de equações ou inequações são representados em MathML como tabelas de dados. A identificação de um elemento `mtable` produz a conversão: `"tabela ou matriz" + conteúdos convertidos + "fim de tabela ou matriz"`.

A identificação de um elemento `mtr` produz uma conversão que depende do número de ocorrências. Por exemplo, se é a primeira ocorrência, a conversão é: `“primeira linha” + conteúdos convertidos + “fim de linha”`. Se for a segunda ocorrência, a conversão passa a ser: `“segunda linha” + conteúdos convertidos + “fim de linha”`, e assim sucessivamente.

A identificação de um elemento `mtd` produz uma conversão também dependente do número de ocorrências, mas desta vez relacionado com a posição vertical em colunas. Por exemplo, se for a primeira ocorrência, a conversão seria: `“primeira coluna” + conteúdos convertidos + “fim de coluna”`.

<table>
<tr>
<td>$\begin{cases} x - y = 2 \\ x + y = 5 \end{cases}$</td>
<td><pre>
<math>
  <mrow>
    <mo>{</mo>
     <mrow>
       <mtable>
         <mtr>
           <mtd>
             <mrow>
               <mi>x</mi>
               <mo>−</mo>
               <mi>y</mi>
               <mo>=</mo>
               <mn>2</mn>
             </mrow>
           </mtd>
         </mtr>
         <mtr>
           <mtd>
             <mrow>
               <mi>x</mi>
               <mo>+</mo>
               <mi>y</mi>
               <mo>=</mo>
               <mn>5</mn>
             </mrow>
           </mtd>
         </mtr>
       </mtable>
     </mrow>
   </mrow>
 </math>
</pre></td>
</tr>
<tr>
<td colspan="2">abrir chaveta tabela ou matriz primeira linha primeira coluna chis menos ípsilon igual a dois fim de coluna fim de linha segunda linha primeira coluna chis mais ípsilon igual a cinco fim de coluna fim de linha fim de tabela ou matriz</td>
</tr>
</table>

Figura 7.31 – Exemplo de conversão de `mtable`, `mtr` e `mtd`.

Conversão de sub índices – `msub`

Os sub índices são representados pelo elemento `msub` em MathML. A conversão deste elemento é baseada na seguinte estrutura: `expressão base convertida + “inicio de índice” + índice convertido + “fim de índice”`.

$a_{n+1} + b_{n+2}$	`<math>`
	`  <mrow>`
	`  <msub>`
	`  <mi>a</mi>`
	`  <mrow>`
	`  <mi>n</mi>`
	`  <mo>+</mo>`
	`  <mn>1</mn>`
	`  </mrow>`
	`  </msub>`
	`  <mo>+</mo>`
	`  <msub>`
	`  <mi>b</mi>`
	`  <mrow>`
	`  <mi>n</mi>`
	`  <mo>+</mo>`
	`  <mn>2</mn>`
	`  </mrow>`
	`  </msub>`
	`  </mrow>`
	`</math>`

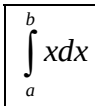
à de índice éne mais um fim de índice mais bê de índice éne mais dois fim de índice	

Figura 7.32 – Exemplo de conversão com índices.

Conversão de integrais – `msubsup`, `munderover` ou `munder`

Os integrais podem ser constituídos por: limite inferior, limite superior e variável de integração. Em MathML, a posição dos limites pode ser representada pelos elementos `msubsup`, `munderover` ou `munder`.

Dependendo do tipo de limites, a estrutura de conversão pode ser: `“integral com” + “limite inferior” + limite convertido + “fim de limite” + “e limite superior” + limite convertido + “fim de limite” + “de” + expressão convertida + variável de integração convertida`.



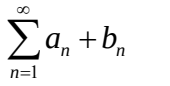
```
<math>
    <mrow>
      <mstyle displaystyle="true">
        <mrow>
          <munderover>
            <mo>&int;</mo>
            <mi>a</mi>
            <mi>b</mi>
          </munderover>
          <mrow>
            <mi>x</mi>
            <mi>d</mi>
            <mi>x</mi>
          </mrow>
        </mrow>
      </mstyle>
    </mrow>
</math>
```

```
integral com limite inferior: à fim de limite e limite superior: bê fim de limite de: chis dê chis
```

Figura 7.33 – Exemplo de conversão de integral.

Conversão de somatórios – `msubsup`, `munderover` ou `munder`

A conversão de somatórios que façam uso dos elementos msubsup, munderover ou munder possui uma estrutura idêntica à dos integrais (descrito anteriormente).



```
<math>
    <mrow>
      <mstyle displaystyle="true">
        <munderover>
          <mo>&sum;</mo>
          <mrow>
            <mi>n</mi>
            <mo>=</mo>
            <mn>1</mn>
          </mrow>
          <mi>&infin;</mi>
        </munderover>
        <mrow>
          <msub>
            <mi>a</mi>
            <mi>n</mi>
          </msub>
          <mo>+</mo>
          <msub>
            <mi>b</mi>
            <mi>n</mi>
          </msub>
        </mrow>
      </mstyle>
    </mrow>
</math>
```

```
somatório com limite inferior: éne igual a um  fim de limite e limite superior: infinito fim de limite de: à de índice éne fim de índice mais bê de índice éne fim de índice
```

Figura 7.34 – Exemplo de conversão de somatório.

Os resultados anteriormente apresentados resultaram de um estudo profundo das estruturas matemáticas na procura e aferição de regras de conversão.

Apresentam-se, de seguida, alguns exemplos de conversão de expressões matemáticas realizadas pelo AudioMathENGINE (código MathML disponível no Anexo D):

$\sqrt{x}+\frac{1}{x-y}>y\times x$ raiz quadrada de chis fim de radicando mais inicio de fracção, numerador: um a dividir por denominador: chis menos ípsilon fim de fracção maior do que ípsilon vezes chis
$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ inicio de fracção, numerador: menos bê mais ou menos raiz quadrada de bê ao quadrado menos quatro à cê fim de radicando a dividir por denominador: dois à fim de fracção
$\lim_{x\to\infty}\frac{x+\sin^2(x)}{x+1}=\lim_{x\to\infty}\frac{1+2\sin(x)\cos(x)}{1}=\lim_{x\to\infty}(1+\sin(2x))$ limite de chis tende para infinito de: inicio de fracção, numerador: chis mais seno ao quadrado de abrir parêntesis chis fechar parêntesis a dividir por denominador: chis mais um fim de fracção igual a limite de chis tende para infinito de: inicio de fracção, numerador: um mais dois seno de abrir parêntesis chis fechar parêntesis cosseno de abrir parêntesis chis fechar parêntesis a dividir por denominador: um fim de fracção igual a limite de chis tende para infinito de: abrir parêntesis um mais seno de abrir parêntesis dois chis fechar parêntesis fechar parêntesis
$\frac{1}{2}+\frac{1}{1+\frac{x}{3}}=\frac{1+\frac{x}{3}}{2}$ inicio de fracção, numerador: um a dividir por denominador: dois fim de fracção mais inicio de fracção, numerador: um a dividir por denominador: um mais inicio de fracção, numerador: chis a dividir por denominador: três fim de fracção fim de fracção igual a inicio de fracção, numerador: um mais inicio de fracção, numerador: chis a dividir por denominador: três fim de fracção a dividir por denominador: dois fim de fracção

$\sqrt[(x-1)]{x+x^{\sqrt{5}-1}} > \sqrt{1-x}$

raiz de índice: abrir parêntesis chis menos um fechar parêntesis fim de índice e base: chis mais chis elevado ao expoente: raiz quadrada de cinco fim de radicando menos um fim de expoente fim de radicando maior do que raiz quadrada de um menos chis fim de radicando

$x^{(a+b)^{(c-5)^2}}$

chis elevado ao expoente: abrir parêntesis à mais bê fechar parêntesis elevado ao expoente: abrir parêntesis cê menos cinco fechar parêntesis ao quadrado fim de expoente fim de expoente

$$\sin\alpha \pm \sin\beta = 2\sin\frac{\alpha \pm \beta}{2}\cos\frac{\alpha \mp \beta}{2}$$

seno de alfa minúsculo mais ou menos seno de beta minúsculo igual a dois seno de inicio de fracção, numerador: alfa minúsculo mais ou menos beta minúsculo a dividir por denominador: dois fim de fracção cosseno de inicio de fracção, numerador: alfa minúsculo menos ou mais beta minúsculo a dividir por denominador: dois fim de fracção

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

tabela ou matriz primeira linha primeira coluna um fim de coluna segunda coluna dois fim de coluna fim de linha segunda linha primeira coluna três fim de coluna segunda coluna quatro fim de coluna fim de linha fim de tabela ou matriz vezes tabela ou matriz primeira linha primeira coluna chis fim de coluna fim de linha segunda linha primeira coluna ípsilon fim de coluna fim de linha fim de tabela ou matriz igual a tabela ou matriz primeira linha primeira coluna cinco fim de coluna fim de linha segunda linha primeira coluna seis fim de coluna fim de linha fim de tabela ou matriz

$$A = \int_{1}^{2} \int_{-3x+6}^{4x-x^2} dydx + \int_{2}^{4} \int_{0}^{4x-x^2} dydx$$

à igual a integral com limite inferior: um fim de limite e limite superior: dois fim de limite de: integral com limite inferior: menos três chis mais seis fim de limite e limite superior: quatro chis menos chis ao quadrado fim de limite de: dê ípsilon dê chis mais integral com limite inferior: dois fim de limite e limite superior: quatro fim de limite de: integral com limite inferior: zero fim de limite e limite superior: quatro chis menos chis ao quadrado fim de limite de: dê ípsilon dê chis

$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

seno de chis igual a somatório com limite inferior: éne igual a zero fim de limite e limite superior: infinito fim de limite de: abrir parêntesis menos um fechar parêntesis elevado ao expoente: éne fim de expoente inicio de fracção, numerador: chis elevado ao expoente: dois éne mais um fim de expoente a dividir por denominador: abrir parêntesis dois éne mais um fechar parêntesis factorial fim de fracção

Figura 7.35 – Exemplos de conversões do AudioMathENGINE.

7.5.4 Navegação de Fórmulas

Durante a análise, interpretação e conversão dos diversos elementos que compõem uma expressão matemática, as sub expressões são marcadas com a etiqueta de marcação `<submat></submat>`. Deste modo é construída uma hierarquia ou árvore de sub expressões (Figura 7.36).

$$\sqrt{a^{2+b^2} + b^2}$$

```
<submat>
    <submat> raiz quadrada de </submat>
    <submat>
      <submat>
         <submat> á </submat>
         <submat> elevado ao expoente: </submat>
         <submat>
            <submat> <num>dois</num> </submat>
            <submat> mais </submat>
            <submat>
               <submat> bê </submat>
               <submat> ao quadrado </submat>
            </submat>
          </submat>
          <submat> fim de expoente </submat>
       </submat>
       <submat> mais </submat>
       <submat>
          <submat> bê </submat>
          <submat> ao quadrado </submat>
       </submat>
     </submat>
     <submat> fim de radicando </submat>
</submat>
```

Figura 7.36 - Aplicação de <submat> para navegação.

É a partir desta árvore de sub expressões que é implementada a navegação das fórmulas matemáticas pelo módulo `navegacao.pm`.

Neste módulo, as informações sobre sub expressões são agregadas em níveis e sub níveis de navegação, em que cada nível possui um conjunto de nós (ou pontos de navegação). Isto é, o módulo de navegação converte e agrega os elementos `submat` em elementos `node`.

Este processo é concretizado através de um analisador XML que concatena os valores de todos os elementos `submat` dentro de um determinado nível, atribuindo-lhe, posteriormente, um elemento `node` com esses valores concatenados.

```
<node value=" raiz quadrada de á elevado ao expoente: dois mais bê ao
quadrado fim de expoente mais bê ao quadrado fim de radicando ">
   <node value=" raiz quadrada de "></node>
   <node value=" á elevado ao expoente: dois mais bê ao quadrado fim
de expoente mais bê ao quadrado ">
      <node value=" á elevado ao expoente: dois mais bê ao quadrado
fim de expoente ">
           <node value=" á "></node>
           <node value=" elevado ao expoente: "></node>
           <node value=" dois mais bê ao quadrado ">
                <node value=" dois "></node>
                <node value=" mais "></node>
                <node value=" bê ao quadrado ">
                      <node value=" bê "></node>
                      <node value=" ao quadrado "></node>
                </node>
            </node>
            <node value=" fim de expoente "></node>
       </node>
       <node value=" mais "></node>
       <node value=" bê ao quadrado ">
            <node value=" bê "></node>
            <node value=" ao quadrado "></node>
       </node>
    </node>
    <node value=" fim de radicando "></node>
</node>
```

Figura 7.37 – Conversão do <submat> em <node>.

A navegação de fórmulas matemáticas será abordada, mais tarde, num capítulo próprio desta dissertação.

8 Plataforma de Testes e Demonstração – AudioMathGUI

No âmbito desta dissertação foi também desenvolvida uma aplicação gráfica de interacção com o utilizador (GUI[80]) - AudioMathGUI - com o objectivo de permitir o teste e demonstração das capacidades de análise, interpretação e conversão do AudioMathENGINE. O AudioMathGUI possui as seguintes características técnicas:

- Desenvolvido em C# para ambientes WINDOWS 2000 e XP;
- Suporte de conversores texto-fala SAPI4 e SAPI5;
- Suporte para *Microsoft Agents*[81];
- Possibilidade de gravação de ficheiros áudio sintetizados;
- Conversão de páginas on-line;
- Representação gráfica de expressões matemáticas;
- Visualização de resultados e estatísticas sob a forma de árvores XML;
- Sistema de navegação de expressões matemáticas incorporado.

Figura 8.1 – Exemplo da aplicação AudioMathGUI.

[80] *Graphical User Interface*
[81] *Microsoft Agent Homepage* - http://www.microsoft.com/msagent/default.asp

Neste capítulo pretende-se apresentar sumariamente a aplicação desenvolvida, a sua arquitectura, funcionalidades e integração com as diversas tecnologias usadas durante esta dissertação, e que permitem a interpretação áudio de expressões matemáticas.

8.1 Arquitectura e Desenho

Sendo uma aplicação em C#, o AudioMathGUI é uma aplicação com um estilo de arquitectura orientado a objectos (*object-oriented*), composta por diversas classes que implementam a sua lógica de forma estruturada.

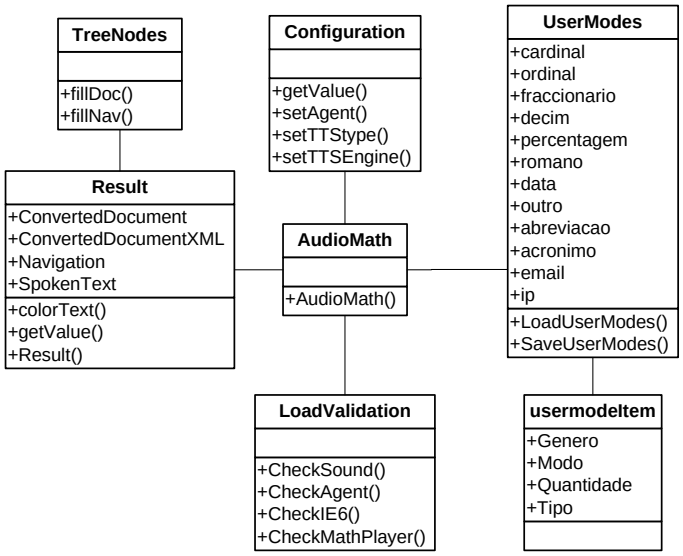


Figura 8.2 – Diagrama de classes simplificado do AudioMathGUI.

Na Figura 8.2 apenas se encontram representadas as classes principais do AudioMathGUI, são elas:

- `AudioMath` – Classe cujos métodos criam os objectos gráficos da aplicação.
- `UserModes` – Classe cujos métodos processam o ficheiro AKML que contém os modos de utilizador para o AudioMathENGINE. Usa a classe `usermodeItem` como estrutura de dados para manipulação de informação nos objectos da aplicação gráfica.
- `usermodeItem` – Estrutura de dados simples que contém as propriedades possíveis de um modo de utilizador.
- `TreeNodes` – Classe cujos métodos criam as árvores de navegação da fórmula matemática e do texto convertido.

- `Configuration` – Classe cujos métodos processam o ficheiro AKML de configuração da aplicação gráfica.
- `LoadValidation` – Classe cujos métodos realizam diversos testes de validação de condições para o uso da aplicação. São realizados os seguintes testes:
    - `CheckSound` – verifica a existência de uma placa de som no computador.
    - `CheckAgent` – verifica a existência de um *Microsoft Agent* no sistema.
    - `CheckIE6` – verifica a existência do *Internet Explorer 6.0* no sistema. Sem esta aplicação, não é possível a visualização gráfica das expressões matemáticas.
    - `CheckMathPlayer` – verifica a existência do *MathPlayer* no sistema. Tal como o IE6, sem esta aplicação não é possível a visualização gráfica das expressões matemáticas.
- `Result` – Classe cujos métodos processam os resultados do AudioMathENGINE. Alguns dos métodos principais são:
    - `colorText` – Identifica conversões do AudioMath e evidencia-as através do uso de um esquema de cores.
    - `getValue` – Recolhe informação estatística sobre as conversões.
    - `Result` – Converte o texto original através do AudioMathENGINE.

O processo de inicialização do AudioMathGUI (Figura 8.3) consiste na inicialização do componente AudioMathENGINE e numa sequência de testes para verificação do ambiente onde a aplicação vai correr. A reprodução áudio das expressões matemáticas depende da existência de uma placa de som e de pelo menos um conversor texto-fala instalado no sistema; enquanto que a visualização gráfica das expressões matemáticas só é conseguida através da aplicação *MathPlayer* e do programa *Internet Explorer 6.0*.

Na Figura 8.4 estão representadas as actividades que o AudioMathGUI executa na conversão de um documento. Este é enviado para o AudioMathENGINE que o analisa, interpreta, converte e retorna um documento AKML com informação estatística, documento convertido e árvore de navegação. À aplicação gráfica compete o processamento deste documento de modo a extrair as diversas informações, e encaminhá-las para as classes que as apresentam ao utilizador.

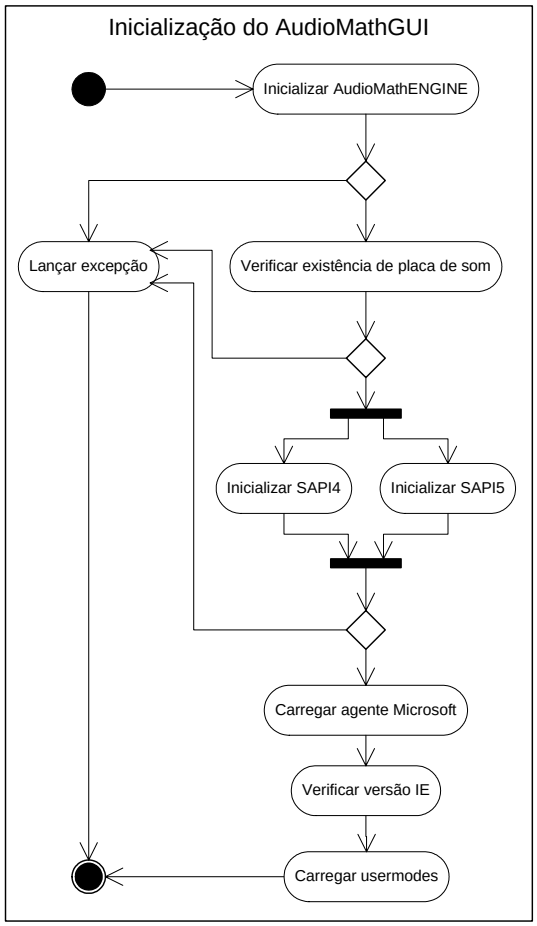


Figura 8.3 – Actividades de inicialização no AudioMathGUI.

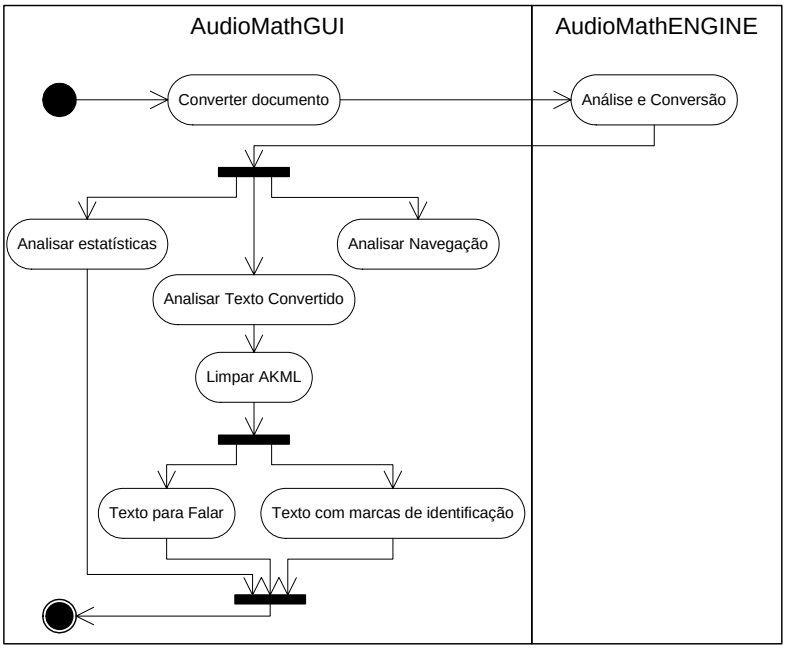


Figura 8.4 – Actividades de conversão no AudioMathGUI.

8.2 AKML no GUI

No capitulo referente ao AudioMathENGINE foi apresentada a especificação da linguagem de marcação AKML. Esta linguagem subdivide-se em quatro subconjuntos de etiquetas de marcação:

- Configuração
- Dicionário
- Conversão e Resultados
- Navegação

É no grupo de Configuração que se encontram as opções do modo de utilizador e de inicializações, utilizadas pelo AudioMathGUI. A Figura 8.5 apresenta os modos de utilizador usados por defeito pelo AudioMathGUI:

```
<akml    version="1.0"    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="../schema/AKML.xsd">
<engine lang="pt">
 <conversionsettings>
   <!-- numerals -->
   <module name="numeral" usermode="0">
    <conversion name="cardinal" usermode="0" gender="m" />
    <conversion name="ordinal" usermode="0" gender="m" quantity="s" />
    <conversion name="fractional" usermode="0" gender="m" />
    <conversion name="decimal" usermode="0" gender="m" />
    <conversion name="percentage" usermode="0" gender="m" />
    <conversion name="roman" usermode="0" gender="m" type="c" />
    <conversion name="date" usermode="0" />
   </module>
   <!-- abbreviations -->
   <module name="abbreviation" usermode="1" />
   <!-- acronyms -->
   <module name="acronym" usermode="1" />
   <!-- network -->
   <module name="network" usermode="0">
    <conversion name="ip" usermode="0" />
   </module>
  </conversionsettings>
</engine>
</akml>
```

Figura 8.5 – Modos de utilizador usados por defeito no AudioMathGUI.

Relativamente às configurações iniciais do AudioMathGUI, estas podem estar relacionadas com:

- localização do ficheiro de modos de utilizador;
- conversor texto-fala usado por defeito na aplicação gráfica;
- tipo de conversor texto-fala (SAPI4 ou SAPI5) usado por defeito;
- uso ou não do *Microsoft Agent*.

A Figura 8.6 demonstra a especificação das configurações iniciais em AKML:

```
<akml    version="1.0"    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="../schema/AKML.xsd">
  <gui lang="pt">
   <options>
<set name="usermodes" value="akml-gui/usermodes.xml" />
<set name="ttsengine" value="BDB8AFED-0397-4A82-B998-F4EE8E485D44" />
<set name="ttstype" value="sapi4" />
<set name="agent" value="on" />
   </options>
  </gui>
</akml>
```

Figura 8.6 – Configurações iniciais usadas por defeito no AudioMathGUI.

8.3 Descrição Sumária de Funcionalidades

A plataforma de testes e desenvolvimento – AudioMathGUI – possui diversas funcionalidades, das quais se destacam:

- Análise, interpretação e conversão de documentos on-line ou não, via AudioMathENGINE;
- Visualização gráfica de expressões matemáticas através da aplicação *MathPlayer*;
- Navegação de expressões matemáticas com recurso à fala sintetizada e ao teclado;
- Recolha de informação estatística acerca das conversões realizadas;
- Identificação visual e em estrutura XML dos elementos convertidos;
- Manipulação dos algoritmos de conversão através dos modos de utilizador;
- Gravação para ficheiros áudio do resultado da conversão texto-fala;
- Selecção e alteração das propriedades de conversores texto-fala do tipo SAPI4 e SAPI5;

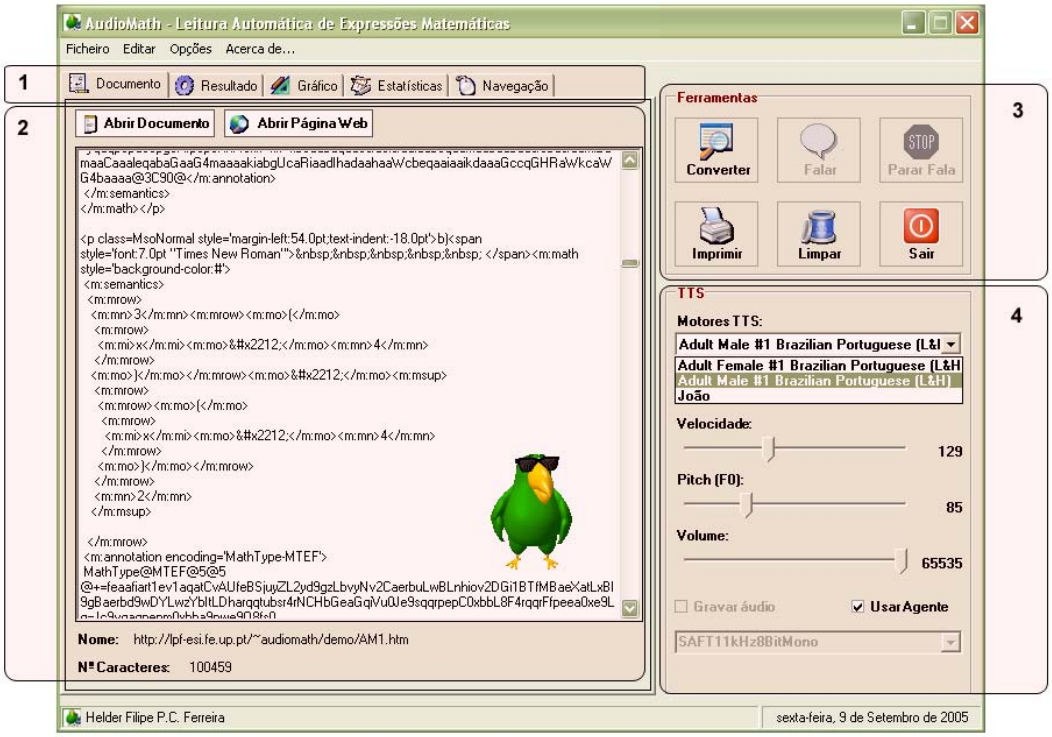


Figura 8.7 – Aspecto visual do AudioMathGUI.

Como se pode observar da Figura 8.7, existem quatro áreas funcionais no AudioMathGUI:

- Vistas de Trabalho (área 1) – existem 5 vistas possíveis:
  - Documento – vista onde é possível abrir um documento e modificá-lo.
  - Resultado – vista onde é apresentado o documento convertido e onde é possível verificar através de um esquema de cores os locais das conversões.
  - Gráfico – vista que apresenta graficamente o documento e respectivas expressões matemáticas (se estas existirem).
  - Estatísticas – vista que apresenta o documento convertido sob a forma de árvore onde se evidenciam os elementos convertidos; e onde é possível recolher e visualizar informação estatística acerca das conversões (por exemplo: tempo de conversão ou o número de abreviações convertidas).
  - Navegação – vista que possibilita a navegação das fórmulas matemáticas em conjugação com a síntese da fala.
- Área de Trabalho ou Visualização (área 2) – espaço reservado à modificação de documentos, visualização de resultados ou outros tipos de interacções.

- Ferramentas gerais (área 3) – conjunto de ferramentas, sendo elas:
    - Converter – envia o documento original para o AudioMathENGINE e responde com os resultados obtidos pela conversão.
    - Falar – reproduz sob a forma de fala sintetizada os resultados obtidos por uma conversão. Utiliza o motor de conversão texto-fala que está activo no momento.
    - Parar Fala – cancela o pedido de sintetização de fala.
    - Imprimir – imprime os resultados obtidos de uma conversão.
    - Limpar – esvazia as caixas de texto, repondo as áreas de trabalho no seu aspecto original.
    - Sair – abandonar o programa.
- Selecção e controlo de conversores texto-fala (área 4) – conjunto de opções que permitem seleccionar o conversor texto-fala com o qual os resultados são reproduzidos oralmente; assim como modificar as suas propriedades mais importantes: velocidade, frequência fundamental e volume. Usando SAPI5 é também possível a gravação para ficheiros áudio dos textos sintetizados.

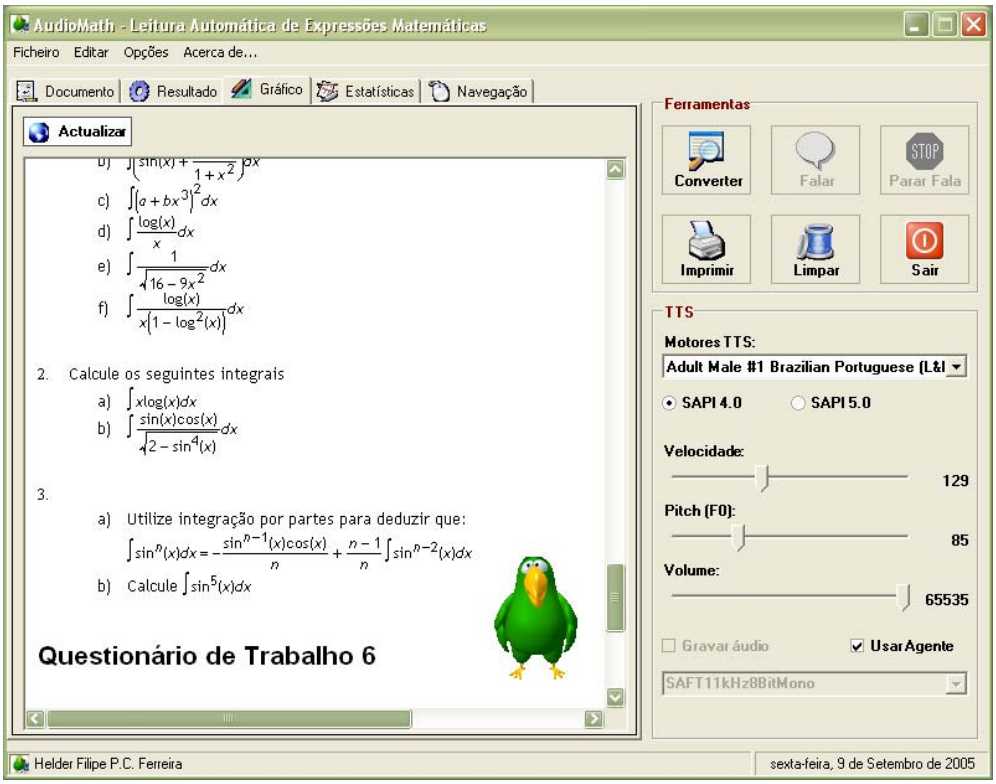


Figura 8.8 - Visualização gráfica de documentos no AudioMathGUI.

