Universidade do Minho
Conselho de Cursos de Engenharia
Mestrado de Informática

Dissertação

Ano lectivo 2007/2008

departamento de informática

Contribuição para a Manipulação de Conteúdo em MusicXML

Samuel Soares da Silva

Orientador:José Nuno Oliveira
José João Dias de Almeida

Novembro, 2008

Data de Recepção	
Responsável	
Avaliação	
Observações	

Manipulação de Música em MusicXML

Samuel Soares da Silva

Novembro, 2008

Resumo

A música pode ser tratada como um conjunto de elementos sonoros, ou como um conjunto de elementos gráficos. Os elementos sonoros abordam aspectos relacionados com a reprodução em áudio enquanto que os elementos gráficos abordam apresentação da notação musical.

Existem muitos formatos para cada perspectiva, e alguns conseguem processar ambos. Formatos de intercâmbio surgiram devido ao crescente número de formatos e à necessidade de trabalhar com diferentes ferramentas e diferentes formatos simultâneamente. Recentemente o MusicXML revelou-se como um formato prático, porque consegue manipular detalhes sonoros e gráficos e usa o XML que é usado pelas principais ferramentas de notação musical.

Por o formato MusicXML ter bastante aceitação em ferramentas de música profissionais, apresenta-se a biblioteca musicxml, implementada em Haskell. Esta biblioteca apresenta a mesma estrutura presente na especificação do formato MusicXML, tendo implementado parsing, pretty-printing e um tipo em Haskell equivalente ao formato MusicXML. Também se apresentam funções que permitem ler ou escrever documentos MusicXML. Esta tradução do DTD para Haskell foi orientada para melhorar a performance da utilização da memória.

No entanto, tanto a especificação do MusicXML como a sua implementação na biblioteca musicxml tem demasiados detalhes para certas operações. Por ser complexo e difícil de processar, apresenta-se uma estratificação do formato. A estratificação do MusicXML em seis níveis, estando a complexidade ordenada de forma crescente, sendo o nível 5 equivale ao formato original. A simplificação efectuada refere-se À complexidade musical e a detalhes de impressão.

A estratificação do MusicXML faz parte da biblioteca hamusic, onde se apresenta uma definição de música abstracta. Esta definição de música também usa vários níveis, sendo alguns níveis similares ao níveis de estratificação do MusicXML. Na mesma biblioteca, apresentam-se aplicações ajudam a análise musical, tal como o `MusicCount` e o `MusicTranslate`. A primeira

aplicação efectua uma contagem de elementos musicais presentes enquanto que o segundo traduz notação musical entre diversos formatos.

Parte do trabalho desenvolvido e apresentado nesta dissertação será usado numa disciplina da Licenciatura de Música. As bibliotecas apresentadas estão disponíveis no `hackage`, repositório de bibliotecas em Haskell, por ser útil a sua divulgação.

Abstract

The music can be handled as a set of sound elements, that address aspects related to playing audio, or as a set of graphic elements, addressing visualization of musical notation. There are many digital formats for music representation in each view and some formats can process both. Exchange formats arise due to the growing number of formats, and the need to work serveral tools and diferents formats at same time. Recently MusicXML reveals itself practical because it can handle sound and graphic details, and use XML that is widely adopted by the main music notational tools.

We present Haskell library musicxml because MusicXML format is largely accepted by professional music tools. This library has the same structure of MusicXML format. This library implements parsing, pretty-printing and an Haskell type equivalent to MusicXML format. It also have functions that allow users to read or write MusicXML documents. This translation from DTD to Haskell was aimed to improve memory performance.

However, both the specification of MusicXML as its implementation in musicxml library has too many details for certain operations. Being complex and difficult to process, we implement a stratification of the format. The stratification of MusicXML has six levels, sorted by increasing complexity, with level 5 being the equivalent of original format. Simplification achievied refers to musical complexity and print details.

The stratification of MusicXML is implemented on library hamusic, which also defines abstract music. This definition of music also uses several levels, being some of them similar to MusicXML stratification levels. Hamusic library have applications to help music analysis, such as MusicCount and MusicTranslate. The first application makes a count of musical elements presents while the second translates musical notation between various formats.

Part of the work presented in this thesis will be used in a module from Music Graduation. These libraries are available in hackage, a repository of Haskell library, because its disclosure is considered useful.

Agradecimentos

O trabalho apresentado neste documento não seria possível sem a ajuda de algumas pessoas.

Gostaria de agradecer aos orientadores, ao Professor José João Almeida e ao Professor José Nuno Oliveira pelo apoio. Pelos comentários pertinentes e pela disponibilidade.

Agradeço ao Dr. Silas Pego e sua família. Pela hospitalidade e me integrarem na família, sendo como o filho mais novo.

Aos meus pais que sempre me apoiam e apoiaram. Pelo o amor, compreensão e tempo dispendido.

Agradeço aos meus colegas Vitor Manuel Sá, Avelino Rego e Marco Caldas pelo apoio, revisão e sugestões.

Agradeço também a inúmeras pessoas que me rodearam nos últimos anos pelos comentários e conhecimento disponibilizado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

Capítulo 1

Introdução

A descrição da música envolve uma notação complexa e rica. A sua evolução permitiu a evolução da notação musical aumentando a sua expressividade. Na notação musical existe a expressividade em termos sonoros e gráficos. Uma forte tende a privilegiar um dos lados. Geralmente a perspectiva sonora é predominante nas ferramentas de notação musical.

A notação musical é a arte de organizar no espaço temporal sons, usando elementos musicais como a melodia e o ritmo. A melodia contém informação sobre a frequência sonora, assim como o ritmo contém informação sobre a duração de cada evento musical.