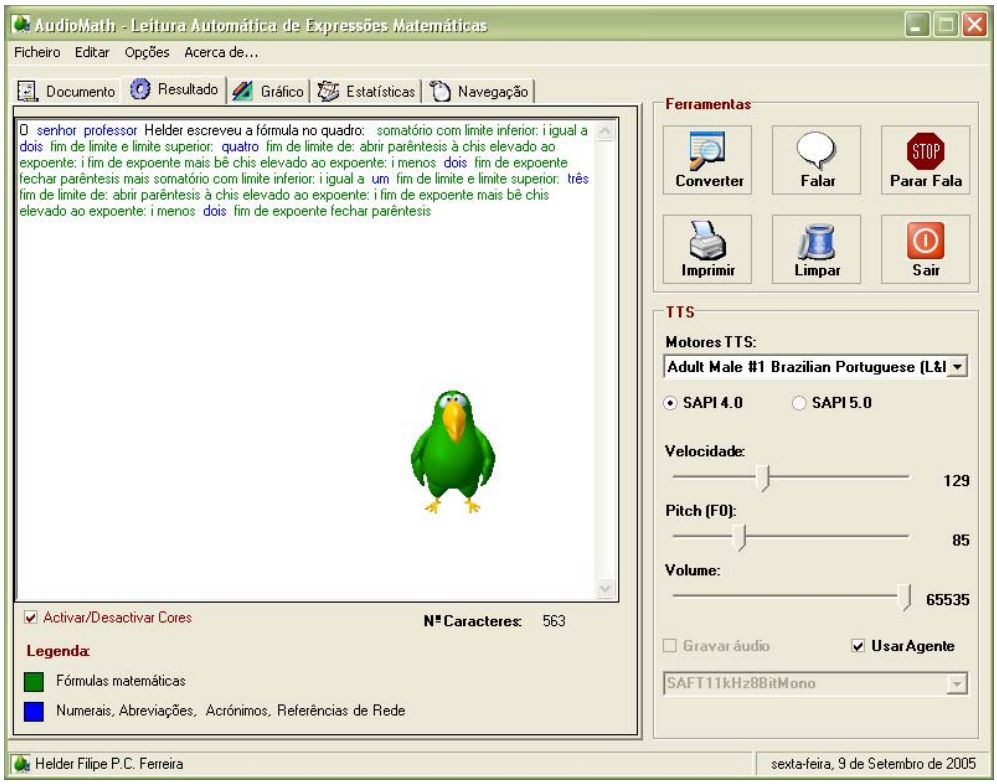


Figura 8.9 - Vista Resultado no AudioMath.

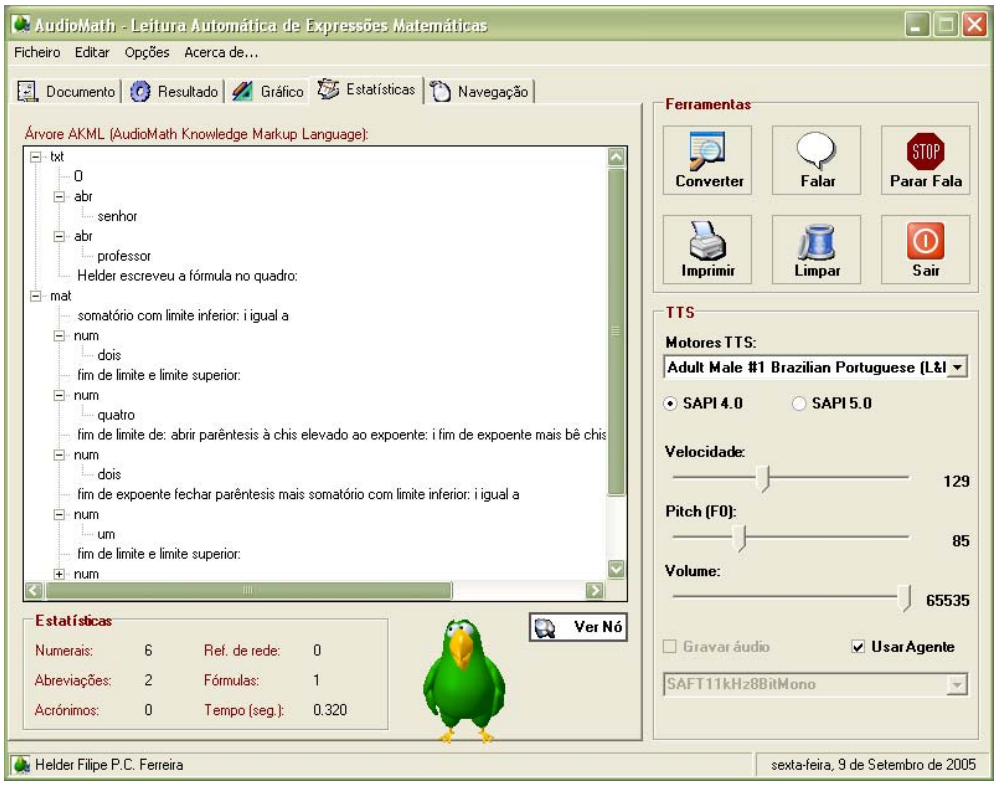


Figura 8.10 - Vista Estatísticas no AudioMath.

Na Figura 8.9 pode-se observar o esquema de cores utilizado para identificar as conversões realizadas num determinado texto. Esta propriedade pode ser activada/desactivada mediante a escolha do utilizador.

Na Figura 8.10 observa-se o mesmo resultado mas num formato diferente, sob a forma de uma árvore AKML, e inclui-se um conjunto de estatísticas da conversão.

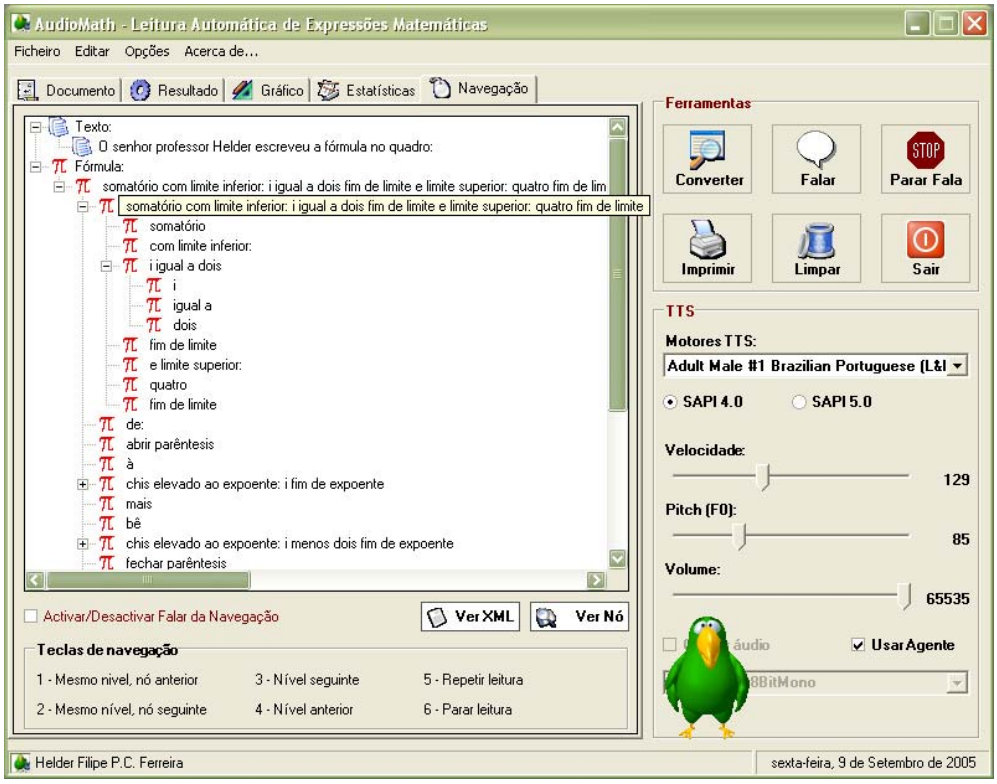


Figura 8.11 – Vista de Navegação no AudioMathGUI.

A Figura 8.11 representa a vista de navegação do AudioMathGUI. Como se pode observar, o bloco de texto normal é separado do bloco da expressão matemática.

A navegação é facilitada através do rato ou do teclado (6 teclas com propriedades especiais). Tendo em consideração que, num processo de testes, não é agradável para o utilizador a constante sintetização falada dos nós da navegação, foi disponibilizada a opção de desactivação da síntese da fala.

Existe ainda uma segunda vista de navegação que permite a visualização do código XML numa janela exterior à aplicação (Figura 8.12).

Na Figura 8.13 observa-se o formulário de modos de utilizador, que permite a alteração do comportamento por defeito, dos algoritmos de análise, interpretação e conversão do AudioMathENGINE.

Figura 8.12 – Segunda vista de navegação no AudioMathGUI.

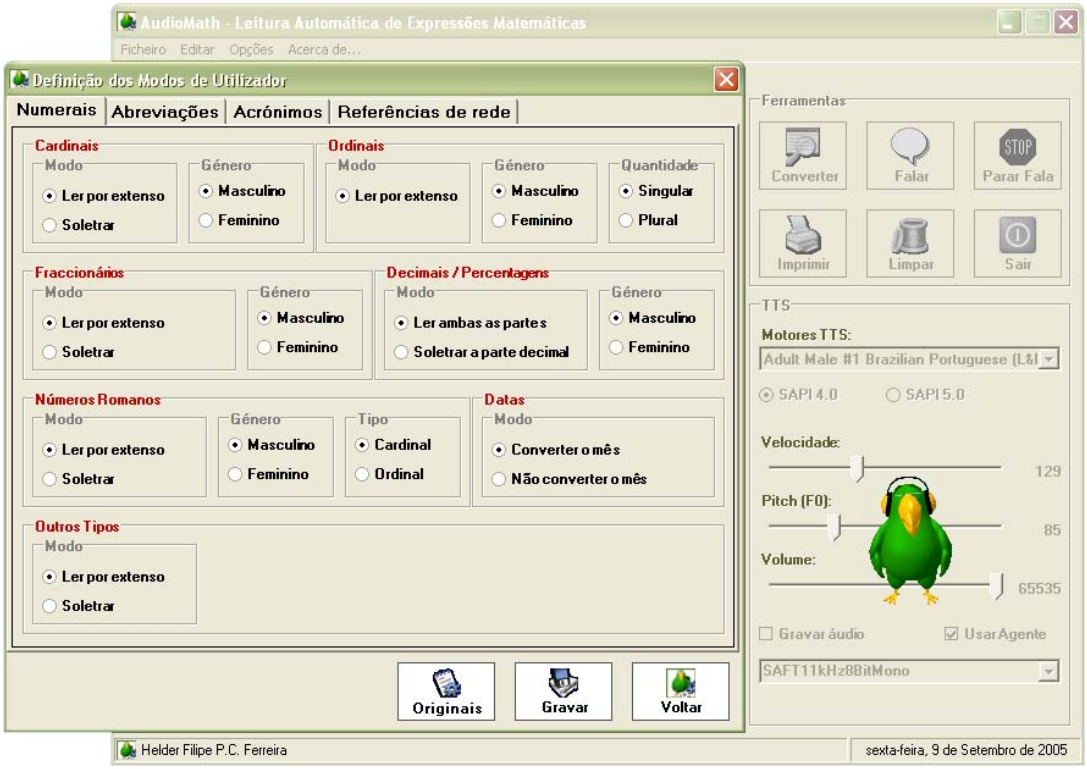


Figura 8.13 – Definição dos modos de utilizador no AudioMathGUI.

8.4 Integração C#, SAPI e AudioMathENGINE

Foi já referido anteriormente que o AudioMathENGINE, desenvolvido no âmbito desta dissertação, é um componente .NET. Assim sendo, o seu uso na aplicação AudioMathGUI pode ser efectuado com o seguinte excerto de código C#:

```
using Thesis.AudioMath;

Thesis.AudioMath.AudioMathEngineNET audiomath =
Thesis.AudioMath.AudioMathEngineNET();

str texto = audiomath.GetURL("http://lpf-esi.fe.up.pt");
int ok = audiomath.Convert(texto);
if (ok==1)
{
      str resultado = audiomath.GetResult();
}
str resultado_sem_AKML = audiomath.CleanAKML(resultado);
```

Figura 8.14 – Integração C# e AudioMathENGINE.

Para além da utilização do AudioMathENGINE, esta aplicação faz uso de conversores texto-fala SAPI4 e SAPI5. Existem diversas diferenças entre ambos os conversores (já abordado no capítulo dedicado aos conversores texto-fala). Em termos práticos, o conversor SAPI4 suporta a síntese da fala com e sem *Microsoft Agent*, enquanto o SAPI5 apenas a síntese da fala sem *Microsoft Agente*. A utilização de SAPI4 sem agente só é possível através de um componente *ActiveX* denominado *Microsoft Direct Speech Synthesis* (DirectSS). A identificação de conversores texto-fala SAPI4 poderia ser descrita pelo seguinte código em C#:

```
// Listar e disponibilizar conversores SAPI4
AxACTIVEVOICEPROJECTLib.AxDirectSS TTS;
AxAgentObjects.AxAgent peedy;
string ttsname = "";
this.motores.Items.Clear();
for(int i=1; i<= TTS.CountEngines; i++)
{
      ttsname = TTS.ModeName(i);
      this.motores.Items.Add(ttsname);
}
// Seleccionar motor para falar
engine = this.motores.SelectedIndex+1;
nengine = TTS.ModeID(engine);
//Escolhe o motor em DirectSS
TTS.CurrentMode = engine;
//Escolhe o motor no agente
peedy.Characters[“Peedy”].TTSModeID = "{"+nengine+"}";
```

Figura 8.15 – Selecção de conversores texto-fala SAPI4.

A identificação de conversores texto-fala SAPI5 poderia ser obtida pelo seguinte excerto de código em C#:

```
// Listar e disponibilizar vozes SAPI5
SpeechLib.SpVoice SAPI5_Voice;
SpeechLib.ISpeechObjectTokens voices;
this.SAPI5_Voice = new SpVoice();
this.voices = this.SAPI5_Voice.GetVoices(null, null);
for(int i= 0; i< voices.Count; i++)
{
   motores.Items.Add(voices.Item(i).GetAttribute("Name").ToString());
}
this.motores.SelectedIndex = 0;
//Seleccionar voz sapi5
SAPI5_Voice.Voice = voices.Item(this.motores.SelectedIndex);
```

Figura 8.16 – Selecção de conversores texto-fala SAPI5.

O controlo da síntese da fala em ambos os conversores texto-fala é feito através de métodos próprios do SAPI4 e SAPI5, que permitem ajustar a velocidade, frequência fundamental e volume da síntese; assim como iniciar, pausar e parar a reprodução áudio. A Figura 8.17 ilustra um exemplo:

```
//controlo de velocidade, F0 e volume em SAPI4 com agente
AxAgentObjects.AxAgent peedy;
AgentObjects.IAgentCtlCharacter speaker;
speaker = peedy.Characters[“Peedy”];
controlTTSSapi4 = "\\spd="+ vel.Text +"\\ \\pit=" + f0.Text+ "\\
\\vol="+ vol.Text +"\\";
speaker.Speak(comtrolTTSSapi4 + TextToSpeak, null);

//controlo em SAPI5
SpeechLib.SpVoice SAPI5_Voice;
SAPI5_Voice.Volume = int.Parse(this.vol.Text);
SAPI5_Voice.Rate = int.Parse(this.vel.Text);
SAPI5_Voice.Speak("<sapi><pitch middle='"+ f0.Text +"'
/>"+TextToSpeak+ "</sapi>", SpeechVoiceSpeakFlags.SVSFlagsAsync);
```

Figura 8.17 – Exemplos de controlo de velocidade, f0 e volume em SAPI4 e SAPI5.

Como se pode verificar pela Figura 8.17, o controlo dos diversos parâmetros da síntese da fala é realizado através de linguagens de marcação para tecnologias de fala, anteriormente descritas nesta dissertação.

Página em branco.

9 Serviço On-line de Demonstração - AudioMathWEB

No âmbito desta dissertação foi desenvolvido um sítio *web* com o objectivo de apresentar informação acerca da mesma, planeamento, progresso, documentação, referências e demonstração de parte dos resultados obtidos. O motor de análise, interpretação e conversão, AudioMathENGINE, foi adaptado para uma aplicação on--line, assumindo agora a denominação de AudioMathWEB.

Neste capítulo pretende-se apresentar sumariamente o sítio web desenvolvido e duas aplicações de relevo: editor de expressões matemáticas on-line; e o serviço de conversão on-line.

9.1 Sítio Web

Durante a dissertação, e com o objectivo de apresentar diversas informações sobre o *AudioMath*, foi desenvolvido um sítio web - http://lpf-esi.fe.up.pt/~audiomath – disponibilizando conteúdos, tais como:

- Resultados obtidos durante o projecto final de licenciatura.
- Documentação sobre a dissertação de mestrado.
- Planeamento e progresso da dissertação.
- Serviço público de demonstração e conversão de páginas on-line.
- Publicações (artigos) e apresentações em conferências.
- Referências on-line.

Incluídos no sítio web encontram-se duas ferramentas on-line:

- **Editor de expressões matemáticas público**, cujo objectivo é auxiliar a criação e produção de conteúdos matemáticos em SVG[82], MathML[83] e LaTeX[84].
- **Serviço público de demonstração e conversão de páginas on-line**, cujo objectivo se prende com a análise, interpretação e conversão simplificada de páginas on-line com conteúdos matemáticos em MathML.

[82] SVG – *Scalable Vector Graphics* - http://www.w3.org/Graphics/SVG/
[83] MathML – *Mathematical Markup Language* - http://www.w3.org/Math/
[84] LaTeX – *a document preparation system* - http://www.latex-project.org/

Figura 9.1 - Página principal do sítio web.

Reforça-se a ideia de que todas as páginas on-line foram desenvolvidas tendo em conta as regras de acessibilidade – WCAG[85] 1.0 (12) – da W3C, onde se procurou implementar todas as opções de prioridade 1, 2 e 3, atingindo o nível máximo de acessibilidade.

9.2 Editor de expressões matemáticas público

Tal como já foi referido anteriormente, existe uma ferramenta on-line no sítio web da dissertação que permite a produção de conteúdos matemáticos em diversos formatos publicáveis na Internet.

A aplicação on-line foi desenvolvida através de um projecto GNU[86], sendo portanto de uso livre, e denomina-se sMArTH - http://sourceforge.net/projects/smarth . Consiste numa aplicação on-line implementada em SVG e que pode ser instalada em qualquer servidor de páginas web.

[85] WCAG – *Web Content Accessibility Guidelines*
[86] GNU - *General Public License* - http://www.gnu.org/ .

Uma das vantagens de ser implementada em SVG prende-se com o facto de ser escalável e funcionar em diversas resoluções. Os utilizadores amblíopes poderão usufruir desta propriedade aumentando a resolução da placa gráfica ou fazendo *zoom* sobre os elementos SVG. (122)(123)

Para se usar a aplicação é necessário possuir os seguintes requisitos:

- *Internet Explorer 6*[87] instalado.
- *Plug-in Adobe SVG Viewer*[88] instalado.
- Ligação à Internet.

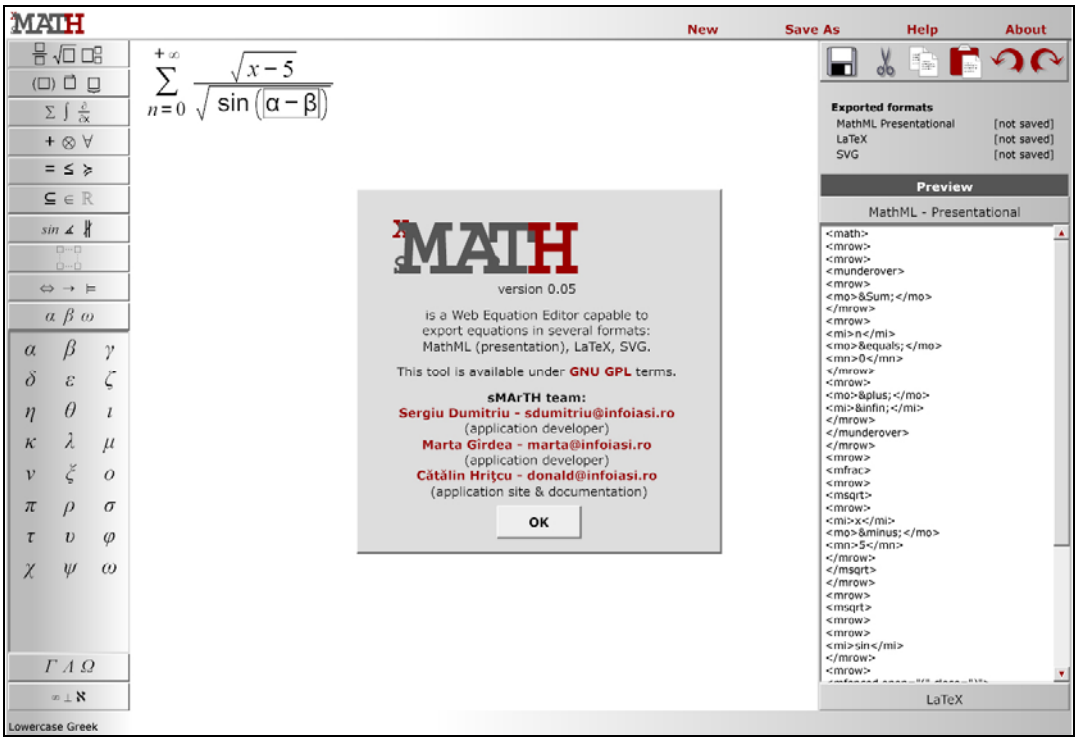


Figura 9.2 - sMArTH on-line.

Como se pode verificar pela Figura 9.2, a aplicação permite a criação e gravação de conteúdo matemático sob o formato de *MathML Presentation Markup*[89], LaTeX e SVG. É também possível a visualização do código que vai sendo construído à medida que a expressão matemática é "desenhada".

[87] Internet Explorer 6 - http://www.microsoft.com/windows/ie/downloads/default.mspx
[88] Adobe SVG Viewer - http://www.adobe.com/svg/viewer/install/main.html
[89] W3C, MathML Presentation Markup - http://www.w3.org/TR/MathML2/chapter3.html

9.3 Serviço de conversão on-line

Durante o projecto final de licenciatura do autor surgiu a ideia de se criar um serviço de conversão on-line público que actuasse como um serviço de acessibilidade para pessoas com necessidades especiais, nomeadamente cegos e amblíopes. Essa ideia foi implementada durante o desenvolvimento desta dissertação, com a disponibilização, sob a forma de demonstração (Figura 9.3), de um serviço de conversão de páginas on-line com conteúdos matemáticos em *MathML Presentation Markup*.

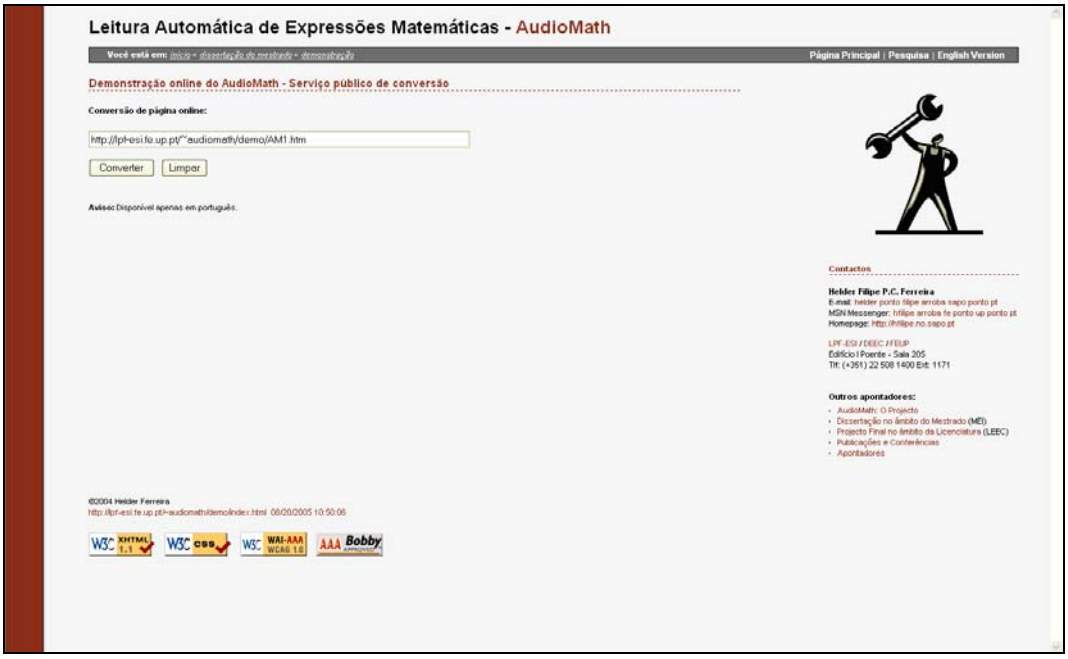


Figura 9.3 - Página de demonstração e serviço público de conversão on-line.

Para a implementação deste serviço foi necessário adaptar o motor AudioMathENGINE, e ajustar a interface da aplicação para um ambiente web. Desta adaptação surgiu o AudioMathWEB. Como se pode observar (Figura 9.4), existe uma aplicação (PERL[90] *script*[91]) que faz o interface entre o utilizador e o AudioMathWEB. Através desta aplicação, denominada `converte.web`, é possível outros autores de sítios web utilizarem as capacidades de conversão do AudioMathWEB. Basta para isso construírem aplicações que chamem o URI: `http://lpf-esi.fe.up.pt/~audiomath/cgi-bin/demo/converte.web?url=URL`, onde a variável URL deve ser substituída pelo endereço da página on-line a converter.

[90] PERL - Practical Extraction and Report Language - http://www.perl.org/
[91] *Script* – conjunto de linhas de código interpretadas e não compiladas.

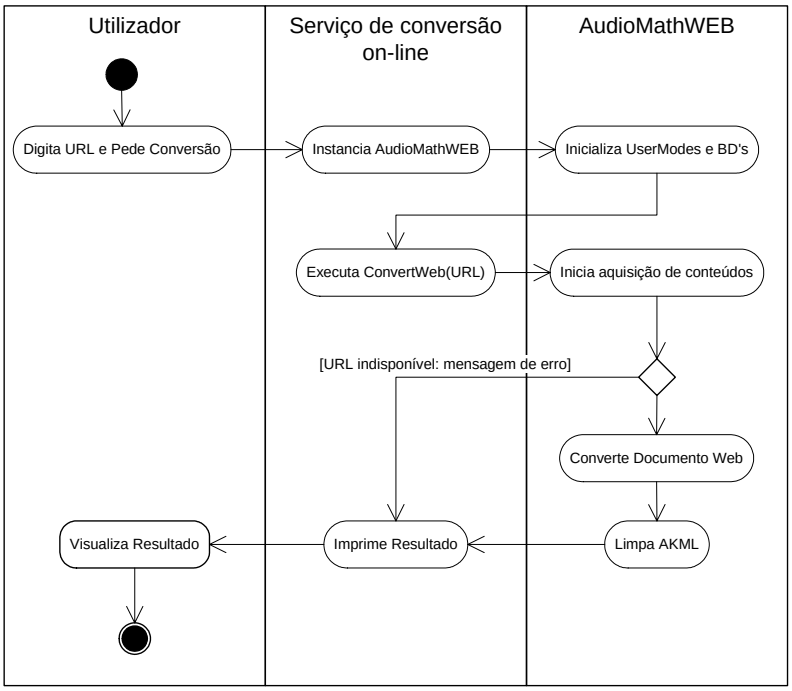


Figura 9.4 - Diagrama de Actividade - Conversão on-line com AudioMathWEB.

Para efeitos de demonstração do serviço de conversão, foi disponibilizado na Internet uma página on-line com um conjunto de exercícios de matemática (Figura 9.5), baseados nas fichas de trabalho da cadeira de Análise Matemática 1 da Licenciatura de Engenharia Electrotécnica e Computadores, da Faculdade de Engenharia da Universidade do Porto.

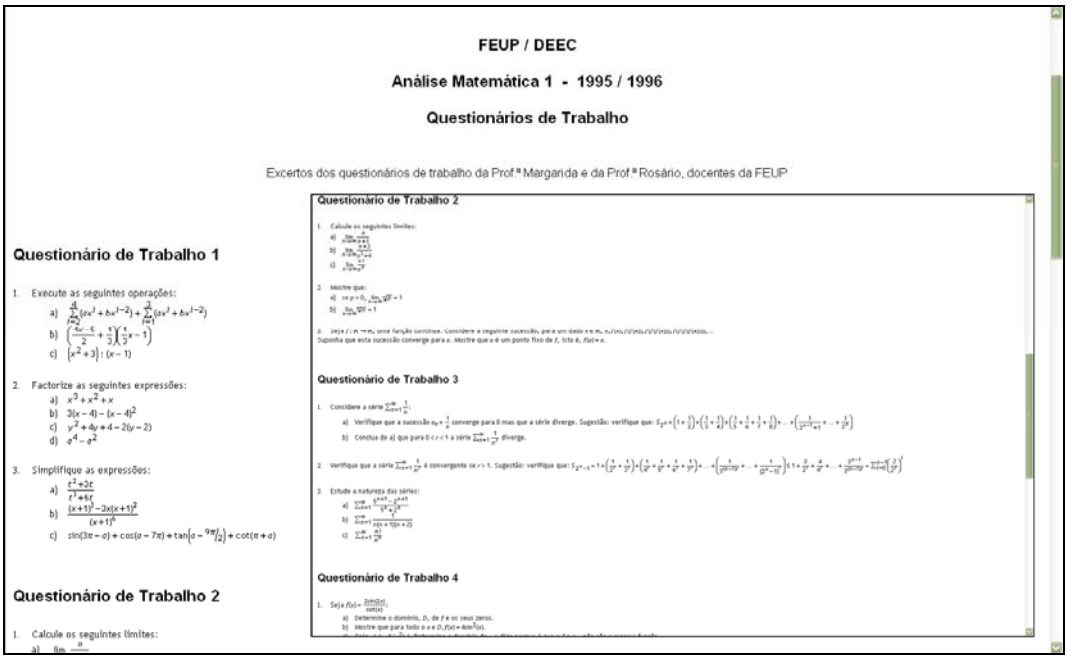


Figura 9.5 - Amostra das fichas de trabalho on-line.

Esta página de demonstração (http://lpf-esi.fe.up.pt/~audiomath/demo/AM1.htm) foi produzida com recurso ao: *Microsoft Word 2003*[92] e *Design Science MathType 5.2*[93]. Após a criação do documento foi activada a macro *MathPage* da aplicação *MathType* que converteu o documento do *Word* num ficheiro *web* com todo o conteúdo matemático codificado em *MathML Presentation Markup*. Posteriormente a página foi disponibilizada num servidor de páginas web.

No entanto, e uma vez que foi utilizado o MathType para produzir os conteúdos, a visualização da página web só funciona a 100% quando vista através do *Internet Explorer 6.0* e em conjunto com o *plug-in MathPlayer*[94] da Design Science.

A Figura 9.6 mostra o resultado da conversão da página de demonstração:

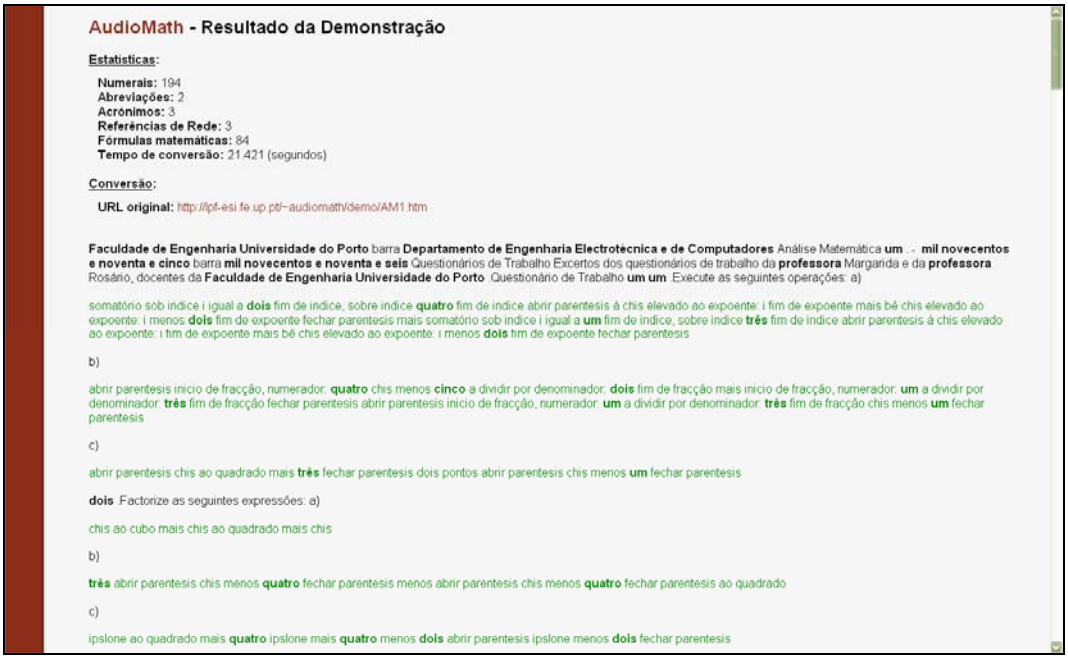


Figura 9.6 - Resultado de uma conversão on-line.

São ainda apresentados resultados estatísticos tais como: a quantidade de numerais, abreviaturas, acrónimos, referências de rede e fórmulas matemáticas que foram identificadas e convertidas; assim como a duração total desde a aquisição dos conteúdos até à sua apresentação.

[92] Microsoft Word 2003 - http://office.microsoft.com/en-us/FX010857991033.aspx
[93] Design Science MathType - http://www.dessci.com/en/products/mathtype/
[94] Design Science MathPlayer - http://www.dessci.com/en/products/mathplayer/

10 Leitura Humana de Expressões Matemáticas

Um dos desafios colocados à realização desta dissertação prendeu-se com o facto da ausência, quase total, de documentos ou estudos relacionados com a leitura de expressões matemáticas. Existem no entanto alguns trabalhos de referência tais como: Fateman (124), Karshmer (54), Stevens (49) e Raman (46).

Apesar de Fateman ter elaborado uma teoria focada no reconhecimento oral de expressões matemáticas, apresentou diversos artigos de opinião[95] sobre como os humanos deveriam ler oralmente expressões matemáticas, nomeadamente: números, expressões algébricas, somatórios e integrais. Karshmer, por sua vez, procurou basear os seus estudos na forma como é percepcionada, na audição, a matemática focando-se na área da psicologia. Enquanto que Stevens e Raman desenvolveram teorias focadas na interpretação e análise prosódia ao nível da frequência fundamental (F0)[96] e pausas. Contudo Stevens baseia-se apenas em modelos de expressões algébricas e Raman não apresenta na sua tese resultados e regras provindos da experimentação, mas sim um método de formatação áudio – AFL, *Audio Formatting Language*.

Nesta secção são apresentados conceitos teóricos relacionados com o aparelho fonador humano; assim como resultados extraídos de diversas experiências realizadas no âmbito desta dissertação, e cujo objectivo se foca na análise prosódica da leitura humana de expressões matemáticas.

10.1 A Fala Humana

A fala é uma forma de comunicação eficiente e natural, adaptada às capacidades cognitivas humanas, que se transmite sob a forma de sinais acústicos produzidos por meios articulatórios do tracto vocal.

Em engenharia, na análise e síntese da fala, estes sinais acústicos são recolhidos através de um microfone e digitalizados. Posteriormente estas longas cadeias numéricas podem ser quantificadas e comprimidas permitindo diversas aplicações sob poderosos algoritmos de codificação, desde a concretização de motores de conversão texto-fala às telecomunicações, passando por temas importantes com impacto social como a acessibilidade e as diferentes formas de comunicação.

---

[95] Lista de artigos de Richard Fateman - http://www.cs.berkeley.edu/~fateman/algebra.html .
[96] Na literatura inglesa, frequência fundamental é traduzida no termo *pitch*.

10.1.1 Produção de fala

É através da libertação de ar dos pulmões e das vibrações das cordas vocais, que o tracto vocal é capaz de produzir os sinais acústicos ou "fala". Na Figura 10.1 pode-se observar a composição mais importante do nosso corpo que nos permite falar, o aparelho fonador humano, constituído por: cavidade nasal, lábios, língua, dentes, cavidade bocal, queixo, palato duro, palato mole, faringe, laringe, esófago, traqueia, pulmões e diafragma.

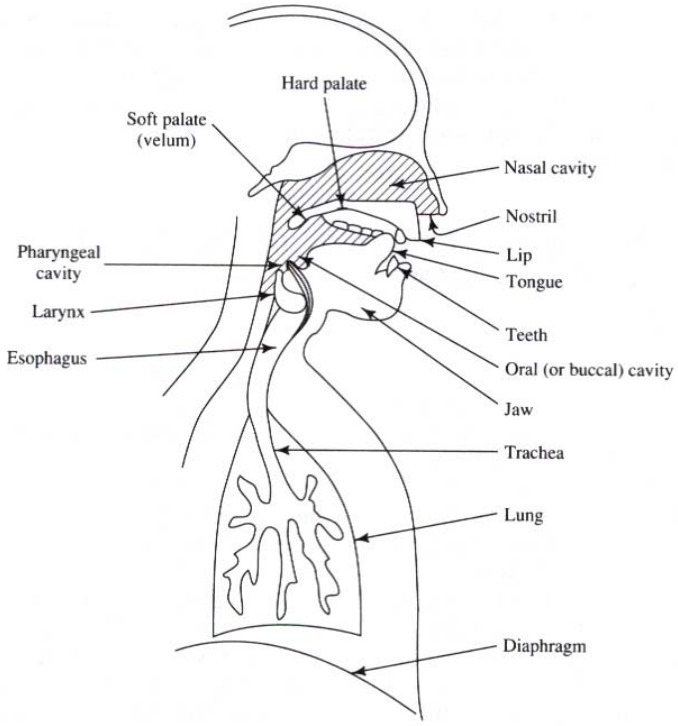


Figura 10.1 - Aparelho fonador humano. (82)

10.1.2 Vozeamento/Não Vozeamento

É na laringe que se encontram as cordas vocais. Conforme a acção das mesmas, o ar liberto pelos pulmões pode ser colocado em vibração ou não. Surge então a primeira separação entre os sons acústicos produzidos, em dois tipos distintos: *Vozeados* e *Não Vozeados*. Os sons vozeados são aqueles em que o ar liberto pelos pulmões é colocado em vibração através da acção das cordas vocais, isto é, através da abertura e fecho da glote (espaço entre as cordas vocais). São exemplos disso as vogais e algumas consoantes como: /m/ e /z/. Os sons não vozeados passam simplesmente pela laringe, sem que as cordas vocais vibrem, produzindo apenas sons com turbulência devido às constrições do tracto vocal. São exemplo disso algumas consoantes como: /s/ e /f/.

Uma forma fácil de experimentar a identificação e distinção entre sons vozeados e não vozeados consiste na colocação de dedos na zona exterior da garganta abaixo do queixo, onde se situam as cordas vocais. A pronunciação de uma vogal ou uma consoante poderá fazer as mesmas vibrarem ou não. No entanto, observando as formas de onda da Figura 10.2, é possível identificar facilmente quando um som é vozeado ou não.

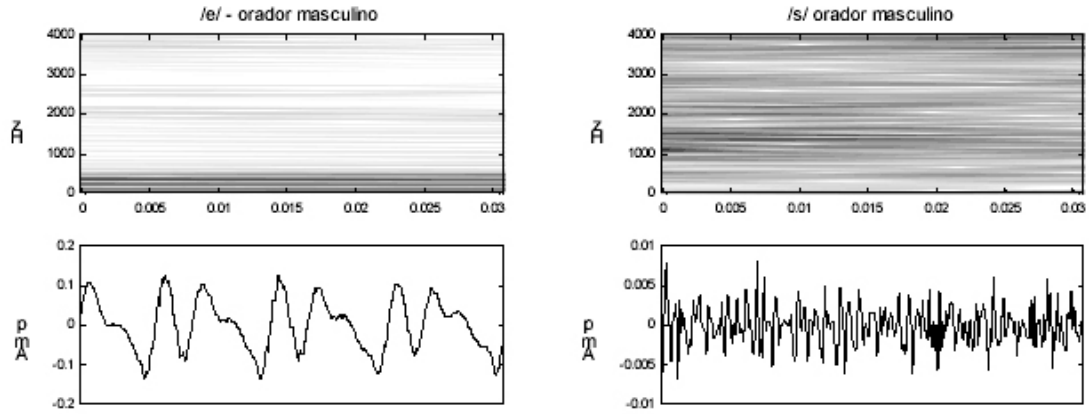


Figura 10.2 – Diferença entre vozeado e não vozeado. (125)

Do lado esquerdo é representado um som vozeado `/e/`, e do lado direito um som não vozeado `/s/`. Por cima das formas de onda estão representados os espectrogramas respectivos. São nítidas as diferenças entre os dois sinais. Repare-se na periodicidade do som vozeado, e na irregularidade do som não vozeado. Pode-se também afirmar que um som não vozeado contém muitas mais passagens por zero do que um som vozeado.

A partir da observação dos espectrogramas é possível afirmar que um som não vozeado possuí em geral maior energia nas altas frequências que as zonas vozeadas.

10.1.3 Frequência Fundamental – F0

Foi já referido anteriormente que os sons vozeados são produzidos pela passagem do ar que vem dos pulmões e que atravessa a laringe onde a abertura e o fecho da glote, juntamente com as constrições do resto do tracto vocal, produzem sons com uma certa periodicidade. Essa periodicidade, dada a frequência de fecho e abertura da glote, é denominada *frequência fundamental*, F0.

A frequência fundamental depende da dimensão e espessura da glote. Por exemplo, para oradores masculinos a F0 varia entre 50-250Hz, enquanto que para oradores femininos a F0 varia entre 120-300Hz. No entanto pode mesmo chegar a atingir os 500Hz no caso das crianças.

Mas nem só a estrutura física influência a F0; o estado psicológico e emocional de uma pessoa influência bastante as características do seu sinal de fala. Quanto mais forçada for a voz (voz emocional), mais rápido será o fecho das cordas vocais.

10.1.4 Tracto Vocal

Para além da vibração ou não das cordas vocais, o resto das constrições que formam o som produzido é provocado pela estrutura física do tracto vocal humano. Existem estudos de modelos de tractos vocais e suas co-articulações, com o objectivo de se produzirem modelos articulatórios para a síntese da fala. São exemplos disso os trabalhos de (126), (127), (128) e (129). Contudo estes modelos são bastante complexos tendo em consideração que é necessário simular diversos fenómenos físicos. Outro factor deve-se ao facto de o tracto vocal ser diferente em cada ser humano. No entanto, existem algumas considerações sobre modelos básicos.

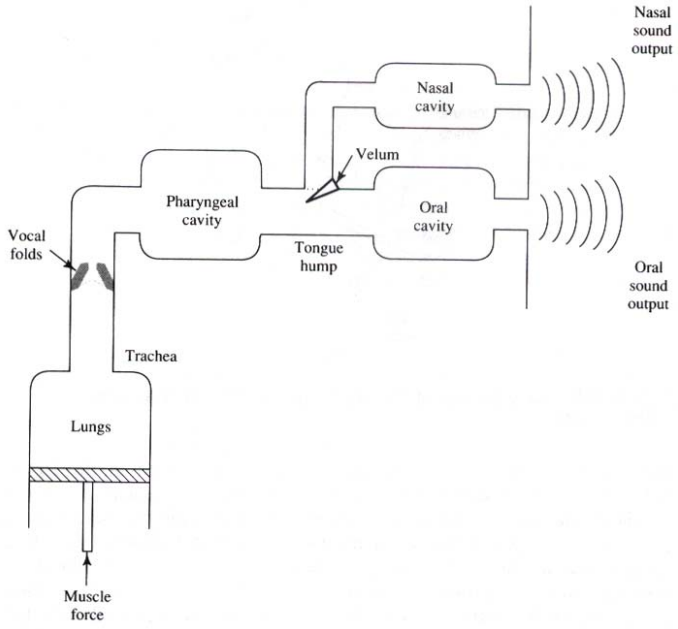


Figura 10.3 - Modelo do Tracto Vocal (82).

Da Figura 10.3 é possível inferir que o tracto vocal pode ser modelado como um conjunto de tubos acústicos com diferentes áreas e secções. Pode-se mesmo fazer uma analogia deste modelo com circuitos RLC, onde se produzem efeitos de ressonâncias e anti-ressonâncias.

O tracto vocal pode ser dividido em quatro zonas principais onde se efectuam as constrições: *zona velar*, *zona linguo-palatal*, *zona linguo-alveolar* e *zona labial*.

O seguinte modelo (Figura 10.4) é uma analogia com o processo de produção de fala, do ponto de vista da engenharia:

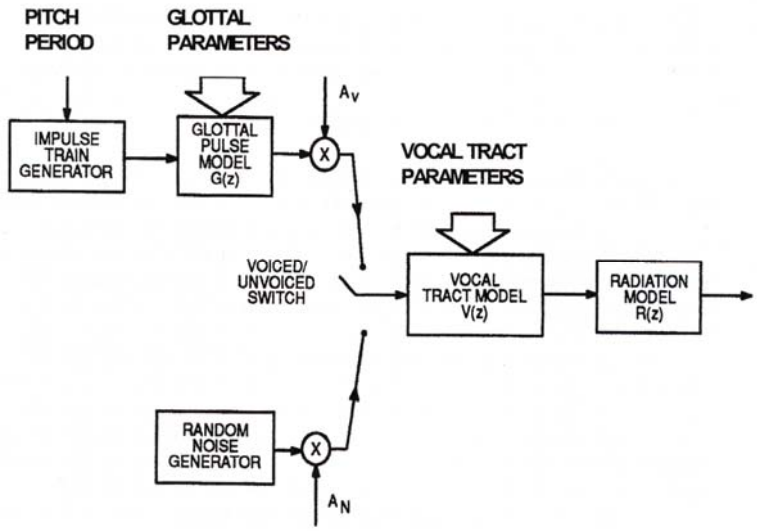


Figura 10.4 - Modelo de produção de fala. (83)

Neste modelo, o som (vozeado ou não vozeado) é controlado através de impulsos glotais $G(z)$ ou ruído branco. Após a selecção do tipo de voz, o tracto vocal é simulado com um filtro $V(z)$ que introduz ressonâncias e não-ressonâncias. À saída do tracto vocal é introduzido um filtro de radiação labial $R(z)$ que procura simular a acção que os lábios exercem sobre o fluxo de ar que sai do tracto vocal, modificando a sua pressão acústica.

10.1.5 Impulsos Glotais

Existem dois modelos principais para se modelar a resposta impulsional dos impulsos glotais que permitem a concretização de um sistema de produção de fala (como o que foi apresentado na secção anterior): *modelo de Rosenberg* e o *modelo de Rowden*.

Modelo de Rosenberg:

$$g(n)=\begin{cases}\dfrac{1}{2}\left[1-\cos\left(\dfrac{n\pi}{N_1}\right)\right], 0\le n\le N_1\\ \cos\left[\dfrac{\pi(n-N_1)}{2N_2}\right], N_1\le n\le N_1+N_2\\ 0, outros\end{cases} \quad (10.1)$$

Modelo de Rowden:

$$G(z) = \frac{-ae\ln(a)z^{-1}}{\left(1-az^{-1}\right)^2}, a = 0.9 \qquad (10.2)$$

Geralmente é usado o modelo de Rosenberg.

10.1.6 Modos de Produção da Fala

Os segmentos fonéticos, para além de se distinguirem pela presença ou ausência de vozeamento, são ainda diferenciados por classes que dependem do modo de articulação que o tracto vocal sofre para produzir o som. Assim sendo identificam-se os seguintes tipos de fala: vogais, ditongos, semi-vogais, semi-consoantes, paragens, nasais, líquidas, fricativas, aspiradas e oclusivas. Estas classes e as respectivas constrições articulatórias descritas, podem ser encontradas na definição fonética de uma tabela IPA[97].

No entanto os caracteres da tabela IPA que identificam um determinado fonema possuem símbolos impossíveis de escrever ou imprimir, por isso usa-se geralmente a representação fonética SAMPA[98], exemplificada na Figura 10.5.

Transcrição em SAMPA	
Análise e Síntese da Fala	6naliz@ j sint@z@ “d6 “fal6

Figura 10.5 – Exemplo de codificação SAMPA.

10.1.7 Formantes

As diferentes zonas do tracto vocal dividem-no num conjunto de câmaras de filtros, no qual o espectro do sinal é amplificado ou atenuado. As zonas amplificadas correspondem às zonas de ressonâncias definidas por: uma frequência central, uma largura de banda e uma determinada energia. Essa frequência central é denominada de *formante*.

Os formantes são normalmente representados por F1, F2, F3, ..., começando pela frequência mais baixa. É possível observar-se estas frequências a partir de um espectrograma.

[97] IPA – International Phonetic Alphabet - http://www2.arts.gla.ac.uk/IPA/ipa.html
[98] SAMPA – Speech Assessment Methods Phonetic Alphabet - http://www.phon.ucl.ac.uk/home/sampa

10.1.8 Audiologia e Psico-acústica

Para um investigador, que trabalhe na área de análise e síntese da fala, é importante ter alguns conceitos básicos acerca do funcionamento do ouvido humano, uma vez que este é o principal receptor de sinais de fala. A ciência que se dedica a identificar e a quantificar a percepção auditiva humana designa-se por *psico-acústica*.

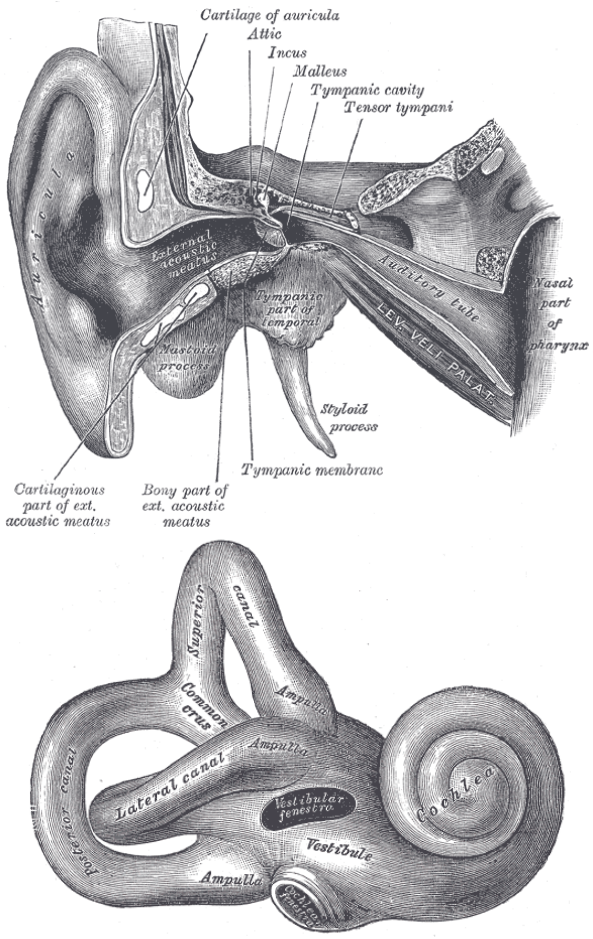


Figura 10.6 – Composição do ouvido humano. (130)

O ouvido humano compõe-se nos seguintes elementos:

- Ouvido externo: aurícula e canal auditivo externo.
- Ouvido médio: tímpano, cavidade timpânica, ossículos e tubo de Eustáquio.
- Ouvido interno: labirinto, órgão de corti (cóclea) e órgãos vestibulares.

A cóclea é sem dúvida um dos órgãos mais importantes, uma vez que funciona como um analisador espectral dos sinais de fala que recebe.

Limiares do ouvido humano

Através de um conjunto de experiências sensoriais auditivas, foi possível elaborar um gráfico que representa os limiares de audição do ouvido humano.

Observando a Figura 10.7, é possível concluir-se que: o ouvido humano é mais sensível para frequências acima de 4KHz, e a 20Hz existe uma perda de audição de 70dBs. Os 140dBs são considerados o limiar de dor do ouvido humano.

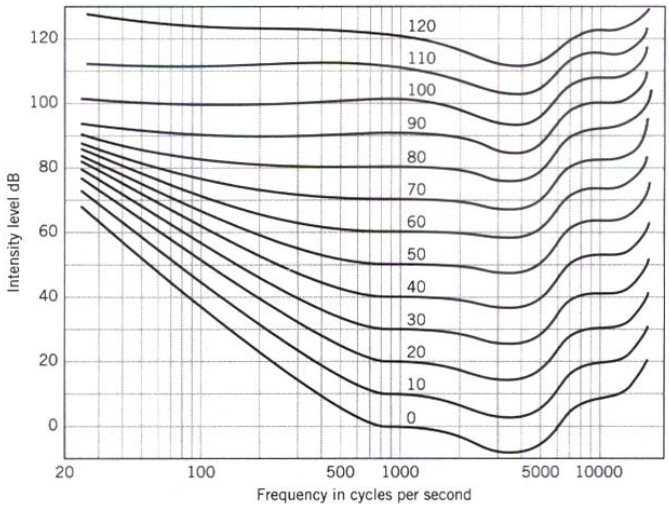


Figura 10.7 – Limiares de audição do ouvido humano. (81)

Bandas críticas

Já foi dito anteriormente que a cóclea é uma espécie de analisador espectral. Na realidade, todo o ouvido humano funciona com um banco de filtros, cada um com a sua banda passante. Um sinal deve ultrapassar uma determinada diferença ou limiar de detecção para que seja detectado dentro da banda passante de um filtro. A estas bandas de frequência dá-se o nome de *bandas críticas*.

No entanto, na vizinhança de tons fortes alguns sons são mascarados, isto é, não se tornam audíveis. Este conceito é extremamente importante e muitos algoritmos de codificação perceptual, tais como a codificação áudio *MPEG*[99] *layer 3* (MP3), baseiam-se em técnicas que aproveitam este efeito de máscara de determinadas frequências para comprimir ao máximo os ficheiros de áudio.

[99] MPEG – Moving Picture Experts Group.

10.2 Corpus Matemático – elaboração e gravação

Verificou-se que os trabalhos anteriormente apresentados, na introdução desta secção, cingiam os seus resultados num grupo restrito de expressões matemáticas. Assim sendo, foi proposta a elaboração de um *corpus*[100] para gravação e análise dos sinais de fala, onde se procurou incluir um conjunto vasto de fórmulas matemáticas agrupadas em nove categorias suportadas pelo AudioMath, sendo elas:

- Álgebra Simples
- Fracções
- Raízes
- Potências
- Trigonometria
- Matrizes e Sistemas de (in)equações
- Integrais
- Derivadas
- Somatórios

A preparação deste corpus matemático foi baseada na pesquisa de diverso material escolar do ensino secundário, assim como nas fichas e exames de Análise Matemática I e II leccionadas nos dois primeiros anos da Licenciatura em Engenharia Electrotécnica e Computadores da Faculdade de Engenharia da Universidade do Porto. Foram utilizados, para cada categoria, os seguintes critérios de selecção:

- A existência de um leque heterogéneo de fórmulas composto por expressões simples e complexas, de combinações variadas.
- A composição das fórmulas deve exprimir cardinalidade diferente no que diz respeito às sub expressões que as compõem.

Por exemplo, na categoria "Fracções" foram incluídas as seguintes expressões:

$$\frac{1}{2} \tag{10.3}$$

[100] O corpus matemático elaborado pode ser encontrado no ANEXO E.

$$\frac{1}{2}+\frac{1}{1+\frac{x}{3}}=\frac{1+\frac{x}{3}}{2} \tag{10.4}$$

De entre mais de 200 expressões matemáticas recolhidas foram seleccionadas 74. Após a selecção do corpus matemático optou-se por obter a sua forma escrita por extenso através da utilização do AudioMathENGINE. A aplicação deste método de conversão garante que os resultados obtidos, durante a análise de sinal, podem ser aplicados posteriormente no conversor AudioMathENGINE.

Durante a gravação foi utilizado o seguinte equipamento:

- Placa de aquisição e reprodução áudio externa – *Sound Blaster Audigy2 NX 24 bits Advanced HD* [101]:
    - Relação Sinal-Ruído (SNR) de 102 dB.
    - Conversores de 24bits com resoluções até 96KHz.
- Microfone SONY F-V320[102]:
    - Microfone dinâmico.
    - Unidireccional.
    - Resposta em frequência: 80-13000 Hz.
    - Impedância de saída: 600 ohms ± 30% a 1 KHz.
    - Sensibilidade: -53 ± 3 dB (0 dB = 1 V/Pa, 1000 Hz)
    - Tripé
- Programas informáticos utilizados:
    - *Creative Media Source 2.0*[103] - utilizado como centro de controlo do sinal de entrada do microfone.
    - *GoldWave 5.0*[104] - utilizado para pré-tratamento dos sinais de fala.
    - *Praat 4.3.04*[105] - utilizado para análise dos sinais de fala, extracção de informação sobre as pausas e contorno F0.

[101] http://www.creative.com/products/product.asp?category=1&subcategory=204&product=9103&nav=2
[102] http://www.sonystyle.com/is-bin/INTERSHOP.enfinity/eCS/Store/en/-/USD/SY_DisplayProductInformation-Start?CategoryName=hav_HomeTheaterDepartmentAccessories_Microphones&ProductSKU=FV320&TabName=specs&var2=
[103] http://www.creative.com/products/product.asp?category=1&subcategory=204&product=9103&nav=3
[104] http://www.goldwave.com/features.php
[105] http://www.fon.hum.uva.nl/praat/

Os ficheiros de áudio foram gravados usando as seguintes opções:

- Mono
- 44100 Hz
- 16 bits

10.3 Análise de sinais de fala – pausas e contorno F0

A análise dos sinais de fala foi concretizada numa abordagem de etiquetagem de cada um dos sinais, através do programa *Praat*. Este método de etiquetagem consiste num processo de marcação dos limites entre palavras no texto gravado. O resultado é especialmente elucidativo no que diz respeito ao uso de pausas durante a gravação. Também a informação acerca do contorno da frequência fundamental (F0) pode ser adquirida usando o *Praat*. A apresentação visual do contorno, em conjunto com as marcações dos limites de palavras, permite-nos inferir padrões prosódicos acerca do comportamento das frequências e de possíveis pausas no texto (Figura 10.8).

Uma vez identificadas determinadas palavras chave (por exemplo: "raiz quadrada de" ou "fim de radicando") numa expressão matemática, estas são caracterizadas no contexto da própria expressão, isto é, a posição onde elas se encontram. Serão apresentados resultados que demonstram inequivocamente que o comportamento prosódico varia consoante o contexto da expressão.

Nas secções seguintes são apresentados resultados provenientes do estudo realizado no âmbito desta dissertação.

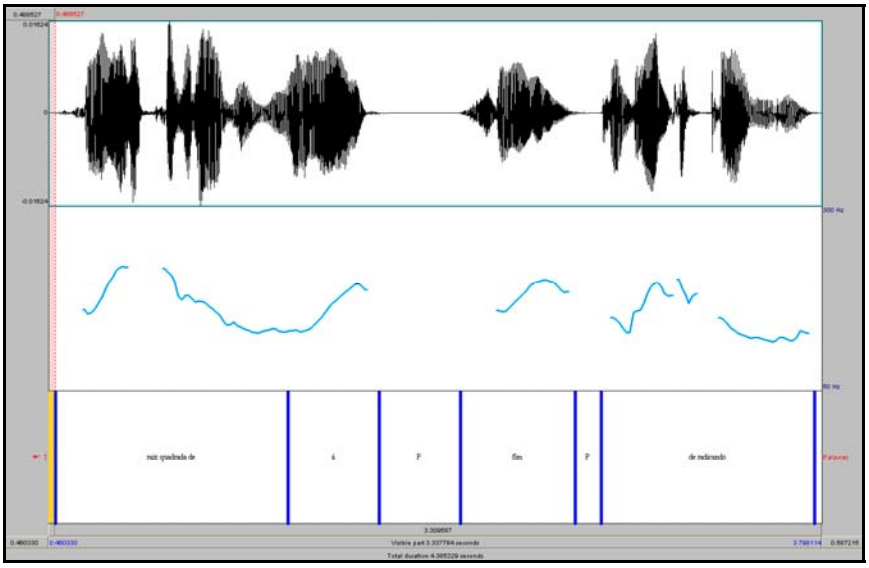


Figura 10.8 - Sinal de áudio etiquetado no Praat.

10.3.1 Álgebra Simples

Foram analisadas 11 amostras de sinais de áudio contendo expressões matemáticas com operadores algébricos, tais como adição, subtracção e multiplicação.