1.1 Enquadramento

No estudo de música, actualmente são utilizadas ferramentas WYSIWYG[1] auxiliando os utilizadores na edição, reprodução, visualização e análise musical. Cada ferramenta usa o seu formato próprio para representar a música em suporte digital. Esta facto traz o inconveniente de não ser fácil a tradução entre os formatos existentes, por que até a maioria são formatos binários. Assim, muitas vezes é necessária a edição do resultado após uma tradução devido a erros de tradução, principalmente em aspectos de notação musical. Para aumentar a portabilidade entre formatos de representação musicais, apareceram formatos com o propósito de servirem de comunicação entre formatos existentes. Quando surgiu o MusicXML[7, 5], baseado em dois formatos académicos existentes, teve aceitação, em parte devido às ferramentas de importação/exportação existente para as ferramentas mais utilizadas.

[1] A designação WYSIWYG, What You See Is What You Get, é uma designação de ferramentas em que no momento da edição se torna possível visualizar o resultado.

O formato *MusicXML* está a ser adoptado pelo software de edição de música, como um formato de intercâmbio entre várias plataformas. A falta de documentação formal sobre este formato, sendo apenas a documentação técnica apresentada em forma de comentários quando o formato é especificado.

No ano lectivo 2008/2009 vai ser leccionada, na Licenciatura de Música na Universidade do Minho, uma disciplina que pretende abordar a análise musical com recurso a ferramentas informáticas. Supõe-se que os alunos tem um repertório musical em vários formatos digitais. Com recurso formatos de intercâmbio, como por exemplo, o MusicXML[57] é possível efectuar conversões entre vários formatos. Por outro lado, a teoria da análise musical permite o estudo da mesma usando notações formais, isto é, existe uma definição formal da música[34, 26] reconhecendo-se propriedades algébricas de muitos elementos e operações.

1.2 Objectivos

São definidos quatro tipos de objectivos neste trabalho: teórico, prático, técnico e científico. No âmbito teórico pretende-se definir a música formalmente e investigar as propriedades algébricas [32] das transformações a implementar.

No âmbito do objectivo prático será desenvolvida uma ferramenta que possa ser utilizada pelos alunos da Licenciatura em Música. Deste modo, são imprescindíveis as funcionalidades de importação/exportação de MusicXML, sendo também útil garantir a reprodução/visualização de partituras. Para facilitar o uso da ferramenta será definida uma Linguagem de Domínio Específico[48, 36].

A utilização de ferramentas que agilizem o processo de desenvolvimento são do âmbito do objectivo técnico. Prevê-se o recurso ao *Haskell* [28, 18] e Perl na agilização do processo. Pretende-se que as ferramentas a desenvolver sejam de fácil instalação usando o Cabal. O desenvolvimento será validado com análise de cobertura com recurso ao `hpc`. A documentação será fornecida, sempre que possível, em estilo *literate programming*, combinando-se deste modo o `lhs2TeX` e o `haddock`.

No âmbito do objectivo científico, a notação formal a definir deverá ser comparada com outros formatos existentes, tais como: Lilypond[40, 39], MusicXML e Haskore[26]. Prevê-se a recolha de vasta bibliografia, que suporte o trabalho realizado.

Tem-se por objectivo criar uma biblioteca em que seja a tradução da especificação do formato MusicXML em *Haskell*. Esta biblioteca terá o nome de MusicXML e usa o *namespace* `Text.XML.MusicXML`. Para a descrição da

notação musical e posterior análise, apresenta-se a biblioteca com o nome de *hamusic* que engloba uma descrição da música abstracta, uma estratificação do formato MusicXML e algumas traduções e operações sobre formatos de notação musical.

1.3 Estrutura da dissertação

No próximo capítulo será apresentado uma revisão do estado da arte, abordando as áreas da informática e da música. No terceiro capítulo apresenta-se o pacote musicxml, desenvolvido em *Haskell* que permite a leitura e escrita de ficheiros MusicXML. No quarto capítulo é apresentado o pacote *hamusic*, desenvolvido em *Haskell* juntando a definição de música abstracta com a estratificação do formato MusicXML além de implementações de traduções entre formatos de notação musical. No mesmo capítulo, são apresentadas algumas aplicações que se incluem no pacote *hamusic*, tal como o `MusicCount` e o `MusicTranslate`. No último capítulo é apresentada a conclusão e o trabalho futuro, discriminando cada área de desenvolvimento.

1.4 Sumário de contribuição

Este documento faz um estudo do formato MusicXML, que está a ser utilizado por software de manipulação de notação musical. Numa primeira instância, e com o objectivo de o estudar é apresentada uma biblioteca que permite fazer a leitura e escrita de documentos em MusicXML. A biblioteca tem uma representação interna do MusicXML, isomorfa à especificação em DTDs. Numa segunda instância é apresentada uma biblioteca com o nome de *hamusic* que permite fazer manipulação sobre notação musical no formato MusicXML, usando para o efeito a primeira biblioteca. A biblioteca *hamusic* inclui algumas aplicações como o `MusicCount` e o `MusicTranslate` que efectuam a contagem de entidades musicais e a tradução entre notação musical, respectivamente.

No próximo capítulo apresenta-se uma revisão do estado da arte, onde se fundamenta o trabalho realizado.

Capítulo 2

Revisão do Estado da Arte

Apresenta-se um resumo da pesquisa científica encontrada na investigação realizada, nomeadamente informação sobre os formatos de notação musical com relevância no estudo musical. Também é apresentado o formato *XML* e o ambiente de desenvolvimento em *Haskell*, cobrindo desde compiladores a ferramentas de análise de código.

2.1 Análise Musical

A compreensão de uma peça musical é um componente da análise musical. De seguida são apresentadas tópicos úteis no estudo da estrutura musical.

2.2 Análise Schenkerianana

Na análise Schenkeriana o estudo da estrutura musical é feito de forma inversa. Enquanto que na análise GTTM se inicia com a música e se fazem abstracções de modo a simplificá-la, na análise Schenkeriana[2], o estudo faz-se da abstracção para a representação.

2.3 GTTM

Generative Theory of Tonal Music[33, 31, 32, 19, 20] aborda a estrutura musical e como fazer abstracções de forma metódica. Esta teoria dedica-se ao seu estudo de uma peça musical no sentido de definir regras para extrair a estrutura musical. A estrutura musical advém do estudo da análise métrica e

da análise de grupo. Após estas análise é possível fazer a análise em árvore através da redução de elementos menos destacados.