As expressões matemáticas possuem a seguinte estrutura: `parcela operador parcela operador parcela operador ...`

Da análise de pausas, concluiu-se o seguinte:

- É vulgar o uso de pausas antes e depois de um operador matemático.
- A pausa que antecede um operador varia consoante o tamanho da sub expressão matemática que antecede o operador:
    - Se a expressão é longa (mais do que uma palavra) então a pausa é de cerca de 1 segundo.
    - Se a expressão é curta (uma palavra) então a pausa é de cerca de 0,5 segundos (metade da pausa anterior).
    - Exemplo: “cinco vírgula quatro” + pausa(1s) + “mais” + pausa + “três” + pausa(0,5s) + “mais” + ...
- A pausa que sucede um operador varia consoante o tamanho do operador:
    - Se operador é longo então a pausa é de cerca de 1 segundo.
    - Se o operador é curto então a pausa é de cerca de 0,5 segundos.
    - Exemplo: “abrir parêntesis” + pausa(1s) + “chis” + pausa(0,5s) + “mais” + ...

Da análise sobre o contorno F0, concluiu-se o seguinte:

- Sempre que é aplicado um operador verifica-se a existência de um padrão F0 do tipo conclusivo.
- As parcelas entre os operadores possuem um padrão F0 do tipo conclusivo seguido de um padrão suspensivo. Excepto na última parcela, onde o padrão é sempre conclusivo.

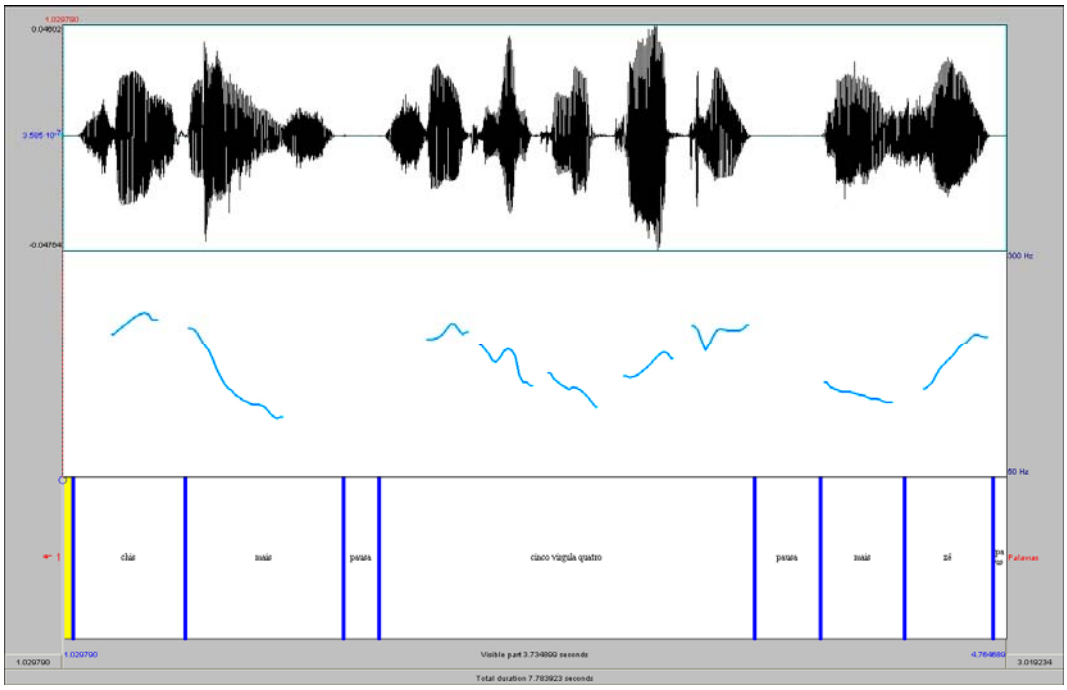


Figura 10.9 – Sinal de fala de álgebra simples.

10.3.2 Fracções

Foram analisadas 7 amostras de sinais de áudio contendo fracções com diversos níveis de complexidade. Foram identificadas as seguintes palavras-chave:

- "inicio de fracção"
- "numerador"
- "a dividir por"
- "denominador"
- "fim de fracção"

As expressões matemáticas possuem a seguinte estrutura: `inicio de fracção numerador <expressão> a dividir por denominador <expressão> fim de fracção`.

Da análise de pausas, concluiu-se o seguinte:

- A palavra-chave "inicio de fracção" é sempre sucedida de uma pausa com cerca de 1,2 segundos.
- A palavra-chave "numerador" é sempre sucedida de uma pausa com cerca de 0,8 segundos.

- Na maioria dos casos dá-se o seguinte padrão: “a dividir por” + pausa + “denominador” + pausa. Nestes casos, a primeira pausa é de cerca de 0,5 segundos e a segunda pausa é de cerca de 1 segundo.
- A palavra-chave “a dividir por” é tipicamente precedida por uma pausa. Esta pausa muda consoante o contexto:
    - Se antes existiu uma palavra-chave do tipo “fim de fracção” ou “fim de radicando”, a pausa é de cerca de 1 segundo.
    - Se antes existiu uma palavra normal, a pausa é de 0,5 segundos.

Da análise sobre o contorno F0, concluiu-se o seguinte:

- As palavras-chave “inicio de fracção”, “a dividir por” e “denominador” são caracterizadas por um padrão F0 suspensivo seguido de um padrão F0 conclusivo.
- As palavras-chave “numerador” e “fim de fracção” são caracterizadas por um padrão F0 do tipo conclusivo.

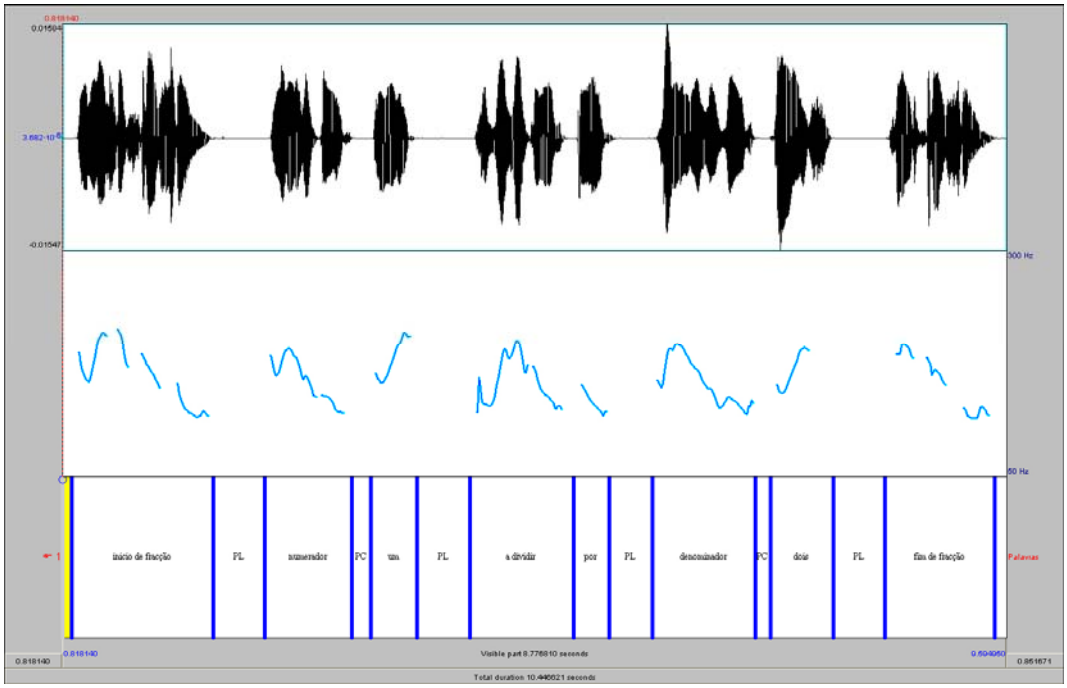


Figura 10.10 – Sinal de fala de uma fracção.

10.3.3 Raízes

Foram analisadas 9 amostras de sinais de áudio contendo raízes com diversos níveis de complexidade. Foram identificadas as seguintes palavras-chave:

- "raiz"
- "de índice"
- "fim de índice"
- "e base"
- "fim de radicando"

As expressões matemáticas possuem a seguinte estrutura: `raiz quadrada de <expressão> fim de radicando` ou `raiz de índice <expressão> e base <expressão> fim de radicando`.

Da análise de pausas, concluiu-se o seguinte:

- Tipicamente só existem pausas a seguir à palavra-chave "raiz" quando se trata de uma raiz com índices complexos. A pausa a seguir à palavra "raiz" e antes das palavras "de índice" é de cerca de 1 segundo. Também nestes casos, e dependendo do contexto, existem pausas antes de "e base". Se antecedendo estas pausas existe uma sub expressão complexa (mais do que uma palavra), então a pausa tem cerca de 1,1 segundos. Caso contrário tem cerca de 0,7 segundos.
- Depois de "e base" existe sempre uma pausa com cerca de 0,7 segundos.
- Entre as palavras "fim de radicando" existe esporadicamente uma pausa com cerca de 0,5 segundos.
- Antes de um "fim de radicando" existe sempre uma pausa. Novamente a questão do contexto é relevante: se a expressão é complexa, então a pausa tem cerca de 1 segundo; se a expressão é simples, então a pausa tem cerca de 0,3 segundos.

Da análise sobre o contorno F0, conclui-se o seguinte:

- Caso seja uma raiz quadrada ou cúbica:
    - "Raiz quadrada de" possui um padrão F0 do tipo suspensivo.
    - "Fim de radicando" possui um padrão F0 do tipo conclusivo.

- Caso seja uma raiz de índice complexo:
    - “raiz” possui um contorno suspensivo.
    - “de índice”, “e base” e “fim de radicando” possuem um contorno F0 do tipo conclusivo.

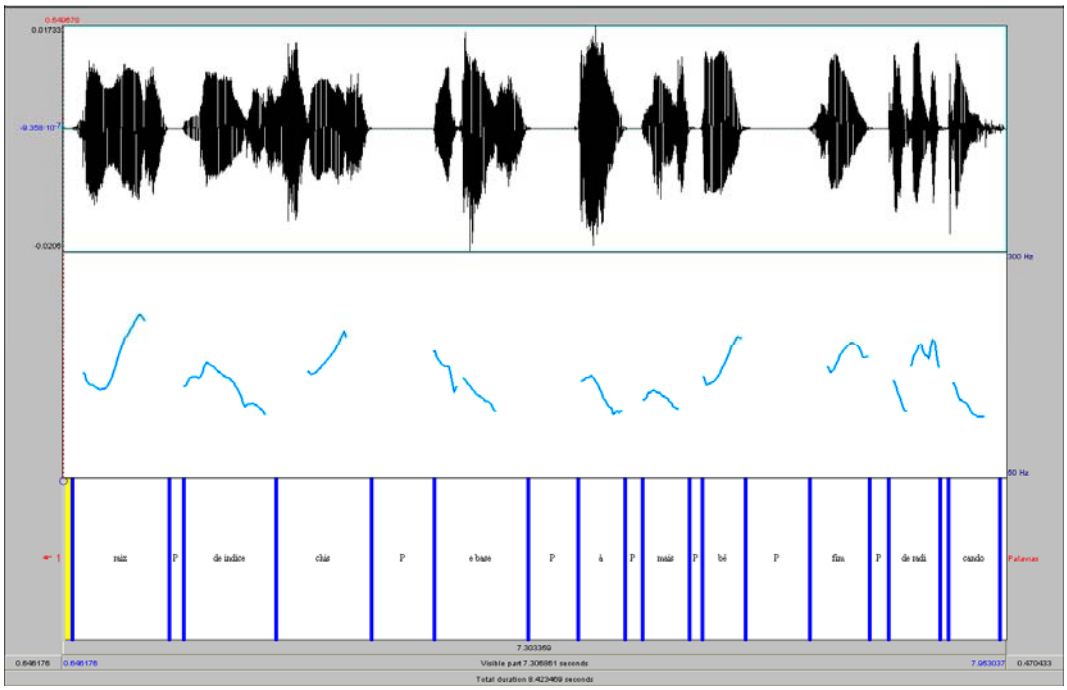


Figura 10.11 – Sinal de fala de uma raiz.

10.3.4 Potências

Foram analisadas 12 amostras de sinais de áudio contendo potências com diversos níveis de complexidade. Foram identificadas as seguintes palavras-chave:

- “elevado ao expoente”
- “fim de expoente”

As expressões matemáticas possuem a seguinte estrutura: `<expressão> ao quadrado` ou `<expressão> ao cubo` ou `<expressão> elevado ao expoente <expressão> fim de expoente.`

Da análise de pausas, concluiu-se o seguinte:

- Tipicamente existe uma pausa antes de “elevado ao expoente”. Se a expressão que precede essa pausa é complexa, a pausa é de cerca de 1 segundo. Caso contrário, a pausa é de cerca de 0,6 segundos.
- Existe quase sempre uma pausa a seguir a “elevado ao expoente” com cerca de 1 segundo.
- Antes de “fim de expoente” existe uma pausa com cerca de 0,6 segundos.
- Depois do “fim de expoente”, e caso a expressão continue, existe uma pausa de 1 segundo. Caso particular é quando a seguir a um “fim de expoente” segue-se outro “fim de expoente”; nesses casos a pausa é de 0,46 segundos.

Da análise sobre o contorno F0, conclui-se o seguinte:

- “ao quadrado” ou “ao cubo” possuem padrões F0 do tipo suspensivo.
- “elevado ao expoente” possui um padrão F0 conclusivo seguido de um padrão suspensivo e outro conclusivo.
- “fim de expoente” é caracterizado por um contorno do tipo conclusivo.

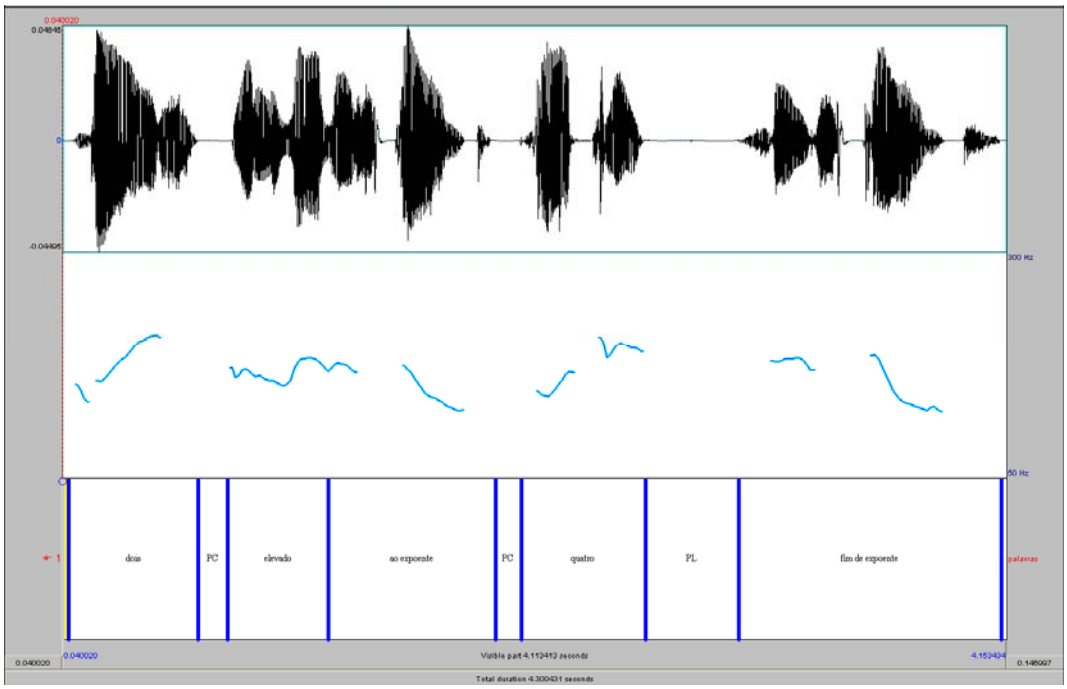


Figura 10.12 – Sinal de fala de uma potência.

10.3.5 Trigonometria

Foram analisadas 7 amostras de sinais de áudio contendo expressões com elementos de trigonometria de diversos níveis de complexidade.

As expressões matemáticas possuem a seguinte estrutura: `<elemento trigonométrico>` de `<expressão>` ou `<elemento trigonométrico>` `<expressão>` de `<expressão>`.

Da análise de pausas, concluiu-se o seguinte:

- Tipicamente a seguir a um elemento trigonométrico mais a palavra "de", existe sempre uma pausa com cerca de 1 segundo. Exceptuando nos casos em que o elemento trigonométrico é base de uma potência; aí a pausa ocorre antes do "de".

Da análise sobre o contorno F0, conclui-se o seguinte:

- Os elementos trigonométricos possuem um padrão do tipo conclusivo.

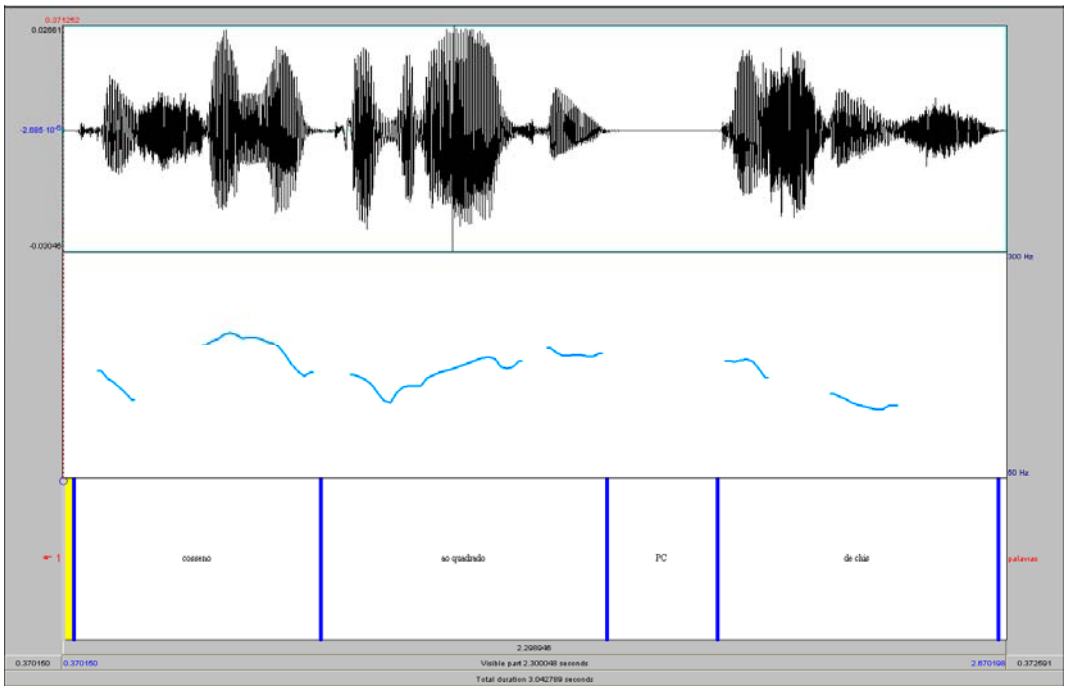


Figura 10.13 – Sinal de fala de trigonometria.

10.3.6 Matrizes ou sistemas de (in)equações

Foram analisadas 4 amostras de sinais de áudio representando, cada um deles, um tipo diferente de matrizes ou sistemas de (in)equações. Foram identificadas as seguintes palavras-chave:

- "tabela ou matriz"
- "primeira|segunda|... linha"
- "primeira|segunda|... coluna"
- "fim de linha"
- "fim de coluna"
- "fim de tabela ou matriz"

As expressões matemáticas possuem uma estrutura que depende da dimensão da matriz. Assumindo, por exemplo, uma matriz com duas linhas e uma coluna, a estrutura seria: `tabela ou matriz primeira linha primeira coluna <expressão> fim de coluna fim de linha segunda linha primeira coluna <expressão> fim de coluna fim de linha fim de tabela ou matriz.`

Da análise de pausas, concluiu-se o seguinte:

- A palavra-chave "tabela ou matriz" é sucedida de uma pausa com cerca de 1.3 segundos.
- A palavra-chave "primeira|segunda|... linha" é seguida por uma pausa com cerca de 0,9 segundos.
- A palavra-chave "primeira|segunda|... coluna" é seguida por uma pausa com cerca de 1 segundo.
- Entre colunas existe uma pausa com cerca de 1 segundo.
- As palavras chave "fim de coluna" e "fim de linha" são seguidas de uma pausa de 0,5 segundos.

Da análise sobre o contorno F0, concluiu-se o seguinte:

- "tabela ou matriz", "primeira linha", "primeira coluna", "fim de linha", "fim de coluna" e "fim de tabela ou matriz" são caracterizadas por padrões do tipo suspensivo, seguidos de um contorno conclusivo.

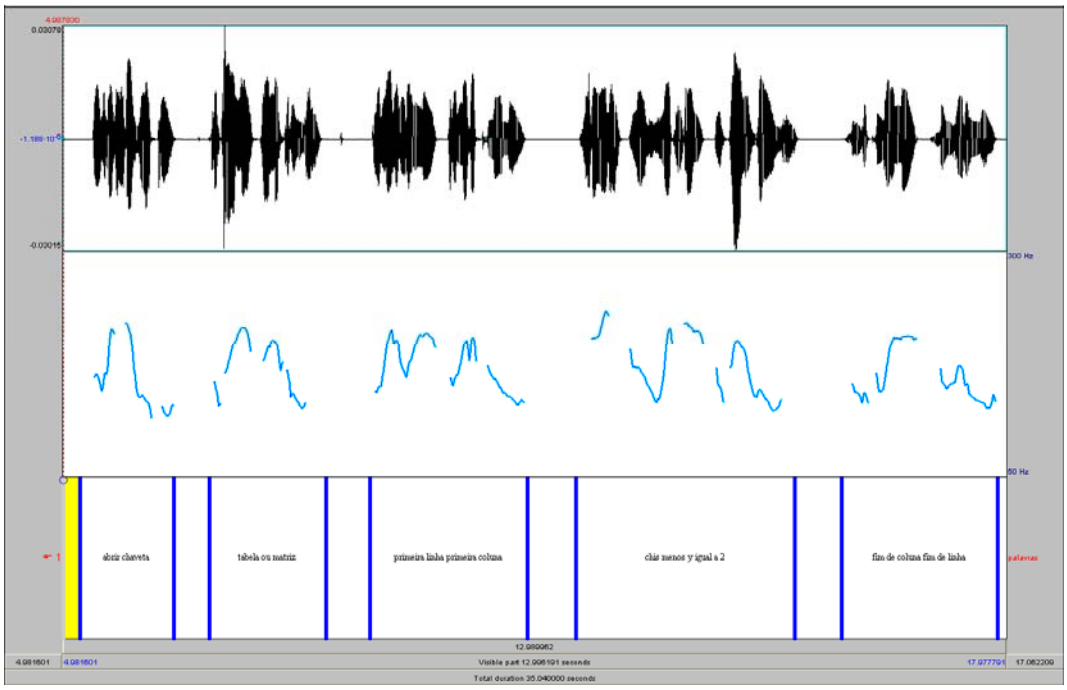


Figura 10.14 – Sinal de fala de um sistema de equações.

10.3.7 Integrais

Foram analisadas 8 amostras de sinais de áudio contendo expressões com integrais de diversos níveis de complexidade. Foram identificadas as seguintes palavras-chave:

- "integral"
- "limite inferior"
- "limite superior"
- "de"

As expressões matemáticas possuem a seguinte estrutura: `integral com limite inferior <expressão> e limite superior <expressão> de <expressão>` ou `integral de <expressão>`.

Da análise de pausas, concluiu-se o seguinte:

- Depois da palavra-chave "integral" existe sempre uma pausa de 1 segundo.
- Após "limite inferior" existe sempre uma pausa de 1 segundo.
- Depois de "limite superior" existe sempre uma pausa que varia com o contexto. Se antes tinha expressão singular, a pausa é de cerca de 0,5 segundos. Caso contrário a pausa é de cerca de 1 segundo.

- Após “limite superior” e antes de “de”, existem pausas que seguem as mesmas regras descritas anteriormente para a palavra-chave “limite superior”.
- Depois da palavra-chave “de” existe sempre uma pausa com cerca de 0,5 segundos.

Da análise sobre o contorno F0, concluiu-se o seguinte:

- “integral de” possui um contorno do tipo conclusivo.
- “integral” é caracterizado por um padrão do tipo conclusivo seguido de um contorno suspensivo.
- “com limite inferior”, “com limite superior” e “de” são caracterizados por um contorno do tipo conclusivo.

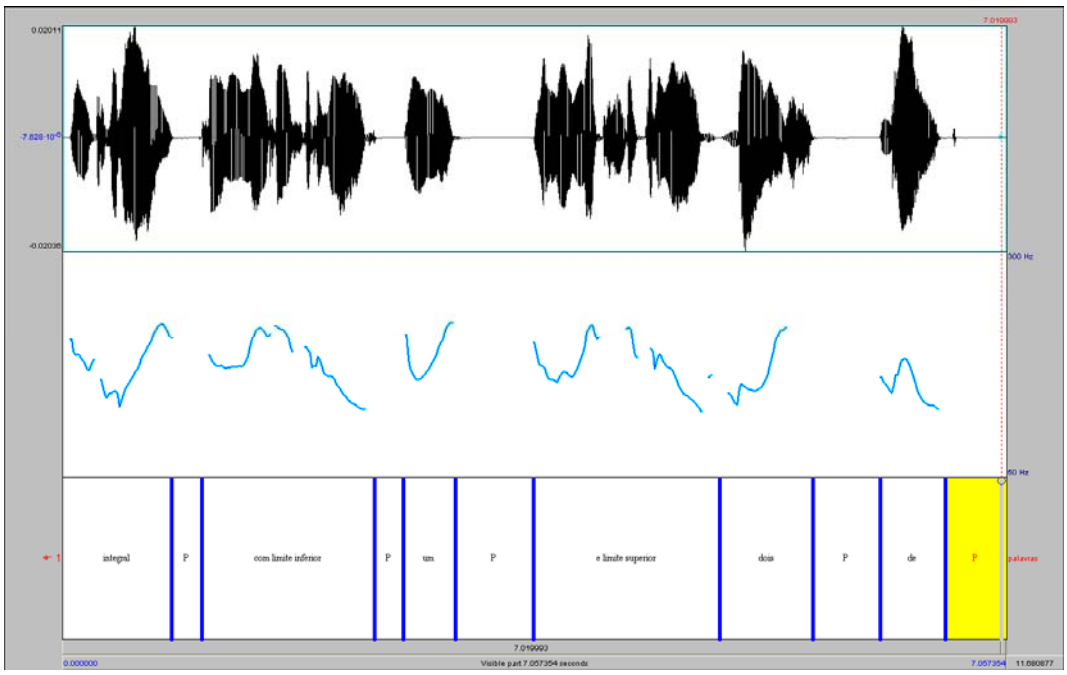


Figura 10.15 – Sinal de fala de um integral.

10.3.8 Derivadas

Foram analisadas 7 amostras de sinais de áudio, embora a conversão das derivadas seja algo deficiente no AudioMathENGINE. Em vez de serem convertidas como "derivada de" quando, por exemplo, a expressão y' é encontrada, o AudioMathENGINE converte-as, usando os seus algoritmos por defeito, em: "ípsilon linha". Este é um dos pontos a melhorar no futuro, nos algoritmos de análise, interpretação e conversão de expressões matemáticas.

Da análise de pausas, concluiu-se o seguinte:

- O uso da expressão "linha" não oferece quaisquer conclusões relevantes.

Da análise sobre o contorno F0, concluiu-se o seguinte:

- "linha" é caracterizada como tendo um padrão conclusivo seguido de um padrão suspensivo.

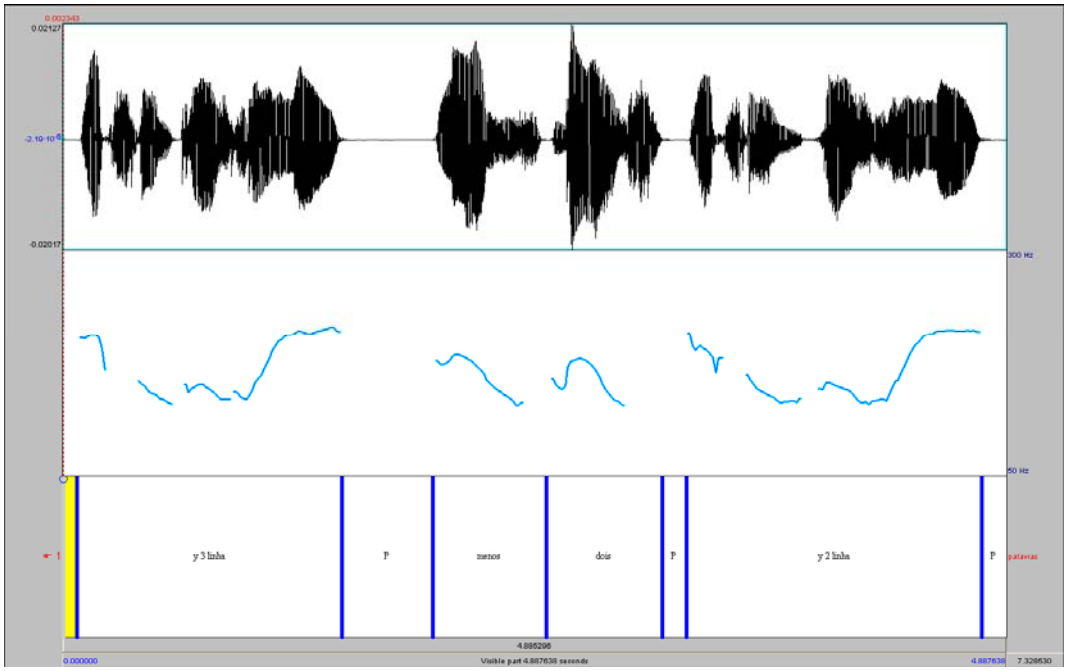


Figura 10.16 – Sinal de fala de uma derivada.

10.3.9 Somatórios

Foram analisadas 9 amostras de sinais de áudio contendo expressões com somatórios com diversos níveis de complexidade. Foram identificadas as seguintes palavras-chave:

- "somatório"
- "limite inferior"
- "limite superior"
- "de"
- "até"

As expressões matemáticas possuem a seguinte estrutura: `somatório com limite inferior <expressão> e limite superior <expressão> de <expressão>` ou `somatório de <expressão>` ou `somatório de <expressão> até <expressão> de <expressão>`.

Da análise de pausas, concluiu-se o seguinte:

- A seguir à palavra-chave "somatório" existe sempre uma pausa de cerca de 1 segundo.
- Após os índices existe uma pausa. Se os índices são expressões singulares, a pausa tem cerca de 0,4 segundos. Caso contrário a pausa tem cerca de 1 segundo.
- A seguir a um "somatório de" ou "limite superior <expressão> de" ou "até <expressão> de" existe sempre uma pausa com cerca de 0,5 segundos. Por vezes esta pausa é maior, caso não tenha existido pausa que sucede a "limite superior".

Da análise sobre o contorno F0, concluiu-se o seguinte:

- "somatório de" e "de" são caracterizadas por padrões F0 do tipo conclusivo.
- "somatório" possui um contorno do tipo conclusivo seguido de um padrão do tipo suspensivo.
- "limite inferior" e "limite superior" são o oposto do anterior. São caracterizados por um padrão do tipo conclusivo, seguido de um padrão do tipo suspensivo.

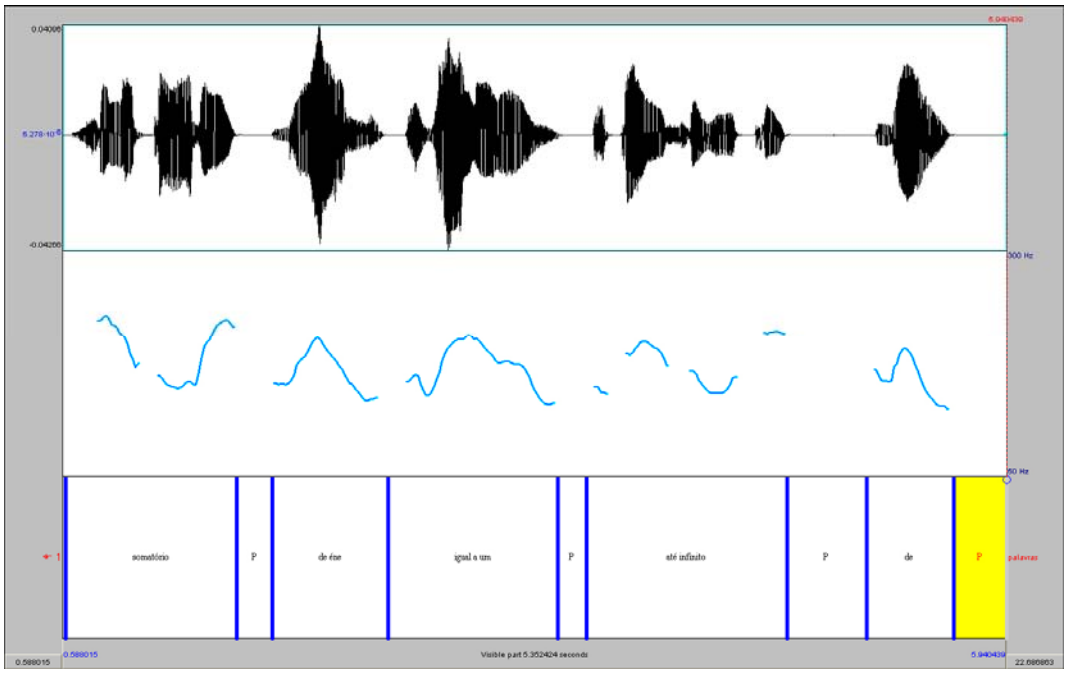


Figura 10.17 – Sinal de fala de um somatório.

11 Navegação em Expressões Matemáticas

A leitura de um texto literário difere da leitura de uma expressão matemática. A apresentação oral de informação complexa, como na matemática, exige a capacidade de navegação e iteração, como mecanismo de auxilio à memória, a forma como são percepcionados os conteúdos e até mesmo auxilio ao mecanismo de cognição que conduz à compreensão da expressão matemática.

Neste capítulo pretende-se apresentar sumariamente alguns conceitos básicos sobre percepção, memória e cognição, como introdução à importância da navegação na leitura da matemática, onde será apresentada uma solução desenvolvida no âmbito desta dissertação.

11.1 Considerações sobre Percepção, Cognição e Memória

Percepção

"O Homem não vê só a luz ou as cores, não ouve simplesmente os sons ou os ruídos, não sente só os odores; ao contrário, sabe que este perfume é de uma rosa, que aquele ruído é da buzina de um automóvel que passa na rua, que esta luz é da lâmpada do seu quarto. Quer dizer, refere sempre as suas sensações ao objecto que as provocou, interpretando-as e objectivando-as, e passa a representar o próprio objecto exterior. A sensação transforma-se deste modo em percepção." (131).

A percepção define-se como a interacção entre a mente e os cinco sentidos humanos: visão, tacto, olfacto, paladar e audição. Existem dois mecanismos distintos associados à percepção:

- *Antecipação perceptiva* – propriedade que permite a um individuo, através de informação sensorial, antecipar algo. No caso da visão esta propriedade possui um papel importante, uma vez que a aquisição de informação dá-se de uma forma muito rápida. Para os cegos ou amblíopes, a antecipação perceptiva é reduzida uma vez que a informação chega de forma mais lenta e demorada. No caso da audição, a antecipação perceptiva só tem expressão através de pistas sonoras. O facto da antecipação perceptiva ser um factor pouco utilizado por um cego faz com que este utilize com maior frequência a sua memória e outras competências cognitivas.

- *Antecipação cognitiva* – propriedade que permite a um individuo antecipar cognitivamente a percepção de determinados estímulos, recorrendo à sua memória.

Ambos os processos estão relacionados: quanto mais se usar a antecipação perceptiva, menos se utiliza a antecipação cognitiva e vice-versa. No caso dos cegos e amblíopes o uso de antecipação cognitiva é mais usado, embora requeira mais esforço.

Cognição

A cognição é tipicamente definida como o processo utilizado pelo Homem para a aquisição de conhecimento. Isto inclui a compreensão, capacidade de recordar, capacidade de argumentar, adquirir conhecimento e criar novas ideias.

O ser humano pode ser caracterizado como sendo um modelo de processamento de informação que adquire informação (estímulo), codifica-a, compara-a com experiências passadas, selecciona acções e executa-as reagindo ao estímulo (132), como exemplifica a Figura 11.1.

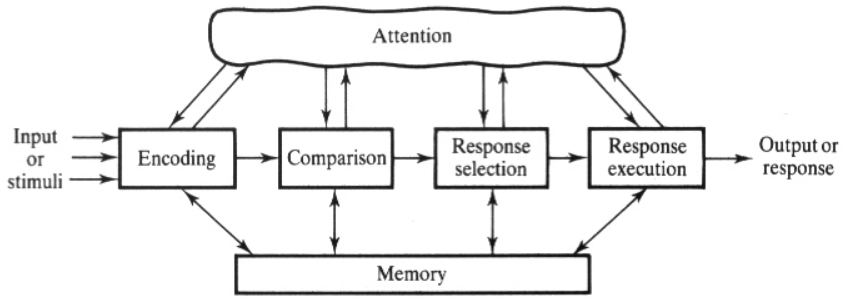


Figura 11.1 - Modelo de processamento de informação num ser humano. (132)

Memória

A memória pode ser representada sob a forma de um modelo de armazém de dados, composto por três áreas diferentes (Figura 11.2):

- *área sensorial* – guarda informação sensorial adquirida através da percepção por um curto espaço de tempo (uns milésimos de segundo).
- *área de memória de termo curto* ou *memória de trabalho* – guarda informação limitada por um curto período de tempo (uns segundos).
- *área de memória longa* – guarda informação por tempo indeterminado.

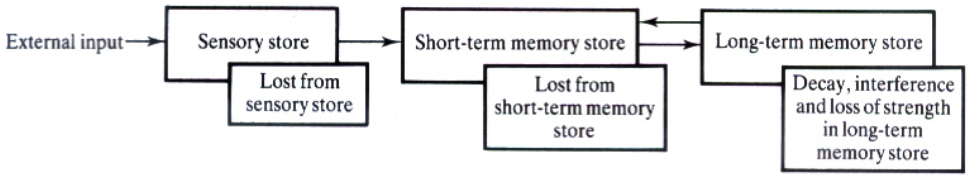


Figura 11.2 - Diferentes áreas que constituem a memória humana. (132)

Da informação que chega à área sensorial, apenas uma pequena parte é processada pela memória de trabalho. O ser humano é capaz de reter rapidamente e com eficácia o acesso a alguns itens na memória (número "mágico": 7 ± 2) (133); no entanto a capacidade de recordar estes itens decai rapidamente. Muitos dos erros em problemas formais, em áreas como a matemática, recorrem do facto de o ser humano procurar resolver os problemas com demasiados itens na memória de trabalho (ou memória de termo curto).

Conclui-se portanto que esta memória é limitada em quantidade e tempo. O tempo de retenção de dados em memória diminui à medida que a necessidade de guardar mais dados em memória aumenta (Figura 11.3).

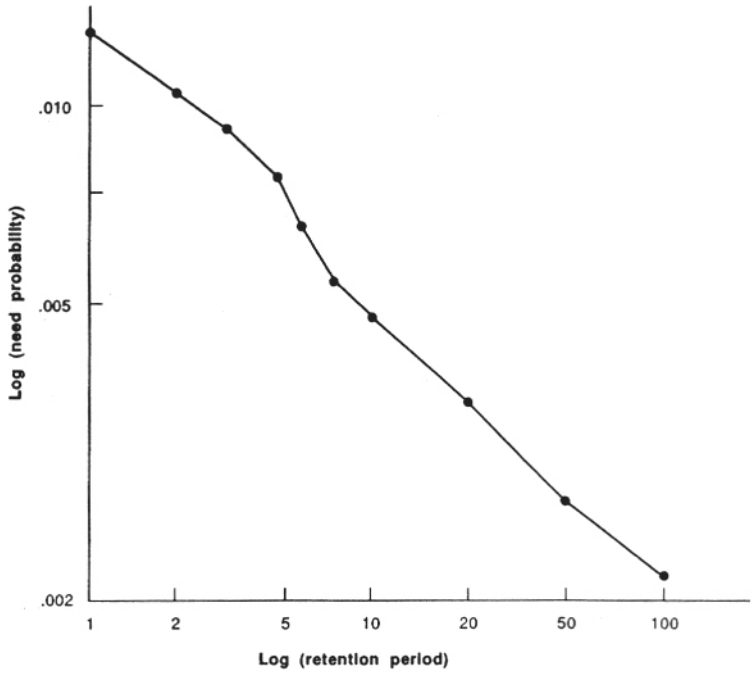


Figura 11.3 - Necessidade de recordar *versus* o tempo de retenção. (134)

Existem diversos fenómenos associados aos processos ou mecanismos de memória, que ajudam o ser humano a recordar (134)(135), tais como:

- Efeito de reconhecimento de palavras em memória;
- Fenómeno de *fan effect*;
- Aprendizagem e prática.

O efeito de reconhecimento de palavras em memória pode ser modelado pela probabilidade:

$$\frac{n(word \mid word\ in\ context)}{n(word)}$$

Figura 11.4 - Probabilidade de reconhecimento de palavras em memória. (134)

Por exemplo, a utilização de palavras chave nas expressões matemáticas pode beneficiar deste efeito de reconhecimento. Este pode ser mais rápido se, junto à palavra que se procura reconhecer, for introduzida uma palavra associada. Por exemplo, antes da leitura de uma fórmula matemática pode-se falar "*Fórmula:*"; ou antes de um texto usar a palavra "*Texto:*".[106]

Existe ainda um outro fenómeno designado *fan effect* (134), que demonstra que o reconhecimento e a recordação de um determinado conceito é mais rápido quando existem mais factos a reforçarem-no. Quantos mais conceitos existirem a reforçar, mais difícil é a recusa perante a evidência. O AudioMathENGINE faz uso deste fenómeno ao inserir comentários descritivos que reforçam a estrutura da expressão matemática em causa. Por exemplo: numa raiz – "*raiz quadrada de*" + "*fim de radicando*"; ou numa fracção – "*inicio de fracção*" + "*numerador:*" + "*denominador*:" + "*fim de fracção*". No entanto, um efeito antagónico é também detectado – *negative fan effect* (134) – que consiste na teoria de que o óbvio pode por vezes implicar a recusa de uma ideia ou conceito.

[106] O AudioMathENGINE faz uso deste efeito e aplica-o no seu conceito de navegação como se verificará na secção 11.3.

Na Figura 11.5 observa-se o resultado de uma experiência de (135), onde se demonstra o tempo de reacção em relação ao número de dias praticados. São comparados dois tipos de frases:

- frases *no fan* – frases com palavras que ocorreram apenas uma vez em todas as frases.
- frases *fan* – frases com palavras que ocorreram mais do que uma vez nas diversas frases.

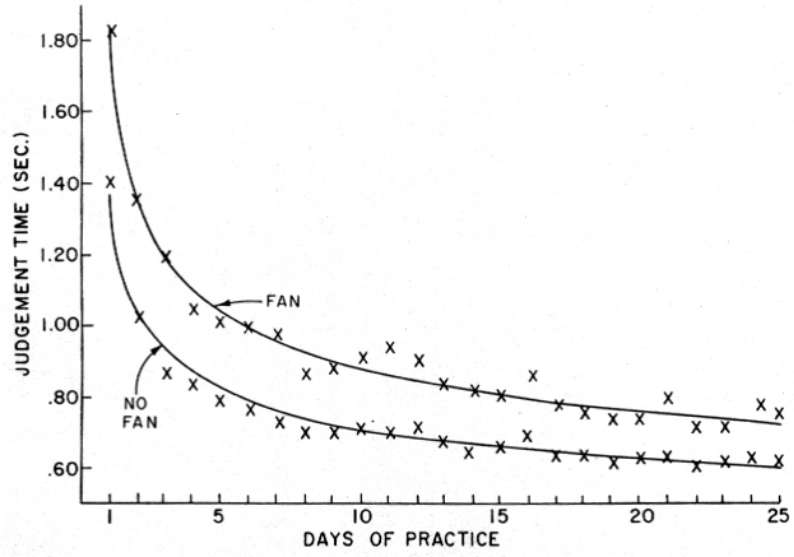


Figura 11.5 - Influência da prática no reconhecimento de conceitos. (135)

Como se pode verificar, à medida que a prática vai aumentando, o ser humano reage mais rapidamente no reconhecimento das palavras.

11.2 Importância da navegação na matemática

O ser humano possui um mecanismo particular de adquirir conhecimento a partir da leitura de um texto. Sabe-se que as pessoas lêem um texto letra a letra começando pelo símbolo mais à esquerda numa linha de texto (se for escrito numa língua cuja leitura se faça da esquerda para a direita); ou pelo símbolo mais à direita (se estiver escrito numa língua cuja leitura se faça da direita para a esquerda). Contudo, a leitura não é directa e sempre seguida. É frequente a releitura de trechos do texto que já passaram para co-relacionar com a informação lida no ponto corrente. (136)

No entanto existem diversas diferenças entre a leitura de um documento com conteúdos matemáticos e a leitura de um documento de foro não técnico (137):

- Enquanto que a leitura de um texto não técnico possui um sentido de leitura bem definido, nalgumas partes de uma expressão matemática o sentido natural de leitura pode ser feito em qualquer sentido.
- A matemática é uma linguagem bidimensional, ao contrário dos textos literários que são unidimensionais.
- Em quase todas as línguas, primeiro aprende-se a falar e posteriormente a escrever, na matemática isto é feito tipicamente ao contrário: a notação vem primeiro e a leitura depois.

A Tabela 11.1 representa as conclusões de um estudo (138) que compara o movimento dos olhos durante a leitura de documentos de ficção e documentos matemáticos.

Documento	**Duração de fixação (ms)**	**Comprimento de saltos[107] (mm)**	**Regressões**	**Palavras por minuto**
Ficção	202	9.2	3	365
Matemática	254	7.3	18	243

Tabela 11.1 – Movimento dos olhos durante leitura de ficção e matemática.

Assim sendo, para o ser humano sem necessidades especiais no campo da visão, compreender uma expressão matemática requer diversas releituras de sub expressões e saltos lógicos entre os mesmos (138)(64). Por exemplo:

$$\sqrt{\frac{a+\frac{b}{c}}{d\times e}}$$

Primeira "passagem": "*é uma raiz quadrada*"

Segunda "passagem": "*tem uma fracção dentro da raiz*"

Terceira "passagem": "$a+b/c$ *no numerador e* $d\times e$ *no denominador*"

Quarta "passagem": **...**

Figura 11.6 - Interpretação de uma expressão matemática. (64)

[107] O comprimento dos saltos efectuados pelos olhos entre um ponto e outro durante a leitura, é denominado na literatura inglesa por *saccade*.

No entanto, o processo exemplificado na Figura 11.6 pode ser uma tarefa demasiado complexa ou mesmo impossível, para os cegos ou amblíopes, uma vez que o nosso cérebro involuntariamente realiza este tipo de iterações sobre a notação visual da expressão matemática. Em (139) é afirmado que estas notações visuais funcionam como memória externa ou auxiliar.

Por outro lado, a leitura completa e seguida de uma fórmula matemática não apresenta uma solução prática devido a restrições de memória e percepção[108]. Assim sendo, a capacidade de navegação dentro de uma expressão matemática torna-se essencial para "simular" a capacidade de iteração que o cérebro executa sobre a expressão matemática a fim de a compreender na totalidade.

Em (49) discute-se a necessidade da navegação devido a um fenómeno onde os utilizadores se sentem perdidos por não conhecerem a estrutura e a orientação da informação que procuram adquirir. Fenómeno esse que atinge proporções maiores quando o utilizador é cego, uma vez que estes apenas se conseguem concentrar numa pequena janela de informação, onde o controlo é difícil e impossível sem os mecanismos de navegação.

Conclui-se, portanto, que a navegação é um mecanismo essencial na interpretação áudio de uma expressão matemática.

11.3 Navegação no AudioMath

A leitura e interpretação de expressões matemáticas complexas só é possível de modo eficaz quando existem mecanismos de navegação implementados. Por esse motivo, foi projectado no desenvolvimento do trabalho desta dissertação uma série de propriedades que permitem a navegação estruturada de uma expressão matemática.

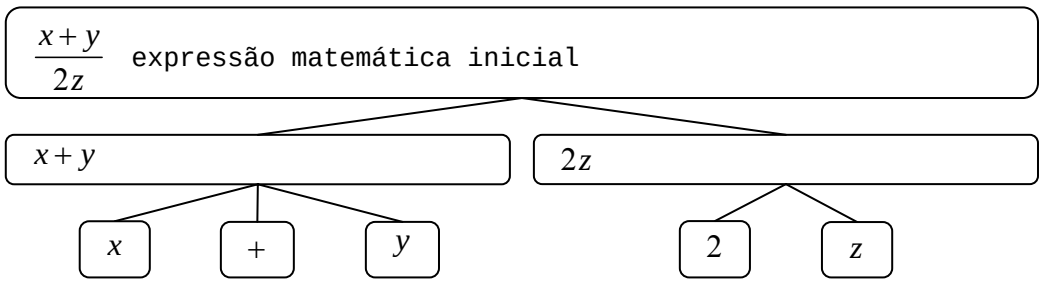


Figura 11.7 - Exemplo da árvore de navegação de uma expressão matemática.

[108] Ver secção 11.1

A solução implementada baseou-se no conceito discutido por (46), onde uma expressão matemática complexa pode ser decomposta em expressões matemáticas mais simples, construindo assim uma árvore de estruturas (Figura 11.7) onde a navegação se desenrola. A questão da navegação foi abordada em duas áreas distintas:

- *Identificação dos pontos de navegação e construção da árvore* – concretizado no desenvolvimento do AudioMathENGINE;
- *Interacção com utilizador* – desenvolvido no AudioMathGUI.

A identificação dos pontos de navegação foi desenvolvida no módulo de análise, conversão e interpretação de expressões matemáticas do AudioMathENGINE, enquanto que a árvore de navegação foi desenvolvida num módulo de navegação do AudioMathENGINE (Figura 11.8).

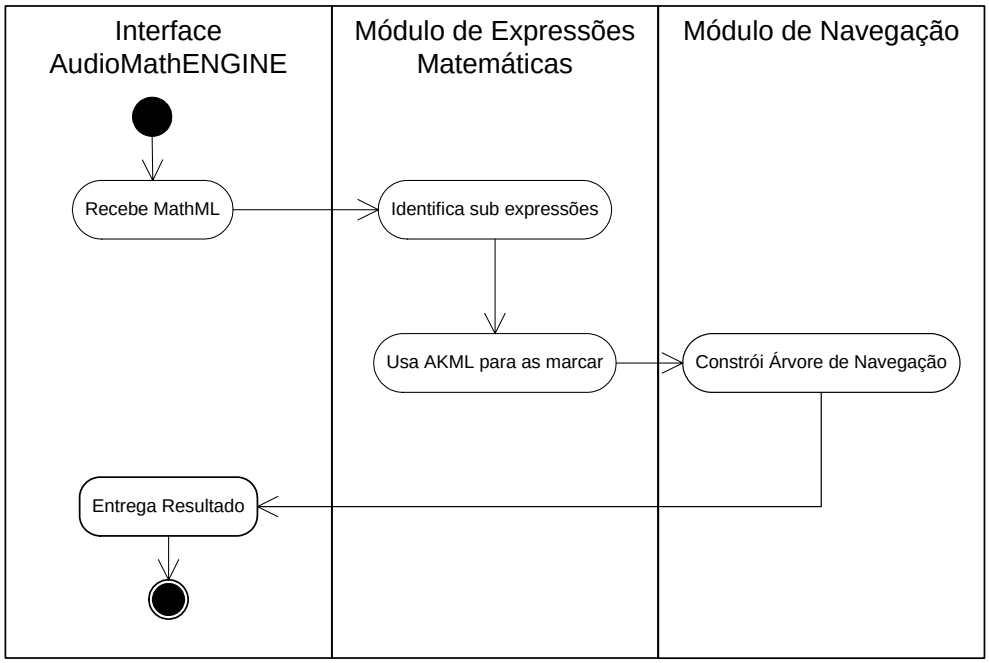


Figura 11.8 - Navegação no AudioMathENGINE.

Uma fórmula matemática contém operadores (Figura 11.9). É através destes que o módulo de expressões matemáticas detecta as diversas sub expressões que existem na expressão matemática em causa. Uma vez detectadas e convertidas, estas são marcadas com a etiqueta de marcação AKML[109] `<submat></submat>` que identifica os pontos de navegação.

[109] Ver secção 7.2.

Como resultado final obtém-se uma pré-estrutura de navegação onde as sub expressões que compõem a expressão matemática final encontram-se evidenciadas (Figura 11.10).

Entrada de MathML

$\sqrt{a^{2+b^2}+b^2}$

```
<math>
    <mrow>
      <msqrt>
        <mrow>
          <msup>
            <mi>a</mi>
            <mrow>
              <mn>2</mn>
              <mo>+</mo>
              <msup><mi>b</mi><mn>2</mn></msup>
            </mrow>
          </msup>
          <mo>+</mo>
          <msup><mi>b</mi><mn>2</mn></msup>
        </mrow>
      </msqrt>
    </mrow>
</math>
```

Figura 11.9 - Exemplo de entrada MathML para navegação.

Sub expressões marcadas identificadas com AKML

```
<mat>
  <submat>
     <submat> raiz quadrada de </submat>
     <submat>
       <submat>
          <submat> á </submat>
          <submat> elevado ao expoente: </submat>
          <submat>
             <submat> <num>dois</num> </submat>
             <submat> mais </submat>
             <submat>
                 <submat> bê </submat>
                 <submat> ao quadrado </submat>
             </submat>
          </submat>
          <submat> fim de expoente </submat>
       </submat>
       <submat> mais </submat>
       <submat>
          <submat> bê </submat>
          <submat> ao quadrado </submat>
       </submat>
     </submat>
     <submat> fim de radicando </submat>
  </submat>
</mat>
```

Figura 11.10 - Exemplo de sub expressões com AKML para navegação.

No entanto a apresentação de um resultado como na Figura 11.10 implicaria que um utilizador tivesse de navegar em todos os detalhes de uma fórmula matemática, mesmo que assim não o desejasse. Para além disso, existe apenas um único nível de navegação detalhado, o que não é desejável nem o pretendido. Por esse motivo foi implementada a construção da árvore de navegação, cujo objectivo se prende com a identificação dos vários níveis de navegação de uma fórmula matemática, desde o nível mais básico ao nível mais detalhado. Pretende-se com este método, dar a escolha ao utilizador de querer ou não detalhar a informação contida na sub expressão matemática.

Árvore de Navegação

```
<mat value="Fórmula:">
  <node value=" raiz quadrada de á elevado ao expoente: dois mais bê
ao quadrado fim de expoente mais bê ao quadrado fim de radicando ">
     <node value=" raiz quadrada de "></node>
     <node value=" á elevado ao expoente: dois mais bê ao quadrado fim
de expoente mais bê ao quadrado ">
         <node value=" á elevado ao expoente: dois mais bê ao quadrado
fim de expoente ">
             <node value=" á "></node>
             <node value=" elevado ao expoente: "></node>
                 <node value=" dois mais bê ao quadrado ">
                     <node value=" dois "></node>
                     <node value=" mais "></node>
                     <node value=" bê ao quadrado ">
                         <node value=" bê "></node>
                         <node value=" ao quadrado "></node>
                     </node>
                 </node>
                 <node value=" fim de expoente "></node>
         </node>
         <node value=" mais "></node>
         <node value=" bê ao quadrado ">
             <node value=" bê "></node>
             <node value=" ao quadrado "></node>
         </node>
     </node>
     <node value=" fim de radicando "></node>
  </node>
</mat>
```

Figura 11.11 - Exemplo de árvore de navegação.

Como se pode observar pela Figura 11.11:

- os níveis (`<node>`) mais elevados são menos detalhados, mas mais complexos. Exemplo: `<node value=" á elevado ao expoente: dois mais bê ao quadrado fim de expoente mais bê ao quadrado ">`
- os níveis (`<node>`) interiores são os mais detalhados, mas mais simples. Exemplo: `<node value=" bê "></node>`

A estrutura resultante da árvore de navegação permite uma melhor interacção com o utilizador e uma organização semelhante à apresentada na Figura 11.6, onde se discutiu o modo de interpretação de uma fórmula matemática.

De notar também o uso de palavras chave, como por exemplo: `<mat value="Fórmula:">`, implementadas com o objectivo de aproveitarem o efeito de *fan* discutido na secção anterior, e antecipando o texto que lhe segue.

Uma vez conseguida a formatação da árvore de navegação, de acordo com conceitos de percepção e memória, resta a interacção com o utilizador. Esta foi implementada no desenvolvimento do AudioMathGUI[110]. Os elementos gerados (`<node>`) são processados pela aplicação gráfica, que os representa numa árvore cujos nós podem ser estendidos ou não (Figura 11.12).

Figura 11.12 - Navegação através do AudioMathGUI - Vista 1.

Em (64) e (46) foi proposto o uso do teclado para navegar numa fórmula matemática. O mesmo foi assumido no trabalho desenvolvido nesta dissertação.

[110] Ver capítulo 8.

Como se pode verificar pela Figura 11.12, existem seis teclas que controlam a leitura do nó que se encontra activo:

- *Tecla 1* – activa o nó anterior, no mesmo nível de navegação.
- *Tecla 2* – activa o nó seguinte, no mesmo nível de navegação.
- *Tecla 3* – permite navegar para o nível seguinte.
- *Tecla 4* – permite regressar ao nível anterior.
- *Tecla 5* – usado para repetir a leitura.
- *Tecla 6* – usado para parar a leitura

Para além da visualização em árvore da Figura 11.12, o AudioMathGUI permite também a visualização do código gerado pelo módulo de navegação do AudioMathENGINE.

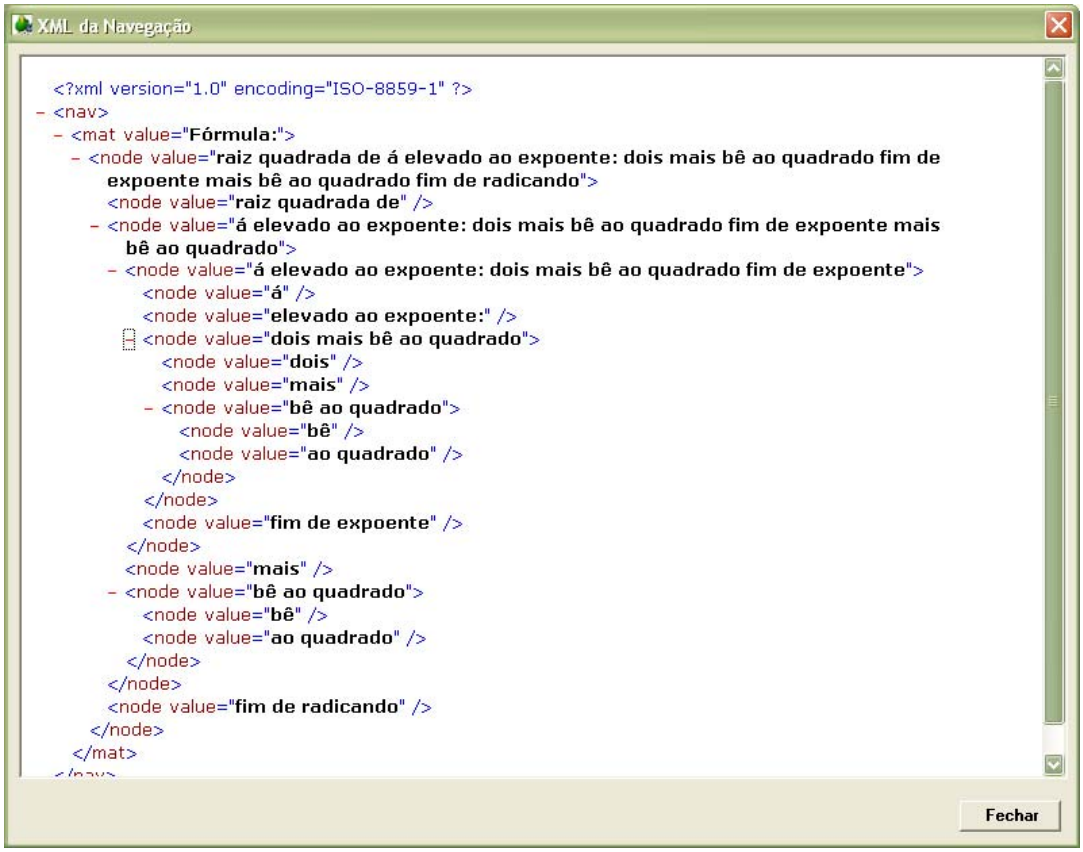


Figura 11.13 - Navegação através do AudioMathGUI - Vista 2.

Desta forma, o trabalho elaborado durante esta dissertação cumpre as exigências de mecanismos de navegação para a interpretação oral das expressões matemáticas.

12 Testes e Validação de Resultados

Neste capítulo pretende-se apresentar o conjunto de testes a que foram sujeitas as aplicações desenvolvidas no âmbito desta dissertação, como forma de validação dos resultados obtidos. Foram elaborados quatro tipos distintos de testes:

- Robustez e qualidade dos algoritmos de análise, interpretação e conversão do AudioMathENGINE;
- Interpretação escrita de uma expressão matemática;
- Interpretação oral de uma expressão matemática;
- Efectividade do mecanismo de navegação de expressões matemáticas.