Na análise segundo a GTTM existem três tipos de regras. As regras de boa-formação indicam as propriedades que se devem preservar, mas será necessário usar as regras de preferência sempre que mais de que uma regra seja aplicável, para se escolher apenas uma regra. As regras de transformação surgem quando a análise não está completa e não é possível aplicar as regras existentes.

Análise de grupo Na análise de grupo existem regras, tais como: Um grupo pode conter grupo mais pequenos, Se um grupo G1 contêm elementos do grupo G2 então deve conter a totalidade do grupo G2. Estas regras revelam propriedades na formação de grupos de tal modo que os grupos devem estar definidos de forma contígua e não existem inclusões parciais de grupos. Existem outras regras que identificam vários elementos contíguos pertencentes a um grupo se tem propriedades semelhantes.

Análise métrica Na análise métrica existem regras que atribuem um valor a cada nota conforme a sua posição no compasso e a sua relação com as outras. Este valor atribuído revela a importância da nota no conjunto da peça musical.

Existem duas teorias de redução através da estabilidade rítmica com a análise rítmica e a análise de grupo. A redução *time-span* é uma redução simples, com o uso dos resultados das análises métrica e de grupo. A redução prolongada permite fazer a análise de música, podendo ser incongruente permitindo a análise fora do âmbito do própria peça musical.

Análise de redução *time-span* O *time-span* é o intervalo de tempo de uma estrutura métrica. Um grupo, identificado pela análise de grupo, também é considerado *time-span*. Através de regras aplicadas após a análise métrica e a análise de grupo é possível fazer a redução *time-span*.

Análise de redução prolongacional A redução prolongacional corresponde à análise Schenkeriana de Ursatz[2, 1], tendo uma representação semelhante à análise descrita anteriormente.

2.4 Ferramentas na análise musical

Existem ferramentas informáticas[13, 11, 12, 14, 15] capazes de fazer automaticamente análise musical, recorrendo à teoria GTTM. Para auxiliar as ferramentas foram desenvolvidos formatos para representar a análise de grupo com o formato GroupingXML, a análise métrica com o formato MetricalXML e a redução *time-span* com o formato Time-spanXML. Estes formato estão ligados ao formato MusicXML através da tecnologia XLink.

Em seguida é apresentada uma imagem correspondente à análise feita pelas ferramentas. Nesta imagem observam-se 3 tipos de análise musical.

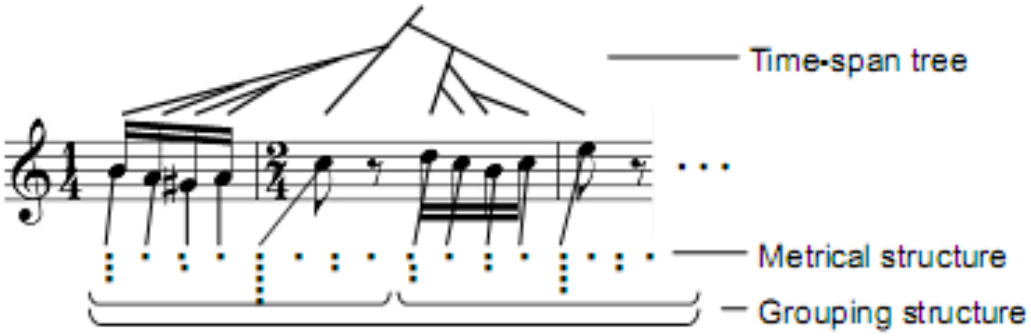


Figura 2.1: Generative Theory of Tonal Music retirada de [13]

2.5 Formatos de Música

Nesta secção é apresentada a forma como são representados diversos formatos de multimédia entre os quais a música.

2.5.1 Música

Um dos formatos de música com grande popularidade é o MP3. No entanto, o formato MP3[1] contém apenas informação áudio. Tal como o MP3, muitos outros formatos que referem a representar música, apenas contém informação relativa ao áudio. Formatos para notação musical abordam a música não só como um documento áudio, mas também como um documento gráfico.

Ferramentas como o Sibelius[44] ou o Finale[37] permitem fazer uma análise musical segundo o áudio ou segundo o aspecto gráfico. Da mesma forma, ainda que menos expressivos, ferramentas sobre os formatos ABC e Lilypond fazem a mesma análise. O formato MusicXML é *XML* armazenando de forma

[1] O formato MP3 é uma forma abreviada para MPEG-1 audio layer 3

expressiva a notação musical. Este formato torna possível a dualidade no tratamento da música.

2.5.2 MusicXML

O MusicXML é um formato em XML que representa a notação musical. Neste formato é possível interpretar a música sob a perspectiva sonora ou sob a perspectiva gráfica. Também é possível descrever os elementos musicais orientados à parte[2] ou orientados ao compasso. A primeira forma tem o nome de partwise e a segunda de timewise. Também é possível, agrupar vários documentos, no contexto de agrupar vários andamentos, sendo o documento agrupador um documento com referências para outros documentos em MusicXML. Uma definição simples do MusicXML é apresentada na seguinte listagem.

Listagem 2.1: Gramática simples do MusicXML

```
1 MusicXML ::= Partwise
2              | Timewise
3              | Opus
4 Opus ::= Partwise*
5        | Timewise*
```

Na distribuição de MusicXML na forma comprimida, é necessária uma nova representação com base de referências, funcionando como um *MANIFEST*. Para a compressão é usada a compressão zip, onde existirá um ficheiro container.xml na pasta META-INF no ficheiro comprimido.

De seguida, é apresentada uma breve descrição para cada documento da especificação DTD(Document Type Definition) do MusicXML. Estes documentos estão disponíveis em [6], onde está presente a especificação do MusicXML usando XML Schemas.

O documento common Este ficheiro contêm as definições elementares da definição da estrutura do MusicXML. Muitas destas definições elementares são entidades paramétricas, quando especificadas em DTDs, de forma a reutilizar componentes.

O ficheiro attributes Este ficheiro define o elemento `attribute` e seus descendentes. O elemento `attribute` é a entidade *XML* que engloba a definição de modo, compasso, clave, número de instrumentos entre outras opções. A indicação da duração de cada nota é relativa ao número de divisões permitida pela entidade `divisions` contida na entidade `attribute`. Cada unidade desta entidade refere-se à duração de uma semínima.

[2] Do inglês *part* que representa a noção de instrumentos/vozes.

O documento link Este ficheiro define um extracto da tecnologia do XLink, definindo os elementos link e barline que usam a referida tecnologia.

São definidas as entidades link e `bookmark` que utilizam a entidade paramétrica `%link-attributes`.

O documento identity Este ficheiro define os meta dados existentes num ficheiro MusicXML, tal como a informação do autor do documento, o software usado, a data em que foi gravado, entre outras. O formato da data em que o documento é criado/modificado segue as normas especificadas no ISO 8601.

O documento barline Este ficheiro define informação sobre as barras de divisão de compasso.

O documento note Este ficheiro define o elemento note e seus descendentes. As notas são definidas por este elemento. Pelo que a música não é descrita de forma hierárquica, encontra-se na definição da nota entidades tais como `staff` ou `instrument` que identificam a nota hierarquicamente. Desta forma, existe expressividade na definição da notação musical quanto à estrutura musical.

O documento score Este ficheiro define uma peça musical. A especificação de uma partitura musical pode ser de duas formas. A forma mais comum é orientada à partitura, e a segunda é a orientada ao compasso. Este documento contém as duas especificações que são invocadas de forma exclusiva.

O documento partwise Na seguinte listagem apresenta-se uma definição simples do formato MusicXML orientado à parte. Esta definição agrupa as várias partes, sendo a mais usual forma de distribuição de música em MusicXML.

Listagem 2.2: MusicXML Partwise

```
1 Partwise ::= part+
2 part ::= measure+
3 measure ::= %music-data+
```

Este documento faz uso de secções condicionais. Opta pela definição `%partwise` incluindo os vários ficheiros que contém a especificação. Esta notação divide uma música em várias partes e cada uma das partes é dividida em vários compassos que contém vários elementos musicais, respectivamente. Esta especificação está disponível em `http://www.recordare.`

com/dtds/partwise.dtd, recomendando-se o uso no cabeçalho do docucumento:

Listagem 2.3: Declaração partwise

```
1  <!DOCTYPE score-partwise PUBLIC
2          "-//Recordare//DTD MusicXML 2.0 Partwise//EN"
3          "http://www.musicxml.org/dtds/partwise.dtd">
```

O documento timewise Esta definição de Timewise, apresentada na seguinte listagem, agrupa os vários elementos musicais em fracções temporais, tendo um compasso os elementos musicais de várias partes. A definição orientada ao tempo é a transposição da definição orientada à parte.

Listagem 2.4: MusicXML Timewise

```
1 Timewise ::= measure+
2 measure ::= part+
3 part ::= %music-data+
```

Este documento faz uso de secções condicionais. Opta pela definição %timewise incluindo os vários ficheiros que contém a especificação. Esta notação divide uma música em vários compassos e cada compasso é dividido várias partes que contém vários elementos musicais, respectivamente. Esta especificação está disponível em http://www.recordare.com/dtds/timewise.dtd, recomendando-se o uso no cabeçalho do documento:

Listagem 2.5: Declaração timewise

```
1  <!DOCTYPE score-timewise PUBLIC
2          "-//Recordare//DTD MusicXML 2.0 Timewise//EN"
3          "http://www.musicxml.org/dtds/timewise.dtd">
```

O documento opus Este documento corresponde à necessidade de agrupar vários documentos no formato MusicXML, de tal forma que é criada uma entidade para o efeito. A entidade tem o nome de opus e define-se em árvore em que as folhas são links para documentos MusicXML, através do uso do XLink. Esta notação agrupa as várias obras num único documento através de XLink. Desta forma é possível, por exemplo, ter vários andamentos numa única especificação. Esta especificação está disponível em http://www.recordare.com/dtds/opus.dtd, recomendando-se o uso no cabeçalho do docucumento:

Listagem 2.6: Declaração opus

```
1  <!DOCTYPE opus PUBLIC
2          "-//Recordare//DTD MusicXML 2.0 Opus//EN"
3          "http://www.musicxml.org/dtds/opus.dtd">
```

O ficheiro container Este documento corresponde à necessidade de reduzir o tamanho dos documentos MusicXML, pelo que se inclui num arquivo indicando o caminho dos outros documentos MusicXML. A compressão utilizada é o algoritmo `DEFLATE` especificado em `http://www.ietf.org/rfc/rfc1951.txt`. Quando o MusicXML usa a forma comprimida, recomenda-se a utilização da extensão ".mxl"e o `mime-type` `application/vnd.recordare.musicxml`. Também se recomenda que na forma comprimida se use meta dados indicando os ficheiros presentes, para isso usando um ficheiro *XML* que seja válido conforme `containers.dtd` Também se sugere que se use no cabeçalho do referido ficheiro *XML* a seguinte declaração:

Listagem 2.7: Declaração container

```
1 <!DOCTYPE container PUBLIC
2                  "-//Recordare//DTD MusicXML 2.0 Container//EN"
3                  "http://www.musicxml.org/dtds/container.dtd">
```

A versão do MusicXML actual é 2.0, havendo compatibilidade entre as versões anteriores. Além de existir total compatibilidade entre as versões anteriores, existem também disponíveis transformações XSL que traduzem o MusicXML da versão 2.0 na 1.1 e da versão 1.1 para 1.0.

A manipulação de MusicXML pode ser efectuada através de formas distintas. Uma das formas é a criação de uma biblioteca que permita a sua manipulação, tal como existe uma biblioteca em C++. Esta biblioteca[30] disponibiliza operações para manipular o MusicXML da versão 0.8 além de pequenas aplicações. Também é com recurso a aplicações que existem aplicações que exportam para MusicXML, tal como o xml2ly. De um modo semelhante, se podem aplicar transformações ao MusicXML para se obter o resultado pretendido, como em abc.xls se pretende transformar um documento MusicXML na versão 0.6 em ABC. O trabalho, neste documento apresentado fornece a possibilidade de importar e exportar documentos MusicXML para um tipo de dados em Haskell.