O primeiro teste foi concretizado através da experimentação intensiva de mais de 400 expressões matemáticas. Para os restantes testes foi escolhido um grupo heterogéneo de catorze utilizadores, com idades compreendidas entre os 22 e os 60 anos e com diversos graus de conhecimento na área da matemática e da informática.

O grupo de utilizadores incluía duas professoras universitárias de matemática e quatro cegos. No caso dos utilizadores sem deficiência visual, esta foi simulada.

12.1 Algoritmos de conversão do AudioMathENGINE

O primeiro módulo de testes incidiu sobre a robustez e qualidade dos algoritmos de análise, interpretação e conversão do AudioMathENGINE. A preparação de materiais para este teste consistiu:

- na criação de testes unitários para cada um dos algoritmos de análise e conversão;
- elaboração de conjuntos de testes específicos para a fase de desenvolvimento de módulos;
- elaboração de conjuntos de testes específicos para a fase de testes finais;
- conversão do corpus matemático;
- conversão de fichas de trabalho de Análise Matemática I;
- análise, interpretação e conversão de diversas páginas on-line do jornal *O Público*.

Neste módulo de testes foram considerados três tipos de erros:

- *Erro de identificação* (Não identificado) – quando o elemento, apesar de bem convertido, foi mal identificado;
- *Erro de regras reduzidas* (Não convertido) – quando o elemento não é convertido por falta de regras conhecidas, por exemplo, quando se desconhece uma abreviação ou um acrónimo;
- *Erro de conversão* (Mal convertido) – quando o resultado de conversão está errado.

12.1.1 Resultados da fase final de testes

Juntando todos os documentos usados nesta fase final de testes, totalizam-se:

- 73176 caracteres;
- 447 numerais;
- 20 abreviações;
- 6 acrónimos;
- 2 referências de rede;
- 108 expressões matemáticas.

Após o processamento do AudioMathENGINE foi produzido um documento de resultado com:

- 26081 caracteres;
- 447 numerais identificados e convertidos;
- 20 abreviações identificadas e convertidas;
- 6 acrónimos identificados e convertidos;
- 2 referências de rede identificadas e convertidas;
- 108 expressões matemáticas identificadas e convertidas;
- 23 segundos de processamento total.

Imediatamente se percebe que a quantidade de caracteres usados antes e após a conversão teve um decréscimo de cerca de 35%, o que se justifica pelo facto das fórmulas matemáticas contidas no documento original se encontrarem em XML (as etiquetas de marcação são um excesso).

Tabela 12.1 - Estatísticas: testes finais de conversão.

Tipo de elemento	Não identificados	Não convertidos	Mal convertidos
Numerais	0	0	0
Abreviações	0	0	1
Acrónimos	0	0	0
Referências de Rede	0	0	0
Expressões Matemáticas	0	0	8

A Tabela 12.1 reúne os resultados do teste em função dos três tipos de erros declarados anteriormente. Como se pode verificar, não existiram erros de identificação ou erros por falta de regras. Na realidade, quase todos os elementos foram bem identificados e convertidos pelos algoritmos do AudioMathENGINE.

No entanto observam-se alguns problemas com as expressões matemáticas. Um dos problemas encontra-se relacionado com a conversão de uma expressão usando *MathML Content Markup*, os outros sete estão relacionados com a conversão de fórmulas que definem derivadas. Como já foi referido em capítulos anteriores, o AudioMathENGINE não possui uma boa conversão para este tipo de expressões matemáticas.

12.1.2 Resultados da análise e conversão do corpus matemático

O documento que representa o corpus matemático é composto por:

- 176818 caracteres;
- 295 numerais;
- 98 expressões matemáticas

Após o processamento do AudioMathENGINE foi produzido um documento de resultado com:

- 295 numerais identificados e convertidos;
- 98 expressões matemáticas identificadas e convertidas;
- 12 segundos de processamento total.

Tabela 12.2 – Estatísticas: conversão do corpus matemático.

Tipo de elemento	Não identificados	Não convertidos	Mal convertidos
Numerais	0	0	0
Abreviações	0	0	0
Acrónimos	0	0	0
Referências de Rede	0	0	0
Expressões Matemáticas	0	0	11

A Tabela 12.2 reúne os resultados do teste em função dos três tipos de erros declarados anteriormente. Novamente, a maioria das expressões matemáticas mal convertidas são as fórmulas que expressam derivadas, e uma fórmula que expressa um limite.

12.1.3 Resultados da análise e conversão das fichas de trabalho de Análise Matemática

O documento que representa o corpus matemático é composto por:

- 100459 caracteres;
- 194 numerais;
- 2 abreviações;
- 3 acrónimos;
- 84 expressões matemáticas

Após o processamento do AudioMathENGINE foi produzido um documento de resultado com:

- 194 numerais identificados e convertidos;
- 2 abreviações identificadas e convertidas;
- 3 acrónimos identificados e convertidos;
- 84 expressões matemáticas identificadas e convertidas;
- 44 segundos de processamento total.

O tempo de processamento aumentou, uma vez que esta página encontrava-se on-line.

Tabela 12.3 – Estatísticas: conversão de fichas matemáticas.

Tipo de elemento	Não identificados	Não convertidos	Mal convertidos
Numerais	0	0	0
Abreviações	0	0	0
Acrónimos	0	0	0
Referências de Rede	0	0	0
Expressões Matemáticas	0	0	7

A Tabela 12.3 reúne os resultados do teste em função dos três tipos de erros declarados anteriormente. Algumas expressões matemáticas apresentaram deficiências na sua conversão. Foram os casos de expressões com limites, intervalos de valores e módulos.

12.1.4 Análise e conversão de documentos on-line

Para a concretização destes testes foram utilizadas reportagens on-line do Jornal O Público. Os documentos analisados e convertidos foram:

- Notícia de Jornal O Público on-line de 11.09.2005 – "Albergaria-a-Velha: um morto e três feridos em despiste de autocarro".
- Notícia de Jornal O Público on-line de 11.09.2005 – "Milhares de pessoas lembram 11 de Setembro e marcham pela liberdade".
- Noticia de Jornal O Público on-line de 10.09.2005 – "Preços de petróleo ameaçam abrandar crescimento da União Europeia".
- Notícia de Jornal O Público on-line de 09.09.2005 – "Consumo de electricidade subiu 7,1 por cento em Agosto face a igual mês de 2004".
- Notícia de Jornal O Público on-line de 09.09.2005 – "Bolsa: Wall Street sobe com novo recuo dos preços do petróleo".
- Notícia de Jornal O Público on-line de 09.09.2005 – "Ministro das Finanças acredita no apoio europeu às medidas de correcção do défice".
- Noticia de Jornal O Público on-line de 09.09.2005 – "Bolsa abre a subir ligeiros 0,04 por cento".
- Noticia de Jornal O Público on-line de 09.09.2005 – "TSD propõem aumento salarial de 3,5 por cento para 2006".

O conjunto destes documentos totaliza:

- 116 numerais;
- 10 acrónimos;

Nenhum dos documentos continha expressões matemáticas.

Tabela 12.4 – Estatísticas: conversão de documentos on-line.

Tipo de elemento	Não identificados	Não convertidos	Mal convertidos
Numerais	14	0	0
Abreviações	0	0	0
Acrónimos	0	10	0
Referências de Rede	0	0	0

A Tabela 12.4 reúne os resultados do teste em função dos três tipos de erros declarados anteriormente. O elevado número de numerais não identificados (embora tenham sido convertidos) devem-se ao facto de estes serem do tipo datas e horas. Tipos esses que não se encontram incluídos nos algoritmos de reconhecimento automático. Os erros de não conversão de acrónimos devem-se à falta de definição de siglas no dicionário AKML. Bastaria a sua introdução no dicionário para eliminar este tipo de erro.

12.1.5 Conclusões

Dos resultados provenientes dos diversos módulos de testes utilizados, concluiu-se:

- Os documentos originais possuem mais caracteres que os documentos convertidos, devido à existência do XML.
- Os erros de conversão de acrónimos ou abreviações devem-se à falta de definições nos dicionários utilizados. Uma simples actualização dos mesmos, eliminaria este tipo de erros.
- Existem três tipos de expressões matemáticas, que apesar de serem convertidas apresentam uma qualidade deficiente. São elas: limites, módulos e derivadas.

Conclui-se que, até ao momento da escrita desta dissertação, os algoritmos de análise, interpretação e conversão eram robustos e estáveis.

12.2 Interpretação Escrita

Este módulo de testes incidiu sobre a forma como é percepcionado, pelos utilizadores, o texto resultante dos algoritmos de análise e conversão. A preparação de materiais para este teste consistiu na realização de um questionário[111] contendo diversas expressões matemáticas convertidas pelo AudioMathENGINE. O questionário era composto por duas partes:

- Grupo de expressões matemáticas ao nível do ensino secundário – 10 fórmulas de: álgebra simples, fracções, raízes e potências;
- Grupo de expressões matemáticas ao nível do ensino superior – 10 fórmulas de: integrais, somatórios, limites, matrizes e trigonometria.

Foi pedido aos utilizadores que lessem a fórmula matemática e a representassem graficamente. Das respostas obtidas pelo grupo de utilizadores apresentado na introdução deste capítulo, concluiu-se que:

- Todos os utilizadores foram capazes de representar graficamente as fórmulas pedidas.
- Um dos utilizadores, apesar de ter representado bem todas as fórmulas, sentiu a necessidade de incluir parêntesis nas sub expressões de determinados expoentes complexos.
- Foi sugerida uma modificação na conversão de matrizes, de modo a ser incluído o grau da matriz antes da descrição detalhada da mesma.

De modo geral, e pela reacção dos intervenientes neste estudo, as conversões resultantes dos algoritmos do AudioMathENGINE demonstraram ser eficazes e não ambíguas, permitindo as suas representações gráficas sem dificuldades.

O resultado destes testes, por parte de utilizadores, valida assim os resultados obtidos através dos algoritmos de análise, interpretação e conversão desenvolvidos.

[111] O questionário entregue aos utilizadores é disponibilizado no ANEXO F.

12.3 Interpretação Oral

Os testes de interpretação oral de expressões matemáticas basearam-se na inteligibilidade dos ficheiros áudio gravados durante a elaboração do corpus matemático do AudioMath. A preparação de materiais para este teste consistiu na realização de um questionário[112] contendo ficheiros áudio de diversas expressões matemáticas. O questionário era composto por duas partes:

- Grupo de expressões matemáticas ao nível do ensino secundário – 8 fórmulas de: álgebra simples, fracções, raízes e potências;
- Grupo de expressões matemáticas ao nível do ensino superior – 6 fórmulas de: integrais, somatórios, limites, matrizes e trigonometria.

Foi pedido aos utilizadores que ouvissem a fórmula matemática e a representassem graficamente. Caso fosse necessário escutar mais que três vezes o ficheiro áudio, o utilizador deveria de assinalar esse acontecimento. Das respostas obtidas concluiu-se que:

- Todos os utilizadores foram capazes de representar graficamente as fórmulas pedidas.
- Nenhum utilizador necessitou de escutar mais de três vezes um ficheiro áudio para conseguir representar graficamente a fórmula matemática. No entanto, nalgumas expressões ao nível do ensino superior (as mais complexas e longas) foi necessário recorrer à repetição do ficheiro áudio, por uma ou duas vezes.

De modo geral, e pela reacção dos intervenientes neste estudo, as gravações de expressões matemáticas demonstraram ser eficazes e não ambíguas. As pausas e a prosódia utilizada contribuíram para que a maioria dos utilizadores identificasse as expressões na primeira leitura, sem necessidade de repetições.

O resultado destes testes, por parte de utilizadores, valida assim os resultados obtidos através da análise dos sinais de fala.

[112] O questionário entregue aos utilizadores é disponibilizado no ANEXO G.

12.4 Mecanismo de Navegação

Para concluir a fase de testes das aplicações desenvolvidas no âmbito desta dissertação, foram realizados testes de eficácia e usabilidade ao mecanismo de navegação do AudioMathGUI.

Neste teste usou-se um documento[113] com quatro expressões matemáticas complexas. Uma destas expressões serviu como base de treino, enquanto que as outras foram analisadas pelos utilizadores. Foi pedido a cada utilizador que, com o monitor do computador desligado, navegasse no documento convertido e representasse graficamente tudo o que era reproduzido pela voz sintetizada.

Todos os utilizadores foram capazes de representar graficamente as fórmulas sintetizadas, no entanto o mecanismo de navegação apresentou algumas deficiências. Das opiniões recolhidas pelos utilizadores, concluiu-se que:

- O mecanismo de navegação exige alguma prática até poder ser dominado. Contudo no caso dos utilizadores cegos a adaptação foi muito rápida (cerca de 5 minutos), em contraste com os utilizadores sem a deficiência visual (cerca de 15 minutos).
- Os sinais sonoros das teclas antes da síntese da fala são algo incomodativos;
- A qualidade da síntese da fala poderia ser melhorada;
- A escolha de teclas é um pouco confusa;
- Falta de pausas na leitura das fórmulas;
- Falta a identificação do nível em que se encontra o cursor;

Verificou-se também que muitos utilizadores dispensaram a maior parte dos níveis da navegação, e conseguiram representar a expressão matemática usando apenas o primeiro e segundo nível da navegação.

Conclui-se assim que o mecanismo de navegação implementado no AudioMathGUI necessita de ser aperfeiçoado. No entanto, demonstra ser útil na interpretação da fórmula matemática até aos primeiros dois níveis de complexidade.

[113] O documento usado no teste de navegação pode ser encontrado no ANEXO H.

Página em branco.

13 Conclusões e Trabalho Futuro

Durante este trabalho foi possível adquirir diversos conhecimentos nas mais variadas áreas, tais como a psicologia, a matemática, a engenharia e a linguística.

A pesquisa de soluções e tecnologias já existentes demonstrou a raridade de estudos e aplicações que permitem eficazmente o acesso à informação técnica e científica por parte das pessoas com necessidades especiais, nomeadamente os cegos e os amblíopes.

A maioria dos métodos de produção e publicação deste tipo de conteúdos on-line revelaram ineficácia e inacessibilidade. Por esse motivo a W3C optou pela criação e especificação da linguagem de marcação matemática – MathML. Esta foi a tecnologia usada como ponto de partida para a elaboração desta dissertação. A sua aplicação relevou-se bastante promissora, uma vez que a fórmula matemática codificada em MathML expressa tanto a sua semântica (*MathML Content Markup*) como a sua notação (*MathML Presentation Markup*). A sua representação numa estrutura XML confere-lhe flexibilidade e portabilidade. Também as capacidades de criação de conteúdos dinâmicos e interactivos on-line e troca de informação matemática entre aplicações, conferem ao MathML um estatuto relevante na produção e publicação de conteúdos matemáticos. Durante a dissertação foram produzidos diversos materiais on--line com XHTML e MathML usando `Microsoft Word` e `MathType`. Materiais esses que foram visualizados no navegador de páginas on-line (*browser*) `Internet Explorer 6.0`, através do mecanismo externo (*plug-in*) `MathPlayer`.

Foi realizado um estudo tendo como objectivo a interpretação oral da codificação MathML, onde o primeiro desafio colocado consistia na escolha do conjunto de etiquetas de marcação MathML a usar. Concluiu-se que apesar do *Content Markup* ser o mais adequado (uma vez que exprime a semântica da fórmula), por uma questão prática foi seleccionado o *Presentation Markup*, uma vez que todos os materiais produzidos e publicados on-line encontram-se codificados desta forma. O segundo desafio colocado consistia na interpretação das etiquetas de marcação do MathML. Nem todas necessitam de ser processadas e interpretadas, conforme foi apresentado no capítulo 2. O terceiro desafio colocado consistia na determinação da forma correcta e não ambígua de leitura de expressões matemáticas. Para esse efeito foi realizado um estudo aprofundado de diversos grupos de expressões matemáticas (álgebra simples, fracções, raízes, potências, trigonometria, matrizes, integrais, derivadas e somatórios) do qual resultou um *corpus*

matemático e onde foram identificadas palavras chave na interpretação oral da matemática.

Com base nos estudos anteriores foi desenvolvida uma aplicação - AudioMathENGINE – com capacidade de análise, interpretação e conversão de diversos elementos num documento, tais como: abreviações, acrónimos, numerais, referências de rede e fórmulas matemáticas. Esta aplicação é composta por um conjunto de diversas expressões regulares que procuram determinados padrões num texto, identificam-nos e processam-nos da forma mais adequada. Foram realizados diversos testes com o objectivo de optimização destas expressões regulares, uma vez que dependendo do tipo de motor usado (determinístico ou não determinístico), e a forma como a expressão regular é declarada, o tempo gasto a identificar um padrão no texto é afectado.

Durante a elaboração destes algoritmos de análise, reconhecimento e conversão foi necessário criar uma linguagem de XML interna da aplicação – AKML - com o objectivo de identificar posteriormente as acções de substituição que ocorreram, assim como estruturar os resultados do AudioMathENGINE. O uso do AKML conferiu aos resultados uma maior flexibilidade, permitindo a sua análise e processamento por parte de outras aplicações de outros autores, desde que o seu XML *Schema* seja conhecido.

Uma vez que a aplicação AudioMathENGINE processa *Presentation Markup*, foi utilizada uma folha de estilo de transformação XML (XSLT) que converte *Content Markup* em *Presentation Markup*. Deste modo, foi possível implementar o suporte para um novo conjunto de etiquetas de marcação do MathML. Foram realizados diversos testes com conhecidas folhas de estilo XML, onde se concluiu sobre a utilização da XSLT fornecida pela W3C.

Durante este desenvolvimento foram também desenvolvidos dicionários Unicode para as entidades do MathML, que fazem a transformação entre o código MathML para Unicode e posteriormente para Português Europeu. Tanto quanto se sabe, estes foram os primeiros dicionários desenvolvidos a permitir esta transformação.

Juntamente com a conversão das fórmulas matemáticas foi implementado um algoritmo de navegação das mesmas. Este algoritmo permite a colocação de pontos de navegação e estruturação em árvore de uma fórmula matemática. A colocação dos pontos de navegação que permite construir a árvore de navegação é baseada nos conceitos de que uma expressão matemática pode ser subdividida em sub expressões

menos complexas e assim sucessivamente. Provou-se, com testes realizados, a utilidade deste tipo de estrutura.

Como complemento ao motor de análise, interpretação e conversão, AudioMathENGINE foram desenvolvidas mais duas aplicações: um interface gráfico que permite a edição, visualização, conversão e navegação de conteúdos matemáticos em MathML – AudioMathGUI; e um serviço on-line de conversão de MathML – AudioMathWEB.

Com o desenvolvimento destas três aplicações (AudioMathENGINE, AudioMathGUI e AudioMathWEB) cumprem-se os objectivos desta dissertação relacionados com a interpretação e conversão de conteúdos matemáticos para texto, e a navegação contextualizada desses conteúdos. Com a finalidade de cumprir o último objectivo da tese (a leitura das fórmulas matemáticas), foi realizado um estudo relacionado com conversores texto-fala e as diversas linguagens de marcação para tecnologias de fala. Do estudo realizado concluiu-se que as linguagens mais importantes a implementar são as relacionadas com o SAPI4, SAPI5, e o SSML. Foram reunidas informações acerca das suas etiquetas de marcação mais relevantes, nomeadamente pausas, frequência fundamental, volume, prosódia e velocidade.

Uma vez conhecidas as formas de estruturar um texto por extenso com estas marcações, restava apenas realizar um estudo aprofundado acerca da prosódia da matemática. O *corpus* matemático definido para o AudioMathENGINE foi gravado e todos os sinais de fala analisados em termos dos padrões da frequência fundamental e das pausas. Do estudo realizado e da análise destes sinais foram extraídas diversas regras da prosódia matemática, nomeadamente na colocação de pausas e na descrição de padrões prosódicos.

Na conclusão deste trabalho, e como forma de validação do mesmo, foram realizados diversos testes com utilizadores visuais e cegos. Nos testes de interpretação escrita foi avaliado a qualidade da conversão do AudioMathENGINE, onde se provou a sua eficácia e ausência de ambiguidade, permitindo a todos os utilizadores a interpretação correcta da fórmula matemática a partir do texto escrito.

Nos testes de interpretação oral foi avaliado a qualidade da prosódia matemática, onde se provou a sua eficácia, permitindo a todos os utilizadores a interpretação correcta da fórmula matemática a partir das gravações do corpus. Todos os utilizadores conseguiram identificar a fórmula no máximo de duas leituras da mesma.

Nos testes de navegação foi avaliado o mecanismo de navegação implementado no AudioMathGUI onde, apesar de se ter provado a sua eficácia, foram recolhidas sugestões importantes para o seu aperfeiçoamento. Contudo, todos os utilizadores conseguiram identificar correctamente a fórmula matemática utilizando apenas os dois primeiros níveis de navegação.

Conclui-se assim que todos os objectivos propostos para esta dissertação foram cumpridos com sucesso, e abriram novos caminhos para trabalhos futuros.

13.1 Trabalho Futuro

O trabalho desenvolvido e o seu objectivo merecem continuidade e dedicação. A aplicação real e eficaz, como contributo para a sociedade, sob a forma de ferramentas educacionais ou de acessibilidade justificam o empenho na continuação e no desenvolvimento dos produtos criados durante a dissertação.

Sendo este um trabalho multidisciplinar na conjugação de diferentes teorias, conceitos e tecnologias, existem com certeza diversas melhorias a ser implementadas e estudadas. Assim, a curto prazo pretende-se:

- Concluir o suporte para *MathML Presentation Markup* e *MathML Content Markup*;
- Adicionar suporte de tecnologias de fala para conversores texto-fala, nomeadamente SAPI4, SAPI5 e SSML;
- Incluir a opção de modos de utilizador para expressões matemáticas;

A longo prazo pretende-se: explorar com mais detalhe a prosódia da matemática, construir um navegador de fórmulas adequado a cegos e amblíopes, suportar a conversão para mais do que uma língua, optimizar os algoritmos de análise, conversão e interpretação de expressões matemáticas; e desenvolver um serviço público e gratuito de acessibilidade a documentos técnicos on-line contendo MathML.

Referências

(1) CERF, Vint – **Brief History of the Internet**. Internet Society. 2004. http://www.isoc.org/internet/history/cerf.shtml

(2) APGC - Associação Portuguesa para a Gestão de Conhecimento – **Educação, Aprendizagem e Tecnologia: Um paradigma para Professores do Século XXI**. Edições Sílabo. 2005. ISBN 972-618-356-1.

(3) **Internet World Stats** - http://www.internetworldstats.com/stats.htm

(4) WHO - World Health Organization – **Magnitude and causes of visual impairment**. 2004. http://www.who.int/mediacentre/factsheets/fs282/en/

(5) BAUTISTA, Rafael (ed.) – **Necessidades Educativas Especiais**. Dinalivro. 1993. ISBN 972-576-116-2.

(6) NIELSON, Lee – **Necessidades Educativas Especiais na Sala de Aula**. Porto Editora. 1999. ISBN 972-0-34503-9.

(7) CRUZ, Vítor – **Dificuldades de Aprendizagem**. Porto Editora. 1999. ISBN 972-0-34504-7

(8) FITZPATRICK, Donal; MONAGHAN, A. – **Browsing Technical Documents: Document Modelling and User Interface Design**. BULAG. 1999.

(9) KING, Alasdair; EVANS, Gareth; BLENKHORN, Paul – **Blind People and the World Wide Web**. 2005. http://www.webbie.org.uk/webbie.htm

(10) CLARK, Joe – **Building Accessible Websites**. 2002. ISBN 0-735-71150-X

(11) W3C – **W3C Math Home**. 2005. http://www.w3.org/Math/

(12) W3C – **Web Content Accessibility Guidelines 1.0**. 1999. http://www.w3.org/TR/WCAG10/

(13) KNUTH, Donald – **The TeXbook**. Addison-Wesley. 1986. ISBN 0-201-13447-0.

(14) MICROSOFT - **Word 2003 On-line**. http://office.microsoft.com/en-us/FX010857991033.aspx

(15) W3C – **HTML 4.01 Specification**. 1999. http://www.w3.org/TR/html4/

(16) ADOBE – **PDF Reference fifth edition version 1.6**. 2004. http://partners.adobe.com/public/developer/pdf/index_reference.html#5

(17) ADOBE – **PostScript Language Reference Manual: Second Edition**. Addison-Wesley. 1990. ISBN 0-201-181127-4.

(18) MICROSOFT – **Rich Text Format (RTF) version 1.5 Specification**. 1997. Microsoft Technical Support. http://www.biblioscape.com/rtf15_spec.htm

(19) SUN – **Applets**. 2005. http://java.sun.com/applets/

(20) W3C – **XHTML 1.0 The Extensible HyperText Markup Language (second edition)**. 2002. http://www.w3.org/TR/xhtml1/

(21) W3C – **Cascading Style Sheets Homepage**. 2001. http://www.w3.org/Style/CSS/

(22) YEE, Ka-Ping – **MINSE: the easiest way to put math on the Web**. 2002. http://lfw.org/math/

(23) W3C – **Mathematical Markup Language (MathML) version 2.0 (second edition)**. 2003. http://www.w3.org/TR/MathML2/

(24) W3C – **Scalable Vector Graphics (SVG) 1.1 Specification**. 2003. http://www.w3.org/TR/SVG/

(25) DRAKOS, Nikos - **LaTeX2HTML**. http://saftsack.fs.uni-bayreuth.de/~latex2ht/

(26) Gurari, Eitan - **TeX4ht**. http://www.cse.ohio-state.edu/~gurari/TeX4ht/mn.html

(27) ADOBE – **Overview of Acrobat products and accessibility**. http://www.adobe.com/products/acrobat/access_overview.html

(28) DESIGN SCIENCE – **WebEQ, Interactive Math on the Web**. http://www.dessci.com/en/products/webeq/

(29) DESIGN SCIENCE – **WebTeX: A Markup Language for WebEQ**. 1997. http://www.mit.edu/afs/athena/software/webeq/currenthome/docs/webtex/

(30) IBM – **Techexplorer**. http://www.alphaworks.ibm.com/tech/techexplorer

(31) DESIGN SCIENCE – **MathPlayer, Displaying MathML in your browser**. http://www.dessci.com/en/products/mathplayer/

(32) HUTCHINSON, Ian – **TtH: The TeX to HTML translator**. http://hutchinson.belmont.ma.us/tth/

(33) W3C – **Cascading Style Sheets under construction (Math)**. http://www.w3.org/Style/CSS/current-work#Math

(34) BUSHWELL, S. et al – **The OpenMath Standard 2.0**. 2004. http://www.openmath.org/cocoon/openmath/standard/om20-2004-06-30/index.html

(35) DESIGN SCIENCE – **MathType**. http://www.dessci.com/en/products/mathtype/

(36) WOLFRAM – **Mathematica**.
http://www.wolfram.com/products/mathematica/index.html

(37) WOLFRAM – **Publicon**.
http://www.wolfram.com/products/publicon/index.html

(38) MACKICHAN – **Scientific Word**. http://www.mackichan.com/

(39) SOFT4SCIENCE – **SciWriter**.
http://www.soft4science.com/products/SciWriter/s4s_SciWriter.html

(40) GARTSIDE, Paul – **Itex2mml**. 2001.
http://pear.math.pitt.edu/mathzilla/itex2mml.html

(41) ORCCA – **MathML Projects**. http://www.orcca.on.ca/MathML/projects.html

(42) HUTCHINSON, Ian – **TtM: The TeX to MathML translator**.
http://hutchinson.belmont.ma.us/tth/mml/

(43) W3C – **MathML Software List**. http://www.w3.org/Math/Software/

(44) W3C – **AudioMath 2005**.
http://www.w3.org/Math/Software/mathml_software_cat_converters.html

(45) RAMAN, T.V. – **ASTER Demonstration**. 1994.
http://www.cs.cornell.edu/Info/People/raman/aster/demo.html

(46) RAMAN, T.V. – **Audio System For Technical Readings**. Tese de Doutoramento. Universidade de Cornell. 1994.

(47) STEVENS, Robert – **The MathTalk Program**.
http://www.cs.york.ac.uk/maths/robert/mathtalk.html

(48) STEVENS, Robert; EDWARDS, Alistair – **MathTalk: Usable Access To Mathematics**. ITD vol.1. 1994.
http://www.rit.edu/~easi/itd/itdv01n4/article5.htm

(49) STEVENS, Robert – **Principles for the Design of Auditory Interfaces to Present Complex Information to Blind People**. Tese de Doutoramento. Universidade de York.1996.

(50) NEMETH, Abraham – **The MathSpeak Initiative**. 2004.
http://www.gh-mathspeak.com/index.php

(51) NEMETH, Abraham – **The Nemeth Code for Mathematics and Science Notation**. APH. 1972.

(52) **The Nemeth Code**. 2002. http://www.dotlessbraille.org/nemethcode.htm

(53) DAISY – **Digital Talking Books**. http://www.daisy.org/about_us/dtbooks.asp

(54) KARSHMER, Arthur; GILLAN Doug – **How Well Can We Read Equations to Blind Mathematics Students: Some Answers from Psychology**. 2003. HCII03 Proceedings, vol. 4, pp. 1290-1294. ISBN: 0-8058-4933-5.

(55) KARSHMER, Arthur; BLEDSOE, C.; STANLEY, P. – **The Architecture of a Comprehensive Equation Browser for the Print Impaired**. 2004. ICCHP04 Proceedings, pp. 614-619. ISBN 3-540-22334-7.

(56) GILLAN, D.; BARRAZA, P.; KARSHMER, A. – **Cognitive Analysis of Equation Reading: Application to the Development of the Math Genie**. ICCHP04 Proceedings, pp. 630-637. ISBN 3-540-22334-7.

(57) FITZPATRICK, D.; KARSHMER, A. – **Multi-modal Mathematics: Conveying Math Using Synthetic Speech and Speech Recognition**. 2004. ICCHP04 Proceedings, pp. 644-647. ISBN 3-540-22334-7.

(58) Karshmer, A. – **The MathGenie Project**. 2005. http://karshmer.lklnd.usf.edu/~mathgenie/

(59) SOIFFER, Neil – **Accessible Mathematics Made Easy**. CSUN Proceedings. 2004.

(60) FOGAROLO, Flavio – **Maths and blind students** – the Lambda project.

(61) NICOTRA, Giuseppe; FOGAROLO, Flavio – **Interfaces for Access to music and mathematics by blind persons**. IST03. 2003.

(62) FERREIRA, Helder; FREITAS, Diamantino – **Audio Rendering of Mathematical Formulae using MathML and AudioMath**. UI4ALL, pp. 391-399. 2004. ISBN 3-540-23375-X.

(63) FERREIRA, Helder; FREITAS, Diamantino – **Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project**. ICCHP04, pp. 678-685. 2004. ISBN 3-540-22334-7.

(64) FERREIRA, Helder – **AudioMath: Speaking Mathematics with MathML**. 2nd European Workshop on MathML & Scientific e-Contents. 2004.

(65) FERREIRA, Helder; FREITAS, Diamantino – **AudioMath: using MathML for speaking mathematics**. XATA05. 2005.