Hello World em MusicXML De seguida é apresentado um pequeno exemplo de MusicXML. Este pequeno exemplo apresenta a codificação de uma nota em MusicXML. A nota codificada é a semibreve dó, na clave de sol com compasso $\frac{4}{4}$, na tonalidade de dó maior.

Listagem 2.8: Hello World em MusicXML

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE score-partwise PUBLIC
3     "-//Recordare//DTD MusicXML 2.0 Partwise//EN"
4     "http://www.musicxml.org/dtds/partwise.dtd">
5 <score-partwise version="2.0">
6     <identification>
```

```
7           <encoding>
8               <software>ProxyMusic 2.0 c</software>
9           </encoding>
10      </identification>
11      <part-list>
12          <score-part id="P1">
13              <part-name>Music</part-name>
14          </score-part>
15      </part-list>
16      <part id="P1">
17          <measure number="1">
18              <attributes>
19                  <divisions>1</divisions>
20                  <key>
21                      <fifths>0</fifths>
22                  </key>
23                  <time>
24                      <beats>4</beats>
25                      <beat-type>4</beat-type>
26                  </time>
27                  <clef>
28                      <sign>G</sign>
29                      <line>2</line>
30                  </clef>
31              </attributes>
32              <note>
33                  <pitch>
34                      <step>C</step>
35                      <octave>4</octave>
36                  </pitch>
37                  <duration>4</duration>
38                  <type>whole</type>
39              </note>
40          </measure>
41      </part>
42  </score-partwise>
```

2.5.3 ABC

O formato ABC[8, 4] é um formato simples para representar notação musical simples. Existe um conjunto de ferramentas que permite manipular este formato de texto, com a ajuda de um sistema WYSIWYG.

De seguida apresenta-se um exemplo de notação musical em ABC. O exemplo é equivalente.

Listagem 2.9: Hello World em ABC

```
1  X:1
2  T:Title
3  M:4/4
4  L:1/1
5  K:C
6  C
```

Na figura 2.2 apresenta-se graficamente o resultado obtido com o abcexplorer, equivalente às listagens de Hello World em alguns formatos musicais,

tais como MusicXML, ABC e Lilypond.

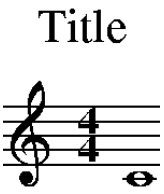


Figura 2.2: Hello World

2.5.4 Lilypond

Lilypond[46, 39, 40] é uma ferramenta de código aberto e uma linguagem para representar notação musical.

De seguida é apresentado um exemplo do Lilypond.

Listagem 2.10: Hello World em Lilypond

```
1 \header{
2   title = "Title"
3 }
4 \relative{
5   c1
6 }
7 \version "2.10.0"  % necessary for upgrading to future LilyPond versions.
```

Esta notação musical além de ser bastante expressiva, tem ferramentas que fazem pontes com outras notações musicais. Por exemplo, xistem ferramentas de importação, com o recurso a linguagens de `scripting`, que convertem documentos com notação musical em ABC, ETF, MIDI e até MusicXML em Lilypond, com o nome de `abc2ly.py`, `etf2ly.py`, `midi2ly.py` e `musicxml2ly.py`, respectivamente. Também exporta para alguns formatos entre os quais o PDF, PNG e MIDI.

O projecto mutopia[41] é um repositório de documentos lilypond e ficheiros gerados para reprodução e para visualização. Este repositório disponibiliza os referidos documentos para download.

2.5.5 Guido

Guido[42, 23, 22] é um formato de notação musical textual que surgiu no início da década de 90, quando os primeiros protótipos do sistema Salieri e da respectiva linguagem foram desenhados e implementados. Posteriormente tornou-se parte integrante do projecto SALIERI[24].

2.5.6 MIDI

O MIDI é um formato de música que contém instruções de processamento de sons. É um protocolo desenvolvido com o objectivo de haver comunicação entre os vários dispositivos electrónicos que manipulavam música, entre os quais sintetizadores. O formato é baseado em mensagens de eventos sonoros como à alternativa à emissão de sinais de áudio ou multimédia. O MIDI pode ser definido como uma sequência de eventos musicais, onde cada evento é caracterizado pelo menos pela frequência, duração e instrumento. Este formato foi utilizando por muito tempo como um formato de intercâmbio entre vários formatos de notação musical. No entanto a sua falta de estruturação musical não permite análise musical.

2.5.7 Haskore

O Haskore[26, 25] é uma biblioteca em *Haskell* que permite a criação, análise e manipulação de música. A reprodução de música é efectuada através de programas externos. Recentemente, o Haskore está a ter uma reestruturação em relação a dependências, de forma a separar o pacote Haskore em pacotes de software mais pequenos para aumentar também a portabilidade. O Haskore está disponível em `http://darcs.haskell.org/haskore/` e agora também no hackage[10].

O Haskore usa uma arquitectura onde se distinguem noções de duração e de frequência. No definição de frequências é mais fácil a utilização absoluta em detrimento das frequências relativas.

O Haskore, além de operações algébricas sobre música, contém interfaces com autotrack, CSound[49] e MIDI. O primeiro é uma aplicação distribuída pela mesma biblioteca, sendo os restantes formatos de áudio.

2.5.8 Outras ferramentas de notação musical

Os dois programas para notação musical mais utilizados são o Sibelius[44] e o Finale[37]. Estas aplicações permitem a edição de música e a sua manipulação. Para a sua manipulação existem extensões e ambientes programáveis nas aplicações.

Formas fáceis de visualizar música com extensões são disponibilizadas para *browsers*, tal como Myriad Music Plugin[38] ou o Scorch Plugin[43]. Estas extensões nos *browsers* permitem ter objectos de música numa página de internet, facilitando a interacção com a notação musical.

2.5.9 Traduções entre formatos Musicais

É apresentada uma tabela onde-se vários formatos previamente apresentados tem descritas ferramentas de tradução. A existência de tradução entre vários formatos de notação musical é útil.

Formato	Ferramentas	Importação	Exportação
MusicXML	Dolet	Sibelius files; Finale files	Sibelius files; Finale files
Lilypond	lilypond	abc2ly; etf2ly; musicxml2ly	Sim
ABC	abcExplorer	Sim	Sim
Haskore	Haskore	Não	Midi; SynCollider

Tabela 2.1: Traduções entre formatos de notação de musical

2.6 XML

O *XML*[52] foi desenvolvido pelo W3C[51] com o objectivo de ser simples e de aproveitar recursos e tecnologia existentes do mundo do SGML. O *XML* também tem associada a *boa-formação* e a validação. Um documento *XML bem-formado* indica que a anotação não entra em conflitos/ambiguidades, isto é, existe apenas um nodo na raiz do documento, o nome da *tag* de abertura é igual ao nome da *tag* de fecho. A validação de um documento indica que um documento *XML* está conforme a especificação indicada. Esta indicação da especificação pode ser dada com o recurso a DTDs e a Schemas além de outras formas.

Um DTD(Document Type Definition) é uma especificação que usa a gramática BNF para especificar a estrutura de um ficheiro *XML*. No entanto, actualmente os *XML* Schemas[53] permitem uma maior expressividade na especificação da estrutura do que a tradicional especificação com recurso ao DTD, já que indicam mais detalhe em relação aos tipos e expressões.

2.7 Haskell

A linguagem de programação *Haskell* [18] surgiu no final da década de 80, com os objectivos de ser de fácil utilização académica, em investigação, em grandes projectos, de ter uma definição formal e reduzir a diversidade das linguagens de programação funcionais.

A arquitectura da linguagem *Haskell* permite-nos dizer que o *Haskell* tem o sistema de avaliação de expressões *lazy*, o que quer dizer que as expressões apenas são avaliadas quando são utilizadas por outras expressões. Este tipo de avaliação de expressões é antagónico ao utilizado por muitas linguagens de programação imperativas, que usam a avaliação estrita, o que quer dizer a avaliação de expressões é feita tal como se escreve. Outra característica do *Haskell*, é que é uma linguagem funcional pura, sendo uma consequência imediata da avaliação *lazy*, dado que não tem efeitos laterais. A pureza de efeitos laterais ajuda a definir funções com propriedades matemáticas de forma literal. No entanto, quando são necessários efeitos laterais, a linguagem tem disponíveis os efeitos monádicos, sendo a monad IO a mais conhecida.

Em 1998 surgiu o *Haskell*98[28] padronizando informalmente o *Haskell*. Desta forma o *Haskell* tem vindo a ter alterações e extensões, tendo como base o *Haskell*98. Actualmente existem pelo menos dois tutoriais de *Haskell*[28, 27].

2.7.1 O compilador - GHC

O GHC[16] é o compilador de *Haskell* com mais funcionalidades, desenvolvido segundo um projecto de código aberto. O GHC foi iniciado após a definição inicial do *Haskell* ter estabilizado. O GHC é um compilador de *Haskell* desenvolvido em *Haskell*.

Em quinze anos de existência o GHC suporta um grande número de funcionalidades, entre as quais: extensões à linguagem e um interpretador interactivo com o nome de GHCi.

Também paralelamente foi desenvolvido o Gofer(GOod For Equational Reasoning), um interpretador desenvolvido em C. Posteriormente a adaptação do Gofer ao *Haskell* tem o nome de Hugs[17](Haskell User's Gofer System).

2.7.2 O analisador de código - HPC

O HPC[3] é uma ferramenta que permite fazer a análise da execução do código. Esta ferramenta cria relatórios em formato de texto e em HTML. As páginas HTML criadas contém o código analisado, com destacando as expressões executadas e as não-executadas, indicando o número de execuções quando aplicável.

Só é possível fazer a análise se o executável gerar um ficheiro com a extensão TIX. Este ficheiro em conjunto com os ficheiros com extensão MIX, gerados para cada módulo, permitem fazer a análise de código. Quando se invoca o GHC para fazer a compilação, adiciona-se a *flag* `fhpc`.

2.7.3 A análise da memória - HP2PS

A aplicação `hp2ps` converte um perfil de utilização da *heap* num gráfico Postscript. O perfil de utilização da *heap* é obtido de um ficheiro com extensão `.hp`, sendo o resultado da execução de uma aplicação compilada pelo GHC com a *flag* `+RTS -hT`. É um utilitário que permite fazer uma análise da *heap* utilizada. O gráfico resultante da execução apresenta o tempo de execução e o tamanho da memória utilizada pelos vários tipos.

2.7.4 Documentação - Haddock

O haddock é uma ferramenta para gerar documentação da API de programas em Haskell, sendo semelhante ao `javadoc` ao recorrer à assinatura de funções e comentários para, com uma notação simples constrói páginas HTML que documentam cada função/tipo de dados existente. Esta documentação é muito útil, pois permite navegar nas funções e tipos de dados, usando *links*. Normalmente esta ferramenta não é utilizada em ambientes `literate programming`.

2.7.5 Literate Programming - lhs2TeX