(66) FERREIRA, Helder; FREITAS, Diamantino – **AudioMath: Towards Automatic Readings of Mathematical Expressions**. HCII05. 2005.

(67) W3C - **Extensible Markup Language (XML) 1.0 (Third Edition)**. 2004. http://www.w3.org/TR/2004/REC-xml-20040204/

(68) GOLDFARB, Charles F. – **The SGML Handbook**. Oxford University Press. 1990. ISBN 0-19-853737-1.

(69) HAROLD, Elliotte – **XML Bible (2nd Edition)**. John Wiley & Sons. 2001. ISBN 0-76-454760-7

(70) W3C – **XML Schema**. 2000. http://www.w3.org/XML/Schema#dev

(71) The Unicode Consortium – **The Unicode Standard, Version 4.0**. 2003. Addison-Wesley. ISBN 0-321-18578-1.

(72) COELHO, Pedro – **XML A nova Linguagem da WEB**. 2000. FCA. ISBN 972-722-214-5

(73) W3C – **XML Namespaces**. 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/Overview.html

(74) W3C – **Document Object Model (DOM)**. 1998. http://www.w3.org/DOM/

(75) HAMPTON, Kip – **High-Performance XML Parsing with SAX**. 2001. O'Reilly XML.COM. http://www.xml.com/pub/a/2001/02/14/perlsax.html

(76) SANDHU, Pavi – **The MathML Handbook**. Charles River Media. 2003. ISBN 1-58450-249-5.

(77) W3C – **The MathML DTD**. MATHML version 2.0 2nd Edition, Appendix A.2.5. 2001. http://www.w3.org/TR/MathML2/appendixa.html#parsing.dtd

(78) UNICODE Consortium – **What is Unicode?** Unicode Homepage. 2005. http://www.unicode.org/standard/WhatIsUnicode.html

(79) W3C – **Characters, Entities and Fonts**. MathML version 2.0 2nd Ed. Recommendation. 2001. Cap. 6. http://www.w3.org/TR/2003/REC-MathML2-20031021/chapter6.html

(80) W3C – **Putting Mathematics on the Web with MathML**. 2003. http://www.w3.org/Math/XSL/

(81) GOLD, Ben e MORGAN, Nelson – **Speech and Audio Signal Processing - Processing and Perception of Speech and Music**. John Wiley & Sons, 2000. ISBN: 0-471-35154-7.

(82) DELLER, John; HANSEN, John; PROAKIS, John – **Discrete-Time Processing of Speech Signals**. John Wiley & Sons. IEEE. 2000. ISBN 0-7803-5386-2.

(83) CHILDERS, D.G. – **Speech Processing and Synthesis Toolboxes**. John Wiley & Sons, 2000. ISBN: 0-471-34959-3.

(84) VALBERT, H. et al – **Voice transformation using PSOLA Technique**. Speech Communication, Vol. 11, EuroSpeech'91. 1992. ISSN: 0167-6393.

(85) BARROS, M. João et al – **HMM-based European Portuguese TTS System**. Interspeech05. 2005.

(86) MICROSOFT – **Microsoft Speech API 4.0 Documentation**. http://www.microsoft.com/speech/download/old/sdk40a.asp

(87) MICROSOFT – **Microsoft Speech API 5.1 Documentation**. http://www.microsoft.com/speech/download/old/sapi5.asp

(88) W3C – **Speech Synthesis Markup Language (SSML) Version 1.0**. W3C Recommendation 7 September 2004. http://www.w3.org/TR/2004/REC-speech-synthesis-20040907/

(89) SPROAT, Richard et al – **A Markup Language for Text-to-Speech Synthesis**. Proceedings of EuroSpeech'97. 1997.

(90) DEROSE, Steven – **The SGML FAQ Book: Understanding the Foundation of HTML and XML**. Kluwer Academic Publishers. 1997. ISBN: 0-7923-9943-9.

(91) W3C – **JSpeech Markup Language**. 2000 - http://www.w3.org/TR/jsml/

(92) SUN – **Java Speech API 1.0**. http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-doc/index.html

(93) SPROAT, Richard et al – **SABLE: A Standard for TTS Markup**. Position paper, 1998. http://www.bell-labs.com/project/tts/sabpap/sabpap.html

(94) BELL, Labs – **SABLE: A Synthesis Markup Language (version 1.0)**. 1999. http://www.bell-labs.com/project/tts/sable.html

(95) SPROAT, Richard e RAMAN, T.V. – **SABLE: an XML-based Aural Display List For The WWW**. 1999. http://compling.ai.uiuc.edu/rws/csssable.html

(96) W3C – **Aural style sheets**. 1998. http://www.w3.org/TR/1998/REC-CSS2-19980512/aural.html

(97) W3C – **CSS3 Speech Module**. 2004. http://www.w3.org/TR/2004/WD-css3-speech-20041216/

(98) WANG, Kuansan – **SALT: A Spoken Language Interface For Web-Based Multimodal Dialog Systems**. ISCA 2002 Proceedings.

(99) GUPTA, G. et al – **Building the Tower of Babel: Converting XML Documents to VoiceXML for Accessibility**. Proceedings ICCHP'03. 2003.

(100) W3C – **Voice Extensible Markup Language (VoiceXML) 2.1 Candidate Recommendation**. 2005. http://www.w3.org/TR/2005/CR-voicexml21-20050613/

(101) HOPCROFT, John; ULLMAN, Jeffrey; MOTWANI, Rajeev - **Introdução à Teoria de Autômatos, Linguagens e Computação**. Editora Campus. 2003. ISBN 85-352-1072-5.

(102) CRESTO, Rui – **Processadores de linguagens: da concepção à implementação**. IST Press. 2001. ISBN 972-8469-18-7.

(103) JURAFSKY, Daniel; MARTIN, James – **Speech and Language Processing: an introduction to natural language processing, computational linguistics, and speech recognition**. Prentice-Hall. 2000. ISBN 0-13-095069-6.

(104) FRIEDL, Jeffrey - **Mastering Regular Expressions 2nd Edition**. O'Reilly. 2002. ISBN 0-596-00289-0.

(105) SCHWARTZ, Randal; PHOENIX, Tom - **Learning Perl 3rd Edition**. O'Reilly 2001. ISBN 0-596-00132-0.

(106) ORWANT, Jon - **Computer Science & Perl Programming**. O'Reilly. 2002. ISBN 0-596-00310-2.

(107) WALL, Larry; CHRISTIANSEN, Tom; ORWANT, Jon - **Programming Perl 3rd Edition**. O'Reilly. 2000. ISBN 0-596-00027-8.

(108) STUBBLEBINE, Tony – **Regular Expression Pocket Reference**. O'Reilly. 2003. ISBN 0-596-00415-X.

(109) FERREIRA, Helder – **Contributo para a leitura automática de textos científicos**. Relatório Final do Projecto de Fim de Curso. 2003.

(110) DOMINUS, Mark – **How Regexes Work**. Computer Science & Perl Programming. O'Reilly. pp. 122-135. 2002.

(111) THOMPSON, Ken – **Regular Expression Search Algorithm**. Communications of the ACM, vol. 11, no.6, pp. 419-422. 1968.

(112) LIN, S.; RADÓ, T. - **Computer Studies of Turing Machine Problems**. Journal of the Association for Computing Machinery, 12, 196–212. 1965.

(113) MOHRI, Mehryar; PEREIRA, Fernando; RILEY, Michael – **Weighted Finite-State Transducers in Speech Recognition**. Proceedings of Automated Speech Recognition: Challenges for the Next Millennium. 2000.

(114) MOHRI, Mehryar; PEREIRA, Fernando; RILEY, Michael – **Weighted Automata in Text and Speech Processing**. ECAI-96. 1996.

(115) ROWEIS, Sam – **Hidden Markov Models**. SCIA Tutorial. 2003.

(116) AHO, Alfred; ULLMAN, Jeffrey – **Foundations of Computer Science. Principles of computer science**. Computer Science Press. 1992.

(117) BUSCHMANN, F. et al – **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons. 1996. ISBN 0-471-95869-7.

(118) GAMMA, Erich et al – **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley. 1995. ISBN 0-201-63361-2.

(119) PORTO EDITORA – **Dicionário da Língua Portuguesa 2003**. ISBN 9-789720-050014

(120) PINTO, José – **Novo Prontuário Ortográfico**. 5ª edição revista. 2004. ISBN 972-770-002-0.

(121) GIL, Fernando et al - **Algoritmo para leitura de siglas em um sintetizador de voz**. SBT'03. 2003.

(122) W3C - **SVG 1.1 Specification, Appendix H: Accessibility Support**. 2003. http://www.w3.org/TR/SVG11/access.html

(123) W3C – **Accessibility Features of SVG**. 2000. http://www.w3.org/TR/SVG-access/

(124) FATEMAN, Richard – **How can we speak math?** Journal of Symbolic Computation, vol. 25, n.2, 1998.

(125) RIBEIRO, Carlos – **Processamento Digital de Fala**. Engenharia de Sistemas de Telecomunicações e Electrónica. ISEL. 2002.

(126) COKER. C.; FUJIMURA, O. - **Model for the specification of the vocal tract area function.** Journal of the Acoustical Society of America, 40, 1271. 1966.

(127) ATAL, B.; RIOUL, O. - **Neural networks for estimating articulatory positions from speech**. Journal of the Acoustical Society of America, 86, 123-131. 1989.

(128) ATAL, B. et al - **Inversion of articulatory-to-acoustic transformation in the vocal tract by a computer-sorting technique**. Journal of the Acoustical Society of America, 63, 1535-1555. 1978.

(129) BAVEGARD, M. - **Towards an articulatory speech synthesizer: Model development and simulations**. TMH-QPSR, 1-15. 1996.

(130) GRAY, Henry; LEWIS, Warren – **Anatomy of the Human Body**. Lea & Febiger. 1918. ISBN 1-58734-102-6.

(131) OLIVEIRA, Magda; FERNANDES, Rosina – **A Cegueira: Mecanismos de Percepção, Aprendizagem e Memória**. Faculdade de Psicologia e Ciências da Educação da Universidade do Porto. 2000.

(132) PREECE, Jenny et al – **Human-Computer Interaction**. Addison-Wesley.1994. ISBN 0-201-62769-8.

(133) MILLER, George – **The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information**. The Psychological Review, vol. 63, pp. 81-97. 1956.

(134) ANDERSON, J.R. – **The Adaptative Character of Thought**. Lawrence Erlbaum Associates. 1990. ISBN 0-8058-0419-6.

(135) ANDERSON, J.R. – **The Architecture of Cognition**. Harvard University Press. 1983.

(136) STÖGER, Bernhard; MIESENBERGER, Klaus; BATUSIC, Mario – **Mathematical Working Environment for the Blind Motivation and Basic Ideas**. ICCHP'04 Proceedings, pp.656-663. 2004. ISBN 3-540-22334-7.

(137) HAYES, Brian – **Speaking of Mathematics**. American Scientist Online. April 1996. http://www.americanscientist.org/template/AssetDetail/assetid/24601

(138) RAYNER, K.; POLLATEK, A. – **The Psychology of Reading**. Lawrence Erlbaum. 1994. ISBN 0-80581-872-3

(139) STEVENS, Robert; EDWARDS, Alistair – **An Approach to the Evaluation of Assistive Technology**. ASSETS'96 Proceedings. 1996. ISBN 0-89791-776-6.

Página em branco.

ANEXO A. Exemplos de MathML Presentation Markup

Números

```
<mn>-6.67</mn> ou <mo>-</mo><mn>6.67</mn> - número decimal negativo
<mn>2002</mn> - cardinal
<mn>1.6e-19</mn> - número no formato de engenharia
<mn>0xAEFF</mn> - número hexadecimal
<mn>1100101</mn> - número binário
<mn>MCVIII</mn> - número romano
<mn>vinte e um</mn> - número por extenso
```

número fraccionário: $\frac{1}{2}$

```
<mfrac><mn>1</mn><mn>2</mn></mfrac>
```

número complexo: $2+3i$

```
<mrow>
  <mn>2</mn><mo>+</mo>
     <mrow>
        <mn>3</mn><mo>&InvisibleTimes;</mo><mi>&ImaginaryI;</mi>
     </mrow>
</mrow>
```

Identificadores (nomes de variáveis ou funções)

```
<mi>&pi;</mi> - símbolo de pi
<mi>x</mi> - variável chis
<mi>sin</mi> - função seno de
<mi>&ExponentialE;</mi> - número de neper
<mi>&ImaginaryI;</mi> - número imaginário
<mi>F</mi> - variável ou função éfe
<mi></mi> - identificador vazio
```

Operadores

```
<mo>+</mo> - mais
<mo>&int;</mo> - integral de
<mo>&sum;</mo> - somatório de
<mo>&lt;</mo> - menor ou igual que
<mo>(</mo> - abrir parêntesis
<mo>&InvisibleTimes;</mo> - vezes
<mo>&ApplyFunction;</mo> - função de
```

xy *- chis vezes ipsilon*

```
<mrow><mi>x</mi><mo>&InvisibleTimes;</mo><mi>y</mi></mrow>
```

A_{12} *- à índice um, dois*

```
<msub>
    <mi>A</mi>
    <mrow><mn>1</mn><mo>&InvisibleComma;</mo><mn>2</mn></mrow>
</msub>
```

$f(x)$ - éfe de chis

```
<mrow>
  <mi>f</mi><mo>&ApplyFunction;</mo>
  <mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
</mrow>
```

Texto

$Teorema 1: Se\ x > 0\ e\ y > 0,\ então\ xy > 0$

```
<mrow>
  <mtext>Teorema 1: Se </mtext>
  <mi>x</mi><mo>&gt;</mo><mn>0</mn>
  <mtext>e</mtext>
  <mi>y</mi><mo>&gt;</mo><mn>0</mn>
  <mtext>, então </mtext>
  <mi>x</mi><mo>&InvisibleTimes;</mo><mi>y</mi>
  <mo>&gt;</mo><mn>0</mn>
</mrow>
```

Outras expressões

$\sum_{a}^{b} f(x)$

```
<mstyle displaystyle="true">
  <munderover>
     <mo>&sum;</mo>
     <mi>a</mi>
     <mi>b</mi>
  </munderover>
  <mrow>
     <mi>f</mi>
     <mo>&ApplyFunction;</mo>
     <mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
  </mrow>
</mstyle>
```

$\sqrt{2}$ `<msqrt><mn>2</mn></msqrt>`

$\sqrt[3]{x}$ `<mroot><mi>x</mi><mn>3</mn></mroot>`

$\binom{n}{r}$

```
<mrow>
  <mo>(</mo>
     <mfrac linethickness="0"><mi>n</mi><mi>r</mi></mfrac>
  <mo>)</mo>
</mrow>
```

$1 \mathbin{/} x + y$

```
<mrow>
  <mfrac bevelled="true">
     <mn>1</mn>
     <mrow><mi>x</mi><mo>+</mo><mi>y</mi></mrow>
  </mfrac>
</mrow>
```

$[0,1)$

```
<mfenced open="["><mn>0</mn><mn>1</mn></mfenced>
```

$(a+b+c)$

```
<mfenced separator="+"><mi>a</mi><mi>b</mi><mi>c</mi></mfenced>
```

$37\overline{)1456}$

```
<mrow>
  <mn>37</mn>
  <menclose notation=”longdiv”><mn>1456</mn></menclose>
</mrow>
```

x^2 `<msup><mi>x</mi><mn>2</mn></msup>`

$$\int_a^b f(x)dx$$

```
<mrow>
  <munderover>
     <mo>&int;</mo>
     <mi>a</mi>
     <mi>b</mi>
  </munderover>
  <mi>f</mi>
  <mo>&ApplyFunction;</mo>
  <mrow>
     <mo>(</mo><mi>x</mi><mo>)</mo>
  </mrow>
  <mo>&DifferentialD;</mo><mi>x</mi>
</mrow>
```

Tabelas e Matrizes

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
<mrow>
  <mo>(</mo>
     <mtable>
        <mtr>
           <mtd><mn>1</mn></mtd>
           <mtd><mn>2</mn></mtd>
        </mtr>
        <mtr>
           <mtd><mn>3</mn></mtd>
           <mtd><mn>4</mn></mtd>
        </mtr>
     </mtable>
  <mo>)</mo>
</mrow>
```

ANEXO B. Exemplos de MathML Content Markup

Números

$\frac{1}{2}$ `<cn type=”rational”>1<sep/>2</cn>`

$3+4i$ `<cn type=”complex-cartesian”>3<sep/>4</cn>`

$EF2_{16}$ `<cn type=”integer” base=”16”>EF2</cn>`

π `<cn type=”constant”>&pi;</cn>`

Identificadores

```
<ci type=”vector”>A</ci> - vector A
<ci>x</ci> - variável chis
```

Símbolos

N_A

```
<csymbol encoding=”text” definitionURL=”exemplo.com/advogadro.htm”>
  <msub>
     <mi>N</mi>
     <mi>A</mi>
  </msub>
</csymbol>
```

Outras expressões

$\sin(x)$

```
<apply>
  <sin/>
     <ci>x</ci>
</apply>
```

$(f+g)(x)$

```
<apply>
  <apply>
     <plus/>
     <ci>f</ci>
     <ci>g</ci>
  </apply>
  <ci>x</ci>
</apply>
```

$\int_0^{\pi} \sin(x)dx$

```
<apply>
  <int/><bvar><ci>x</ci></bvar>
  <interval>
     <cn>0</cn><pi/>
  </interval>
  <apply>
     <sin/><ci>x</ci>
  </apply>
</apply>
```

$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

```
<matrix>
    <matrixrow>
       <cn>1</cn><cn>0</cn>
    </matrixrow>
    <matrixrow>
       <cn>0</cn><cn>1</cn>
    </matrixrow>
</matrix>
```

ANEXO C. AKML Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="akml">
      <xs:annotation>
            <xs:documentation>root element</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="engine" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element ref="dictionary" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element ref="result" minOccurs="0"/>
                  <xs:element ref="gui" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="version" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="1.0"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
</xs:element>
<xs:element name="conversion">
      <xs:annotation>
      <xs:documentation>conversion definitions</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:attribute name="name" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="cardinal"/>
                              <xs:enumeration value="date"/>
                              <xs:enumeration value="decimal"/>
                              <xs:enumeration value="fractional"/>
                              <xs:enumeration value="ip"/>
                              <xs:enumeration value="ordinal"/>
                              <xs:enumeration value="percentage"/>
                              <xs:enumeration value="roman"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="usermode" type="xs:integer"
use="required"/>
            <xs:attribute name="gender">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="m"/>
                              <xs:enumeration value="f"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="quantity">
                  <xs:simpleType>
```

```
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="s"/>
                              <xs:enumeration value="p"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="type">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="c"/>
                              <xs:enumeration value="o"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
</xs:element>
<xs:element name="conversiondefaults">
      <xs:annotation>
            <xs:documentation>default conversion
definitions</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="module" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="conversionsettings">
      <xs:annotation>
            <xs:documentation>config for
conversions</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="module" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="engine">
      <xs:annotation>
            <xs:documentation>engine set-up</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="options" minOccurs="0"/>
                  <xs:element ref="conversiondefaults" minOccurs="0"/>
                  <xs:element ref="conversionsettings" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="lang" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="en"/>
                              <xs:enumeration value="pt"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
</xs:element>
<xs:element name="gui">
      <xs:annotation>
            <xs:documentation>gui set-up</xs:documentation>
```

```
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="options"/>
            </xs:sequence>
            <xs:attribute name="lang" type="xs:string"
use="required"/>
      </xs:complexType>
</xs:element>
<xs:element name="module">
      <xs:annotation>
            <xs:documentation>conversion module</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="conversion" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="abbreviation"/>
                              <xs:enumeration value="acronym"/>
                              <xs:enumeration value="network"/>
                              <xs:enumeration value="numeral"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="usermode" type="xs:string"
use="required"/>
      </xs:complexType>
</xs:element>
<xs:element name="options">
      <xs:annotation>
            <xs:documentation>config options</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="set" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="set">
      <xs:complexType>
            <xs:attribute name="name" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="abbreviation"/>
                              <xs:enumeration value="acronym"/>
                              <xs:enumeration value="usermodes"/>
                              <xs:enumeration value="ttsengine"/>
                              <xs:enumeration value="ttstype"/>
                              <xs:enumeration value="agent"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="value" type="xs:string"/>
      </xs:complexType>
</xs:element>
<xs:element name="definition">
      <xs:annotation>
```

```
            <xs:documentation>definition entry</xs:documentation>
      </xs:annotation>
      <xs:complexType mixed="true">
            <xs:attribute name="alias" type="xs:string"
use="required"/>
      </xs:complexType>
</xs:element>
<xs:element name="dictionary">
      <xs:annotation>
            <xs:documentation>dictionary set-up</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="definition" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="abbreviation"/>
                              <xs:enumeration value="acronym"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="type">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="chemistry"/>
                              <xs:enumeration value="money"/>
                              <xs:enumeration value="others"/>
                              <xs:enumeration value="physics"/>
                              <xs:enumeration value="social"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="lang" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="pt"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
      </xs:complexType>
</xs:element>
<xs:element name="result">
      <xs:annotation>
            <xs:documentation>result document</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="stats"/>
                  <xs:element ref="doc"/>
                  <xs:element ref="nav" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="stats">
      <xs:annotation>
            <xs:documentation>statistics block</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:sequence>
```

```
                  <xs:element ref="stat" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
<xs:element name="stat">
      <xs:annotation>
            <xs:documentation>statistics definition</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:attribute name="name" use="required">
                  <xs:simpleType>
                        <xs:restriction base="xs:string">
                              <xs:enumeration value="abbreviation"/>
                              <xs:enumeration value="acronym"/>
                              <xs:enumeration value="numeral"/>
                              <xs:enumeration value="math"/>
                              <xs:enumeration value="convtime"/>
                              <xs:enumeration value="network"/>
                        </xs:restriction>
                  </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="value" type="xs:string"
use="required"/>
      </xs:complexType>
</xs:element>
<xs:element name="doc">
      <xs:annotation>
            <xs:documentation>doc set-up</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:choice maxOccurs="unbounded">
                  <xs:element ref="txt"/>
                  <xs:element ref="mat"/>
            </xs:choice>
      </xs:complexType>
</xs:element>
<xs:element name="txt">
      <xs:annotation>
            <xs:documentation>text result</xs:documentation>
      </xs:annotation>
      <xs:complexType mixed="true">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                  <xs:element ref="abr"/>
                  <xs:element ref="num"/>
                  <xs:element ref="net"/>
                  <xs:element ref="acr"/>
            </xs:choice>
            <xs:attribute name="value" type="xs:string"
use="optional"/>
      </xs:complexType>
</xs:element>
<xs:element name="mat">
      <xs:annotation>
            <xs:documentation>math result</xs:documentation>
      </xs:annotation>
      <xs:complexType mixed="true">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                  <xs:element ref="node"/>
                  <xs:element ref="num"/>
            </xs:choice>
            <xs:attribute name="value" type="xs:string"
```

```
use="optional"/>
      </xs:complexType>
</xs:element>
<xs:element name="abr" type="xs:string">
      <xs:annotation>
            <xs:documentation>abbreviation</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="acr" type="xs:string">
      <xs:annotation>
      <xs:documentation>acronym</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="net" type="xs:string">
      <xs:annotation>
            <xs:documentation>network reference</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="num" type="xs:string">
      <xs:annotation>
            <xs:documentation>numeral</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="nav">
      <xs:annotation>
            <xs:documentation>navigation result</xs:documentation>
      </xs:annotation>
      <xs:complexType>
            <xs:choice maxOccurs="unbounded">
                  <xs:element ref="txt"/>
                  <xs:element ref="mat"/>
            </xs:choice>
      </xs:complexType>
</xs:element>
<xs:element name="node">
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="node" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="value" type="xs:string"
use="required"/>
      </xs:complexType>
</xs:element>
</xs:schema>
```

ANEXO D. Exemplos MathML do AudioMathENGINE

$$\sqrt{x}+\frac{1}{x-y}>y\times x$$

```
<math>
    <mrow>
      <msqrt><mi>x</mi></msqrt>
      <mo>+</mo>
      <mfrac>
        <mn>1</mn><mrow><mi>x</mi><mo>&minus;</mo><mi>y</mi></mrow>
      </mfrac>
      <mo>&gt;</mo><mi>y</mi><mo>&times;</mo><mi>x</mi>
    </mrow>
</math>
```

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

```
<math>
    <mrow>
      <mfrac>
        <mrow>
          <mo>&minus;</mo><mi>b</mi><mo>&plusmn;</mo>
          <msqrt>
            <mrow>
              <msup><mi>b</mi><mn>2</mn></msup>
              <mo>&minus;</mo><mn>4</mn><mi>a</mi><mi>c</mi>
            </mrow>
          </msqrt>
        </mrow>
        <mrow><mn>2</mn><mi>a</mi></mrow>
      </mfrac>
    </mrow>
</math>
```

$$\lim_{x\to\infty}\frac{x+\sin^2(x)}{x+1}=\lim_{x\to\infty}\frac{1+2\sin(x)\cos(x)}{1}=\lim_{x\to\infty}(1+\sin(2x))$$

```
<math>
    <mrow>
      <munder>
        <mrow><mi>lim</mi><mo>&ApplyFunction;</mo></mrow>
        <mrow><mi>x</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
      </munder>
      <mfrac>
        <mrow>
          <mi>x</mi><mo>+</mo>
          <msup>
            <mrow><mi>sin</mi><mo>&ApplyFunction;</mo></mrow>
            <mn>2</mn>
          </msup>
          <mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
        </mrow>
        <mrow><mi>x</mi><mo>+</mo><mn>1</mn></mrow>
      </mfrac>
      <mo>=</mo>
```

```
      <munder>
        <mrow><mi>lim</mi><mo>&ApplyFunction;</mo></mrow>
        <mrow><mi>x</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
      </munder>
      <mfrac>
        <mrow><mn>1</mn><mo>+</mo><mn>2</mn><mi>sin</mi>
          <mo>&ApplyFunction;</mo>
          <mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
          <mi>cos</mi><mo>&ApplyFunction;</mo>
          <mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
        </mrow>
        <mn>1</mn>
      </mfrac>
      <mo>=</mo>
      <munder>
        <mrow><mi>lim</mi><mo>&ApplyFunction;</mo></mrow>
        <mrow><mi>x</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
      </munder>
      <mrow>
        <mo>(</mo>
        <mrow><mn>1</mn><mo>+</mo><mi>sin</mi><mo>&ApplyFunction;</mo>
          <mrow>
            <mo>(</mo><mrow><mn>2</mn><mi>x</mi></mrow><mo>)</mo>
          </mrow>
        </mrow>
        <mo>)</mo>
      </mrow>
    </mrow>
</math>
```

$$\frac{1}{2}+\frac{1}{1+\frac{x}{3}}=\frac{1+\frac{x}{3}}{2}$$

```
<math>
    <mrow>
      <mfrac><mn>1</mn><mn>2</mn></mfrac>
      <mo>+</mo>
      <mfrac>
        <mn>1</mn>
        <mrow><mn>1</mn><mo>+</mo>
          <mfrac><mi>x</mi><mn>3</mn></mfrac>
        </mrow>
      </mfrac>
      <mo>=</mo>
      <mfrac>
        <mrow>
          <mn>1</mn>
          <mo>+</mo>
          <mfrac><mi>x</mi><mn>3</mn></mfrac>
        </mrow>
        <mn>2</mn>
      </mfrac>
    </mrow>
</math>
```

$\sqrt[(x-1)]{x+x^{\sqrt{5}-1}} > \sqrt{1-x}$

```
<math>
    <mrow>
      <mroot>
        <mrow>
          <mi>x</mi><mo>+</mo>
          <msup>
            <mi>x</mi>
            <mrow>
              <msqrt><mn>5</mn></msqrt><mo>&minus;</mo><mn>1</mn>
            </mrow>
          </msup>
        </mrow>
        <mrow>
          <mo stretchy="false">(</mo>
          <mi>x</mi><mo>&minus;</mo><mn>1</mn>
          <mo stretchy="false">)</mo>
        </mrow>
      </mroot>
      <mo>&gt;</mo>
      <msqrt>
        <mrow><mn>1</mn><mo>&minus;</mo><mi>x</mi></mrow>
      </msqrt>
    </mrow>
</math>
```

$x^{(a+b)^{(c-5)^2}}$

```
<math>
    <mrow>
      <msup>
        <mi>x</mi>
        <mrow>
          <msup>
            <mrow>
              <mo stretchy="false">(</mo>
              <mi>a</mi><mo>+</mo><mi>b</mi>
              <mo stretchy="false">)</mo>
            </mrow>
            <mrow>
              <msup>
                <mrow>
                  <mo stretchy="false">(</mo>
                  <mi>c</mi><mo>&minus;</mo><mn>5</mn>
                  <mo stretchy="false">)</mo>
                </mrow>
                <mn>2</mn>
              </msup>
            </mrow>
          </msup>
        </mrow>
      </msup>
    </mrow>
</math>
```

$$\sin\alpha \pm \sin\beta = 2\sin\frac{\alpha \pm \beta}{2}\cos\frac{\alpha \mp \beta}{2}$$

```
<math>
    <mrow>
      <mi>sin</mi><mo>&ApplyFunction;</mo><mi>&alpha;</mi>
      <mo>&plusmn;</mo>
      <mi>sin</mi><mo>&ApplyFunction;</mo><mi>&beta;</mi>
      <mo>=</mo>
      <mn>2</mn><mi>sin</mi><mo>&ApplyFunction;</mo>
      <mfrac>
        <mrow><mi>&alpha;</mi><mo>&plusmn;</mo><mi>&beta;</mi></mrow>
        <mn>2</mn>
      </mfrac>
      <mi>cos</mi><mo>&ApplyFunction;</mo>
      <mfrac>
        <mrow><mi>&alpha;</mi><mo>&mnplus;</mo><mi>&beta;</mi></mrow>
        <mn>2</mn>
      </mfrac>
    </mrow>
</math>
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

```
<math><mrow>
      <mrow>
        <mo>[</mo>
        <mrow>
          <mtable>
            <mtr><mtd><mn>1</mn></mtd><mtd><mn>2</mn></mtd></mtr>
            <mtr><mtd><mn>3</mn></mtd><mtd><mn>4</mn></mtd></mtr>
          </mtable>
        </mrow>
        <mo>]</mo>
      </mrow>
      <mo>&times;</mo>
      <mrow>
        <mo>[</mo>
        <mrow>
          <mtable>
            <mtr><mtd><mi>x</mi></mtd></mtr>
            <mtr><mtd><mi>y</mi></mtd></mtr>
          </mtable>
        </mrow>
        <mo>]</mo>
      </mrow>
      <mo>=</mo>
      <mrow>
        <mo>[</mo>
        <mrow>
          <mtable>
            <mtr><mtd><mn>5</mn></mtd></mtr>
            <mtr><mtd><mn>6</mn></mtd></mtr>
          </mtable>
        </mrow>
        <mo>]</mo>
      </mrow>
    </mrow></math>
```

$$A=\int_{1}^{2}\int_{-3x+6}^{4x-x^2} dydx+\int_{2}^{4}\int_{0}^{4x-x^2} dydx$$