Esta aplicação pretende utilizar o mesmo documento tanto para a aplicação como para documentação. O `lhs2TeX`[35] é uma ferramenta apenas usada em Literate *Haskell* para formatar o código *Haskell* presente num documento LaTeX, pois traduz o ficheiro de Literate *Haskell* num documento LaTeX, preservando o texto LaTeXe reescreve o código *Haskell*. Esta ferramenta interpreta alguns comandos especiais que estão incluídos no documento LaTeX.

2.7.6 Gestor de pacotes - Cabal

Cabal é o acrónimo para "`Common Architecture for Building Applications and Libraries`", sendo um dos seus objectivos simplificar a forma da distribuição de software desenvolvido em *Haskell*. A unidade de distribuição para o Cabal é o pacote. Um pacote pode englobar uma biblioteca, disponibilizando vários módulos *Haskell* e/ou um ou mais programas desenvolvidos em *Haskell*.

Neste sistema declara-se o pacote de software num ficheiro .cabal que será processado pelo cabal, usando um módulo *Haskell* que importe definições do cabal. Este módulo *Haskell* geralmente tem o nome `Setup.hs` ou `Setup.lhs`. É usual fazer a seguinte invocação: `runhaskell Setup.hs`

<comando>, dado que `runhaskell` vai invocar um interpretador de *Haskell*, seja ele o GHCi ou o Hugs ou outro e executar a função `main`.

Também existe um pacote de software com o nome `cabal-install` que permite a instalação recursiva de vários pacotes, sendo efectuado o download sempre que necessário. Este pacote de software é muito semelhante ao `cpan`, que facilita a instalação de pacotes de `perl`, usando o repositório `hackage`. O repositório `hackage` pode ser utilizado de forma manual, ou de forma automática, com a referida aplicação.

O pacote `cabal-install` contem um executável de nome cabal. P a instalação de um pacote executa-se o comando `cabal install pacote`. A aplicação verifica a existência do pacote localmente e num repositório de modo a efectuar o download se necessário. Para actualizar a lista de pacotes disponívels é útil a execução de `cabal update`.

2.7.7 Biblioteca - HaXml

Existem algumas bibliotecas desenvolvidas em *Haskell* que manipulam *XML*, entre as quais destacamos o HaXml, o HXT, libxml.

Uma das bibliotecas de manipulação de ficheiros *XML* de mais fácil utilização e funcional é a HaXml. Esta biblioteca tem ferramentas úteis para a manipulação de *XML*, entre as quais um conversor de ficheiros no formato DTD para código *Haskell* que permite a sua manipulação.

A biblioteca HaXml[47, 21, 54, 55] é um conjunto de módulos *Haskell* que permitem ler e escrever *XML*, além de existirem facilidades na transformação de *XML*. Esta biblioteca inclui:

- Um parser para *XML*
- Definição exaustiva do *XML* em Tipos de Dados
- Facilidade na transformação da árvore *XML* no tipo de dados *Haskell* equivalente.
- Transformação de um DTD em *Haskell*.

Esta biblioteca tem uma interface minimalista que aumenta a sua usabilidade, não comprometendo a robustez.

Esta biblioteca permite além do parser *XML*, a transformação do DTD em *Haskell*, facilitando a utilização. Apesar de a primeira componente funcionar sem problemas aparentes, a segunda não consegue processar toda a expressividade dos DTDs. Após várias manipulações nos DTDs ainda revelava alguns problemas no processamento do *XML* correspondente.

DtdToHaskell É uma ferramenta disponibilizada pela biblioteca HaXml que permite transformar um DTD num módulo *Haskell*. No entanto, levanta ainda alguns problemas na leitura de DTDs complexos. Na definição de elementos, quando se usam elementos aninhados ou elementos mistos não consegue processar. No primeiro caso por falha na Validação, apesar se o DTD ser válido, no segundo caso por entrar num *ciclo infinito*.

Validate É uma ferramenta disponibilizada pela biblioteca HaXML que permite a Validação de um ficheiro *XML* dado um ficheiro DTD. Apresenta os mesmos problemas na validação que a ferramenta descrita anteriormente.

2.8 Sumário

Apresenta-se uma lista de vários formatos para notação musical. Destacam-se os formatos MusicXML, ABC e Lilypond. O primeiro por ser objecto de estudo na dissertação, além de ser utilizado por ferramentas profissionais de notação musical. Os outros formatos são abordados por serem usados por ferramentas sem custos e as ferramentas no processamento de notação musical apresentarem resultados satisfatórios.

De seguida é apresentada o Haskell como sendo a linguagem de programação para a implementação e ferramentas utilizadas no desenvolvimento, tais como o GHC para compilar, o haddock para gerar documentação técnica, o lhs2tex para ser possível integrar a documentação com a implementação, o hp2ps para analisar a memória utilizada, o hpc para analisar a cobertura de código, além de bibliotecas como o HaXml, que facilita a manipulação de documentos em XML.

O MusicXML pela sua aceitação e utilização pelas principais ferramentas de notação musical é abordada uma implementação em Haskell. Esta biblioteca é a base do trabalho desenvolvido.

Capítulo 3

musicxml: Implementação em *Haskell*

O presente capítulo apresenta a implementação do formato *MusicXML* em *Haskell* e de funções que permitem importar/exportar documentos em MusicXML para a implementação em *Haskell*. Pretendeu-se com esta implementação não efectuar qualquer interpretação da especificação do MusicXML, mas apenas a sua tradução para *Haskell*, com certas especificações na tradução.

3.1 Introdução

Este documento contêm código *Haskell* formatado com a ferramenta `lhs2TeX`.

A definição de tipos está conforme a especificação feita pela Recordare, apresentada na segunda versão do DTD do MusicXML.

No momento de escrita desta biblioteca a Recordare estava a publicar especificações de MusicXML usando *XML* Schemas, enquanto estava sendo discutida pela comunidade.

A biblioteca MusicXML usa a biblioteca HaXml para manipular os ficheiros *XML*, isto é, para efectuar a leitura e escrita.

A arquitectura da biblioteca é apresentada na figura 3.1.

A definição de tipos não usa `datatype`, usando `type`. Esta opção diminui a percepção de código, melhorando a sua performance. Para melhorar a manutenção foi construído o módulo `Util`, o qual apresenta funções elementares de leitura e escrita. Para minimizar o código, na leitura do formato MusicXML foi usado uma monad de Estado, definida no módulo `Util` e denominada de `STM`. No entanto para a escrita, são apenas usadas funções puras, isto é, sem recurso a *monads*.

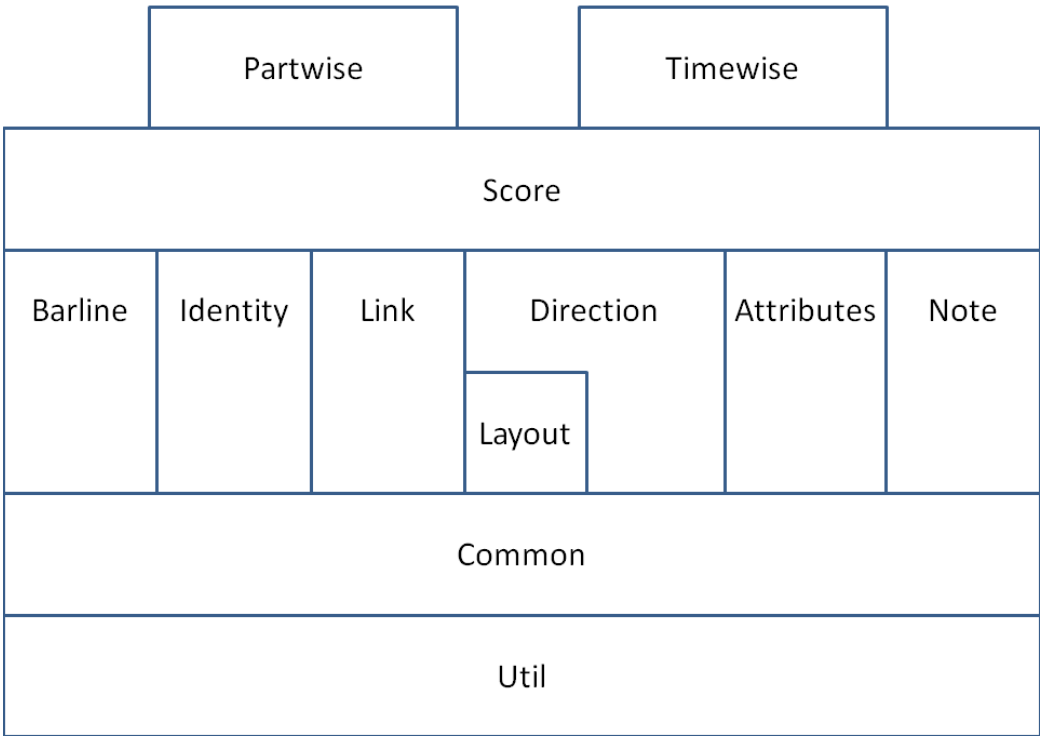


Figura 3.1: Estrutura do pacote MusicXML

Esta abordagem ajuda a manutenção, de tal modo que a implementação em *Haskell* é similar à sua especificação em DTDs.

3.2 Problemas de Implementação

Vislumbrou-se a possibilidade de utilizar a ferramenta DtdToHaskell para converter a especificação em código *Haskell*. No entanto, a ferramenta não suporta de forma funcional entidades paramétricas, largamente usadas pela especificação, de modo que não conseguia ler a especificação. A ferramenta detectava um falso erro de sintaxe nos ficheiros DTDs.

Foi conseguida a substituição de entidades paramétricas, perdendo-se a estrutura de vários ficheiros num único ficheiro. Este único ficheiro sem entidades paramétricas após pequenas alterações devido a erros de sintaxe, devido ao não reconhecimento de XLink. O resultado é um ficheiro DTD, com toda a especificação equivalente aos anteriores. No entanto este ficheiro, em conjunto com a documentação existente tem cerca de 70KB. A ferramenta DtdToHaskell consegue transformar este documento num ficheiro com código *Haskell* que fica com 700KB e com quase 15000 linhas.

Quando é feita a compilação deste ficheiro de 700KB de código, usando o GHC, apesar de necessitar de muita memória também necessita de algum tempo, dado que o ficheiro objecto tem mais de 10MB.

Havendo duas especificações em DTDs, a definição em `partwise.dtd` e em `timewise.dtd`, sendo necessária a execução do processo anterior por duas vezes, uma para cada ficheiro.

Deste modo, a solução não é viável devido ao enorme pedido de recursos além da divergência entre a especificação e a implementação, sendo no mínimo com perdas da estrutura. Além de que na execução de alguns casos, verificou-se que o código gerado não terminava. Para provar este facto, criou-se um DTD, com um elemento misto.

Listagem 3.1: DTD com elemento misto