```
<math>
    <mrow>
      <mi>A</mi><mo>=</mo>
      <mstyle displaystyle="true">
        <mrow>
          <munderover><mo>&int;</mo><mn>1</mn><mn>2</mn></munderover>
          <mrow>
            <mstyle displaystyle="true">
              <mrow>
                <munderover>
                  <mo>&int;</mo>
                  <mrow>
                    <mo>&minus;</mo><mn>3</mn><mi>x</mi><mo>+</mo>
                    <mn>6</mn>
                  </mrow>
                  <mrow>
                    <mn>4</mn><mi>x</mi><mo>&minus;</mo>
                    <msup><mi>x</mi><mn>2</mn></msup>
                  </mrow>
                </munderover>
                <mrow><mi>d</mi><mi>y</mi></mrow>
              </mrow>
            </mstyle>
            <mi>d</mi><mi>x</mi>
          </mrow>
        </mrow>
      </mstyle>
      <mo>+</mo>
      <mstyle displaystyle="true">
        <mrow>
          <munderover><mo>&int;</mo><mn>2</mn><mn>4</mn></munderover>
          <mrow>
            <mstyle displaystyle="true">
              <mrow>
                <munderover>
                  <mo>&int;</mo><mn>0</mn>
                  <mrow>
                    <mn>4</mn><mi>x</mi><mo>&minus;</mo>
                    <msup><mi>x</mi><mn>2</mn></msup>
                  </mrow>
                </munderover>
                <mrow><mi>d</mi><mi>y</mi></mrow>
              </mrow>
            </mstyle>
            <mi>d</mi><mi>x</mi>
          </mrow>
        </mrow>
      </mstyle>
    </mrow>
</math>
```

$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

```
<math>
    <mrow>
      <mi>sin</mi><mo>&ApplyFunction;</mo><mi>x</mi><mo>=</mo>
      <mstyle displaystyle="true">
        <munderover>
          <mo>&sum;</mo>
          <mrow><mi>n</mi><mo>=</mo><mn>0</mn></mrow>
          <mi>&infin;</mi>
        </munderover>
        <mrow>
          <msup>
            <mrow>
              <mo stretchy="false">(</mo>
              <mo>&minus;</mo><mn>1</mn>
              <mo stretchy="false">)</mo>
            </mrow>
            <mi>n</mi>
          </msup>
          <mfrac>
            <mrow>
              <msup>
                <mi>x</mi>
                <mrow><mn>2</mn><mi>n</mi><mo>+</mo><mn>1</mn></mrow>
              </msup>
            </mrow>
            <mrow>
              <mo stretchy="false">(</mo>
              <mn>2</mn><mi>n</mi><mo>+</mo><mn>1</mn>
              <mo stretchy="false">)</mo>
              <mo>!</mo>
            </mrow>
          </mfrac>
        </mrow>
      </mstyle>
    </mrow>
</math>
```

ANEXO E. Corpus matemático

Álgebra Simples

1. $x+5,4+z+1=3$

chis mais cinco vírgula quatro mais zê mais um igual a três

2. $1+2+3+\ldots+8+9+10>0$

um mais dois mais três mais reticências mais oito mais nove mais dez maior do que zero

3. $(x+y)+z=0$

abrir parêntesis chis mais ípsilon fechar parêntesis mais zê igual a zero

4. $(x-y)-z=0$

abrir parêntesis chis menos ípsilon fechar parêntesis menos zê igual a zero

5. $\frac{1}{2}-\sqrt{2}\geq 0$

inicio de fracção, numerador: um a dividir por denominador: dois fim de fracção menos raiz quadrada de dois fim de radicando maior ou igual a zero

6. $0,01-(x-y-10)=0,02-(z-w-20)$

zero vírgula zero um menos abrir parêntesis chis menos ípsilon menos dez fechar parêntesis igual a zero vírgula zero dois menos abrir parêntesis zê menos dabliu menos vinte fechar parêntesis

7. $x\times y\times\ldots\times z$

chis vezes ípsilon vezes reticências vezes zê

8. $x+y=5\times(x-y)$

chis mais ípsilon igual a cinco vezes abrir parêntesis chis menos ípsilon fechar parêntesis

9. $\frac{x}{2}+y=4\times x\times y$

inicio de fracção, numerador: chis a dividir por denominador: dois fim de fracção mais ípsilon igual a quatro vezes chis vezes ípsilon

10. $\sqrt{x}+\frac{1}{x-y}>y\times x$

raiz quadrada de chis fim de radicando mais inicio de fracção, numerador: um a dividir por denominador: chis menos ípsilon fim de fracção maior do que ípsilon vezes chis

11. $\lim_{x\to+\infty}\frac{x+\sin^2(x)}{x+1}$

limite quando chis tende para mais infinito de: inicio de fracção, numerador: chis mais seno ao quadrado de abrir parêntesis chis fechar parêntesis a dividir por denominador: chis mais um fim de fracção

Fracções

12. $\frac{1}{2}$

inicio de fracção, numerador: um a dividir por denominador: dois fim de fracção

13. $\frac{dy}{dx}$

inicio de fracção, numerador: dê ípsilon a dividir por denominador: dê chis fim de fracção

14. $\frac{\frac{a}{b}}{\frac{c}{d}}$

inicio de fracção, numerador: inicio de fracção, numerador: à a dividir por denominador: bê fim de fracção a dividir por denominador: inicio de fracção, numerador: cê a dividir por denominador: dê fim de fracção fim de fracção

15. $\frac{1+\sqrt{5}}{2}$

inicio de fracção, numerador: um mais raiz quadrada de cinco fim de radicando a dividir por denominador: dois fim de fracção

16. $\frac{1}{2}+\frac{1}{1+\frac{x}{3}}=\frac{1+\frac{x}{3}}{2}$

inicio de fracção, numerador: um a dividir por denominador: dois fim de fracção mais inicio de fracção, numerador: um a dividir por denominador: um mais inicio de fracção, numerador: chis a dividir por denominador: três fim de fracção fim de fracção igual a inicio de fracção, numerador: um mais inicio de fracção, numerador: chis a dividir por denominador: três fim de fracção a dividir por denominador: dois fim de fracção

17. $\frac{\ln x}{y}>\frac{\sin x}{\pi y}$

inicio de fracção, numerador: logaritmo neperiano de chis a dividir por denominador: ípsilon fim de fracção maior do que inicio de fracção, numerador: seno de chis a dividir por denominador: pi minúsculo ípsilon fim de fracção

18. $\frac{x}{3-x}-x>0$

inicio de fracção, numerador: chis a dividir por denominador: três menos chis fim de fracção menos chis maior do que zero

Raízes

19. $\sqrt{a}$

raiz quadrada de à fim de radicando

20. $\sqrt{x-b}$

raiz quadrada de chis menos bê fim de radicando

21. $\sqrt[x]{a+b}$

raiz de índice: chis fim de índice e base: à mais bê fim de radicando

22. $\sqrt[3]{a+b}-c$

raiz cúbica de à mais bê fim de radicando menos cê

23. $\sqrt{2+\sqrt{5}-\sqrt[3]{8}}$

raiz quadrada de dois mais raiz quadrada de cinco fim de radicando menos raiz cúbica de oito fim de radicando fim de radicando

24. $\sqrt[x-5]{2-z}+35$

raiz de índice: chis menos cinco fim de índice e base: dois menos zê fim de radicando mais trinta e cinco

25. $\frac{-b\pm\sqrt{b^2-4ac}}{2a}$

inicio de fracção, numerador: menos bê mais ou menos raiz quadrada de bê ao quadrado menos quatro à cê fim de radicando a dividir por denominador: dois à fim de fracção

26. $\sqrt[(x-1)]{x+x^{\sqrt{5}-1}}>\sqrt{1-x}$

raíz de índice: abrir parêntesis chis menos um fechar parêntesis fim de índice e base: chis mais chis elevado ao expoente: raiz quadrada de cinco fim de radicando menos um fim de expoente fim de radicando maior do que raiz quadrada de um menos chis fim de radicando

27. $\sqrt[3]{\sqrt{y+3\sqrt{5-x}}-2}=x+y$

raíz cúbica de raiz quadrada de ípsilon mais três raiz quadrada de cinco menos chis fim de radicando fim de radicando menos dois fim de radicando igual a chis mais ípsilon

Potências

28. 5^{23}

cinco elevado ao expoente: vinte e três fim de expoente

29. 2^4

dois elevado ao expoente: quatro fim de expoente

30. $(-4)^x$

abrir parêntesis menos quatro fechar parêntesis elevado ao expoente: chis fim de expoente

31. 2^{x+5}

dois elevado ao expoente: chis mais cinco fim de expoente

32. 3^{x^2}

três elevado ao expoente: chis ao quadrado fim de expoente

33. $x^{(a+b)^{(c-5)^2}}$

chis elevado ao expoente: abrir parêntesis à mais bê fechar parêntesis elevado ao expoente: abrir parêntesis cê menos cinco fechar parêntesis ao quadrado fim de expoente fim de expoente

34. $(x+5)^3-10$

abrir parêntesis chis mais cinco fechar parêntesis ao cubo menos dez

35. $(x+5)^3-10=0$

abrir parêntesis chis mais cinco fechar parêntesis ao cubo menos dez igual a zero

36. $5-(x-3)^5$

cinco menos abrir parêntesis chis menos três fechar parêntesis elevado ao expoente: cinco fim de expoente

37. $x^{(y-2)}+y^2>x^y$

chis elevado ao expoente: abrir parêntesis ípsilon menos dois fechar parêntesis fim de expoente mais ípsilon ao quadrado maior do que chis elevado ao expoente: ípsilon fim de expoente

38. $(x-3^2)^{(x-2^3)}-5=0$

abrir parêntesis chis menos três ao quadrado fechar parêntesis elevado ao expoente: abrir parêntesis chis menos dois ao cubo fechar parêntesis fim de expoente menos cinco igual a zero

39. $(a+b)^{1/3}>\frac{a^2}{b^3}$

abrir parêntesis à mais bê fechar parêntesis elevado ao expoente: inicio de fracção, numerador: um a dividir por denominador: três fim de fracção fim de expoente maior do que inicio de fracção, numerador: à ao quadrado a dividir por denominador: bê ao cubo fim de fracção

Trigonometria

40. $\sin\alpha=\cos\mu$

seno de alfa minúsculo igual a cosseno de miu minúsculo

41. $\sin^2\alpha+\cos^2\alpha=1$

seno ao quadrado de alfa minúsculo mais cosseno ao quadrado de alfa minúsculo igual a um

42. $\cos 3\alpha=4\cos^3\alpha-3\cos\alpha$

cosseno de três alfa minúsculo igual a quatro cosseno ao cubo de alfa minúsculo menos três cosseno de alfa minúsculo

43. $\sin^3 \alpha = \frac{1}{4}(3\sin\alpha - \sin 3\alpha)$

seno ao cubo de alfa minúsculo igual a inicio de fracção, numerador: um a dividir por denominador: quatro fim de fracção abrir parêntesis três seno de alfa minúsculo menos seno de três alfa minúsculo fechar parêntesis

44. $\arcsin\theta - \left(\frac{\pi}{2} - \theta\right) = 0$

arco-seno de téta minúsculo menos abrir parêntesis inicio de fracção, numerador: pi minúsculo a dividir por denominador: dois fim de fracção menos téta minúsculo fechar parêntesis igual a zero

45. $\cos^2 x$

cosseno ao quadrado de chis

46. $\sin\alpha \pm \sin\beta = 2\sin\frac{\alpha \pm \beta}{2}\cos\frac{\alpha \mp \beta}{2}$

seno de alfa minúsculo mais ou menos seno de beta minúsculo igual a dois seno de inicio de fracção, numerador: alfa minúsculo mais ou menos beta minúsculo a dividir por denominador: dois fim de fracção cosseno de inicio de fracção, numerador: alfa minúsculo menos ou mais beta minúsculo a dividir por denominador: dois fim de fracção

Matrizes e Sistemas de (in)equações

47. $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$

tabela ou matriz primeira linha primeira coluna à de índice onze fim de índice fim de coluna segunda coluna à de índice doze fim de índice fim de coluna fim de linha segunda linha primeira coluna à de índice vinte e um fim de índice fim de coluna segunda coluna à de índice vinte e dois fim de índice fim de coluna fim de linha fim de tabela ou matriz

48. $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

tabela ou matriz primeira linha primeira coluna um fim de coluna segunda coluna dois fim de coluna fim de linha segunda linha primeira coluna três fim de coluna segunda coluna quatro fim de coluna fim de linha fim de tabela ou matriz vezes tabela ou matriz primeira linha primeira coluna chis fim de coluna fim de linha segunda linha primeira coluna ípsilon fim de coluna fim de linha fim de tabela ou matriz igual a tabela ou matriz primeira linha primeira coluna cinco fim de coluna fim de linha segunda linha primeira coluna seis fim de coluna fim de linha fim de tabela ou matriz

49. $\begin{cases} x - y = 2 \\ x + y = 5 \end{cases}$

abrir chaveta tabela ou matriz primeira linha primeira coluna chis menos ípsilon igual a dois fim de coluna fim de linha segunda linha primeira coluna chis mais ípsilon igual a cinco fim de coluna fim de linha fim de tabela ou matriz

50. $\begin{cases} a^2 + b^2 + c^2 > 1 \\ b^2 - \sqrt{c} = 0 \\ \dfrac{a}{c+1} = 5 \end{cases}$

abrir chaveta tabela ou matriz primeira linha primeira coluna à ao quadrado mais bê ao quadrado mais cê ao quadrado maior do que um fim de coluna fim de linha segunda linha primeira coluna bê ao quadrado menos raiz quadrada de cê fim de radicando igual a zero fim de coluna fim de linha terceira linha primeira coluna inicio de fracção, numerador: à a dividir por denominador: cê mais um fim de fracção igual a cinco fim de coluna fim de linha fim de tabela ou matriz

Integrais

51. $\int dx = x + k$

integral de dê chis igual a chis mais capa

52. $\int \frac{dx}{\sqrt{1-x^2}} = \arcsin(x) + k$

integral de inicio de fracção, numerador: dê chis a dividir por denominador: raiz quadrada de um menos chis ao quadrado fim de radicando fim de fracção igual a arco-seno de abrir parêntesis chis fechar parêntesis mais capa

53. $\int_{1-x}^{x+1} \frac{t-1}{t} dt$

integral com limite inferior: um menos chis fim de limite e limite superior: chis mais um fim de limite de: inicio de fracção, numerador: tê menos um a dividir por denominador: tê fim de fracção dê tê

54. $\int_{1}^{2} \sqrt{1+e^{-2x}} dx$

integral com limite inferior: um fim de limite e limite superior: dois fim de limite de: raiz quadrada de um mais é elevado ao expoente: menos dois chis fim de expoente fim de radicando dê chis

55. $\int_B f(x,y,z)dV = \int_a^b \int_c^d \int_u^v f(x,y,z)dzdydx$

integral com limite inferior: bê fim de limite de: éfe abrir parêntesis chis vírgula ípsilon vírgula zê fechar parêntesis dê vê igual a integral com limite inferior: à fim de limite e limite superior: bê fim de limite de: integral com limite inferior: cê fim de limite e limite superior: dê fim de limite de: integral com limite inferior: u fim de limite e limite superior: vê fim de limite de: éfe abrir parêntesis chis vírgula ípsilon vírgula zê fechar parêntesis dê zê dê ípsilon dê chis

56. $A = \int_{1}^{2} \int_{-3x+6}^{4x-x^2} dydx + \int_{2}^{4} \int_{0}^{4x-x^2} dydx$

à igual a integral com limite inferior: um fim de limite e limite superior: dois fim de limite de: integral com limite inferior: menos três chis mais seis fim de limite e limite superior: quatro chis menos chis ao quadrado fim de limite de: dê ípsilon dê chis mais integral com limite inferior: dois fim de limite e limite superior: quatro fim de limite de: integral com limite inferior: zero fim de limite e limite superior: quatro chis menos chis ao quadrado fim de limite de: dê ípsilon dê chis

57. $\int_{C} (z + \frac{1}{z})^3 dz$

integral com limite inferior: cê fim de limite de: abrir parêntesis zê mais inicio de fracção, numerador: um a dividir por denominador: zê fim de fracção fechar parêntesis ao cubo dê zê

58. $\int_{[a,b]} f(z)dz$

integral com limite inferior: abrir parêntesis rectos à vírgula bê fechar parêntesis rectos fim de limite de: éfe abrir parêntesis zê fechar parêntesis dê zê

Derivadas

59. $y''' - 2y'' + 2y' = 2\cos 4x$

ípsilon linha linha linha menos dois ípsilon linha linha mais dois ípsilon linha igual a dois cosseno de quatro chis

60. $\frac{\partial y}{\partial t} - \frac{\partial y}{\partial x} = 0$

inicio de fracção, numerador: derivada parcial ípsilon a dividir por denominador: derivada parcial tê fim de fracção menos inicio de fracção, numerador: derivada parcial ípsilon a dividir por denominador: derivada parcial chis fim de fracção igual a zero

61. $\frac{dy}{dx} = f(x, y)$

inicio de fracção, numerador: dê ípsilon a dividir por denominador: dê chis fim de fracção igual a éfe abrir parêntesis chis vírgula ípsilon fechar parêntesis

62. $\frac{\partial^2 y}{\partial x^2} = \frac{\partial y}{\partial t}$

inicio de fracção, numerador: derivada parcial ao quadrado ípsilon a dividir por denominador: derivada parcial chis ao quadrado fim de fracção igual a inicio de fracção, numerador: derivada parcial ípsilon a dividir por denominador: derivada parcial tê fim de fracção

63. $y'' = \sqrt{1 + y'^2}$

ípsilon linha linha igual a raiz quadrada de um mais ípsilon linha ao quadrado fim de radicando

64. $\frac{d}{dx}\left[(1-x^2)y'\right]=0$

inicio de fracção, numerador: dê a dividir por denominador: dê chis fim de fracção abrir parêntesis rectos abrir parêntesis um menos chis ao quadrado fechar parêntesis ípsilon linha fechar parêntesis rectos igual a zero

65. $\left.\frac{\partial(t,x)}{\partial x}\right|_{x=0}=-1$

inicio de fracção, numerador: derivada parcial abrir parêntesis tê vírgula chis fechar parêntesis a dividir por denominador: derivada parcial chis fim de fracção barra ao alto de índice chis igual a zero fim de índice igual a menos um

Somatórios

66. $\sum a_n \neq \sum b_n$

somatório de à de índice éne fim de índice diferente de somatório de bê de índice éne fim de índice

67. $\sum \frac{n!}{2^{2n}}$

somatório de inicio de fracção, numerador: éne factorial a dividir por denominador: dois elevado ao expoente: dois éne fim de expoente fim de fracção

68. $\sum_0^{\infty}(-1)^n\frac{z^{2n}}{(2n)!}$

somatório com limite inferior: zero fim de limite e limite superior: infinito fim de limite de: abrir parêntesis menos um fechar parêntesis elevado ao expoente: éne fim de expoente inicio de fracção, numerador: zê elevado ao expoente: dois éne fim de expoente a dividir por denominador: abrir parêntesis dois éne fechar parêntesis factorial fim de fracção

69. $\sum_{n=0}^{\infty}1/2^n=2-(1/2)^{n-1}$

somatório com limite inferior: éne igual a zero fim de limite e limite superior: infinito fim de limite de: um a dividir por dois elevado ao expoente: éne fim de expoente igual a dois menos abrir parêntesis um a dividir por dois fechar parêntesis elevado ao expoente: éne menos um fim de expoente

70. $\sum_{n=1}^{\infty}\frac{\sin(n\pi)}{n}$

somatório com limite inferior: éne igual a um fim de limite e limite superior: infinito fim de limite de: inicio de fracção, numerador: seno de abrir parêntesis éne pi minúsculo fechar parêntesis a dividir por denominador: éne fim de fracção

71. $$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

somatório com limite inferior: éne igual a um fim de limite e limite superior: infinito fim de limite de: inicio de fracção, numerador: um a dividir por denominador: éne ao quadrado fim de fracção igual a inicio de fracção, numerador: pi minúsculo ao quadrado a dividir por denominador: seis fim de fracção

72. $$z\cosh(z^2) = \sum_{n=0}^{\infty} \frac{z^{4n+1}}{(2n)!}$$

zê cosseno hiperbólico de abrir parêntesis zê ao quadrado fechar parêntesis igual a somatório com limite inferior: éne igual a zero fim de limite e limite superior: infinito fim de limite de: inicio de fracção, numerador: zê elevado ao expoente: quatro éne mais um fim de expoente a dividir por denominador: abrir parêntesis dois éne fechar parêntesis factorial fim de fracção

73. $$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

seno de chis igual a somatório com limite inferior: éne igual a zero fim de limite e limite superior: infinito fim de limite de: abrir parêntesis menos um fechar parêntesis elevado ao expoente: éne fim de expoente inicio de fracção, numerador: chis elevado ao expoente: dois éne mais um fim de expoente a dividir por denominador: abrir parêntesis dois éne mais um fechar parêntesis factorial fim de fracção

74. $$\sum_{i=2}^{4} \left(ax^i + bx^{i-2} \right) + \sum_{i=1}^{3} \left(ax^{i-1} + bx^{i-1} \right)$$

somatório com limite inferior: i igual a dois fim de limite e limite superior: quatro fim de limite de: abrir parêntesis à chis elevado ao expoente: i fim de expoente mais bê chis elevado ao expoente: i menos dois fim de expoente fechar parêntesis mais somatório com limite inferior: i igual a um fim de limite e limite superior: três fim de limite de: abrir parêntesis à chis elevado ao expoente: i menos um fim de expoente mais bê chis elevado ao expoente: i menos um fim de expoente fechar parêntesis

ANEXO F. Questionário de Interpretação Escrita

Teste 1 – Nível de Ensino secundário

Questão 1)

chis mais cinco vírgula quatro mais zê mais um igual a três

$x + 5,4 + z + 1 = 3$

Questão 2)

abrir parêntesis chis mais ípsilon fechar parêntesis mais zê igual a zero

$(x + y) + z = 0$

Questão 3)

inicio de fracção, numerador: um a dividir por denominador: dois fim de fracção menos raiz quadrada de dois fim de radicando maior ou igual a zero

$\frac{1}{2} - \sqrt{2} \geq 0$

Questão 4)

zero vírgula zero um menos abrir parêntesis chis menos ípsilon menos dez fechar parêntesis igual a zero vírgula zero dois menos abrir parêntesis zê menos dabliu menos vinte fechar parêntesis

$0,01 - (x - y - 10) = 0,02 - (z - w - 20)$

Questão 5)

raiz quadrada de chis fim de radicando mais inicio de fracção, numerador: um a dividir por denominador: chis menos ípsilon fim de fracção maior do que ípsilon vezes chis

$\sqrt{x} + \frac{1}{x - y} > y \times x$

Questão 6)

inicio de fracção, numerador: inicio de fracção, numerador: à a dividir por denominador: bê fim de fracção a dividir por denominador: inicio de fracção, numerador: cê a dividir por denominador: dê fim de fracção fim de fracção

$\frac{\frac{a}{b}}{\frac{c}{d}}$

Questão 7)

inicio de fracção, numerador: chis a dividir por denominador: três menos chis fim de fracção menos chis maior do que zero

$\frac{x}{3 - x} - x > 0$

Questão 8)

inicio de fracção, numerador: menos bê mais ou menos raiz quadrada de bê ao quadrado menos quatro à cê fim de radicando a dividir por denominador: dois à fim de fracção

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Questão 9)

dois elevado ao expoente: quatro fim de expoente

$$2^4$$

Questão 10)

abrir parêntesis chis mais cinco fechar parêntesis ao cubo menos dez igual a zero

$$(x+5)^3 - 10 = 0$$

Teste 2 – Nível de Ensino Superior

Questão 1)

limite quando chis tende para infinito de: inicio de fracção, numerador: chis mais seno ao quadrado de abrir parêntesis chis fechar parêntesis a dividir por denominador: chis mais um fim de fracção igual a limite quando chis tende para infinito de: inicio de fracção, numerador: um mais dois seno de abrir parêntesis chis fechar parêntesis co-seno de abrir parêntesis chis fechar parêntesis a dividir por denominador: um fim de fracção igual a limite quando chis tende para infinito de: abrir parêntesis um mais seno de abrir parêntesis dois chis fechar parêntesis fechar parêntesis

$$\lim_{x\to\infty} \frac{x + \sin^2(x)}{x+1} = \lim_{x\to\infty} \frac{1 + 2\sin(x)\cos(x)}{1} = \lim_{x\to\infty} \left(1 + \sin(2x)\right)$$

Questão 2)

inicio de fracção, numerador: um a dividir por denominador: dois fim de fracção mais inicio de fracção, numerador: um a dividir por denominador: um mais inicio de fracção, numerador: chis a dividir por denominador: três fim de fracção fim de fracção igual a inicio de fracção, numerador: um mais inicio de fracção, numerador: chis a dividir por denominador: três fim de fracção a dividir por denominador: dois fim de fracção

$$\frac{1}{2} + \frac{1}{1 + \frac{x}{3}} = \frac{1 + \frac{x}{3}}{2}$$

Questão 3)

raiz de índice: abrir parêntesis chis menos um fechar parêntesis fim de índice e base: chis mais chis elevado ao expoente: raiz quadrada de cinco fim de radicando menos um fim de expoente fim de radicando maior do que raiz quadrada de um menos chis fim de radicando

$$\sqrt[(x-1)]{x + x^{\sqrt{5}-1}} > \sqrt{1-x}$$

Questão 4)

chis elevado ao expoente: abrir parêntesis à mais bê fechar parêntesis elevado ao expoente: abrir parêntesis cê menos cinco fechar parêntesis ao quadrado fim de expoente fim de expoente

$$x^{(a+b)^{(c-5)^2}}$$

Questão 5)

abrir parêntesis chis menos três ao quadrado fechar parêntesis elevado ao expoente: abrir parêntesis chis menos dois ao cubo fechar parêntesis fim de expoente menos cinco igual a zero

$$(x-3^2)^{(x-2^3)}-5=0$$

Questão 6)

seno de alfa minúsculo mais ou menos seno de beta minúsculo igual a dois seno de inicio de fracção, numerador: alfa minúsculo mais ou menos beta minúsculo a dividir por denominador: dois fim de fracção cosseno de inicio de fracção, numerador: alfa minúsculo menos ou mais beta minúsculo a dividir por denominador: dois fim de fracção

$$\sin\alpha \pm \sin\beta = 2\sin\frac{\alpha \pm \beta}{2}\cos\frac{\alpha \mp \beta}{2}$$

Questão 7)

tabela ou matriz primeira linha primeira coluna um fim de coluna segunda coluna dois fim de coluna fim de linha segunda linha primeira coluna três fim de coluna segunda coluna quatro fim de coluna fim de linha fim de tabela ou matriz vezes tabela ou matriz primeira linha primeira coluna chis fim de coluna fim de linha segunda linha primeira coluna ípsilon fim de coluna fim de linha fim de tabela ou matriz igual a tabela ou matriz primeira linha primeira coluna cinco fim de coluna fim de linha segunda linha primeira coluna seis fim de coluna fim de linha fim de tabela ou matriz

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\times\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}5\\ 6\end{bmatrix}$$

Questão 8)

integral de inicio de fracção, numerador: dê chis a dividir por denominador: raiz quadrada de um menos chis ao quadrado fim de radicando fim de fracção igual a arco-seno de abrir parêntesis chis fechar parêntesis mais capa

$$\int \frac{dx}{\sqrt{1-x^2}} = \arcsin(x) + k$$

Questão 9)

à igual a integral com limite inferior: um fim de limite e limite superior: dois fim de limite de: integral com limite inferior: menos três chis mais seis fim de limite e limite superior: quatro chis menos chis ao quadrado fim de limite de: dê ípsilon dê chis mais integral com limite inferior: dois fim de limite e limite superior: quatro fim de limite de: integral com limite inferior: zero fim de limite e limite superior: quatro chis menos chis ao quadrado fim de limite de: dê ípsilon dê chis

$$A = \int_{1}^{2} \int_{-3x+6}^{4x-x^2} dydx + \int_{2}^{4} \int_{0}^{4x-x^2} dydx$$

Questão 10)

seno de chis igual a somatório com limite inferior: éne igual a zero fim de limite e limite superior: infinito fim de limite de: abrir parêntesis menos um fechar parêntesis elevado ao expoente: éne fim de expoente inicio de fracção, numerador: chis elevado ao expoente: dois éne mais um fim de expoente a dividir por denominador: abrir parêntesis dois éne mais um fechar parêntesis factorial fim de fracção

$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

ANEXO G. Questionário de Interpretação Oral

Teste 1 – Nível de Ensino secundário

Som	Teve de ouvir mais do que 3 vezes para desenhar a fórmula? (S ou N)	Fórmula
		$x+5,4+z+1=3$
		$(x+y)+z=0$
		$\frac{1}{2}-\sqrt{2}\geq 0$
		$\frac{x}{2}+y=4\times x\times y$
		$\sqrt{x}+\frac{1}{x-y}>y\times x$
		$\sqrt{x-b}$
		$(-4)^{x}$
		2^{x+5}

Teste 2 – Nível de Ensino Superior

Som	Teve de ouvir mais do que 3 vezes para desenhar a fórmula? (S ou N)	Fórmula
		$\sqrt[(x-1)]{x+x^{\sqrt{5}-1}}>\sqrt{1-x}$
		$(x-3^2)^{(x-2^3)}-5=0$
		$(a+b)^{1/3}>\frac{a^2}{b^3}$
		$\int\frac{dx}{\sqrt{1-x^2}}=\arcsin(x)+k$
		$A=\int_{1}^{2}\int_{-3x+6}^{4x-x^2}dydx+\int_{2}^{4}\int_{0}^{4x-x^2}dydx$
		$\sum_{n=1}^{\infty}\frac{\sin(n\pi)}{n}$

ANEXO H. Documento usado no teste de navegação

O Prof. João olhou para os alunos e procurou 4 fórmulas para o trabalho de casa na pág. 123 do caderno de exercícios. A 1ª fórmula foi:

$$\frac{x-2^2}{(x-y)}+\frac{\left(x+5^3\right)^{(x-6)^2}}{3}$$

A 2ª fórmula foi:

$$\frac{\sqrt{b^2-4ac}}{\sqrt{a^2+b^2}+\frac{a^{(c-1)}-1}{b}}$$

A 3ª fórmula que ele ditou foi:

$$a+b+\frac{a-b}{\frac{ab}{a+b}}+\frac{\frac{a}{2}}{\frac{3}{b}}=a^b-a^2b^3$$

E por fim:

$$\sum_{n=1}^{+\infty}\frac{2\times n}{(\pi n)^5}+\frac{1}{2}\times\int_A(4n)-n^3dn$$

Concluído isto, a aula acabou e os alunos saíram apressados.
Eram 2 da tarde e ainda havia muito que fazer.