```
1 <!-- DTD score ... -->
2 <!-- (C) ... Wed Jun 25 18:13:33 2008 -->
3 <!ELEMENT score (#PCDATA|a|b)* >
4 <!-- a=1 | b=1 -->
5 <!ELEMENT a (#PCDATA) >
6 <!-- #PCDATA=1 -->
7 <!ELEMENT b (#PCDATA) >
8 <!-- #PCDATA=1 -->
```

Estava previsto o uso da ferramenta `DTD2Haskell` para a conversão automática da especificação DTD para código *Haskell* com a capacidade de ler e escrever ficheiros *XML* segundo a especificação indicada.

3.3 Especificações da Implementação

A especificação em *Haskell* abrange a especificação de *Container*, *Opus* além dos ficheiros que contém notação musical, tais como: *Partwise* e *Timewise*.

Foi efectuada uma tradução manual da especificação contida nos DTDs para a especificação presente em *Haskell*. Foi preservada a estrutura da especificação, mantendo-se ficheiros com nomes equivalentes.

O ficheiro `Note.lhs` refere-se à especificação do ficheiro `note.mod`, assim como os ficheiros `Attributes.lhs`, `Barline.lhs`, `Common.lhs`, `Direction.lhs`, `Identity.lhs`, `Layout.lhs` e `Link.lhs` se referem à especificação presente em `attributes.mod`, `barline.mod`, `common.mod`, `direction.mod`, `identity.mod`, `layout.mod` e `link.mod`, respectivamente. Existe uma excepção para os ficheiros `score.mod`, `partwise.dtd` e `timewise.dtd` devido à utilização da inclusão condicional em DTD que não é possível fazer o equivalente usando *Haskell*. Deste modo, manteve-se a orientação na arquitectura até onde foi possível, separando a especificação que entrava em conflito no ficheiros `Partwise.lhs` e `Timewise.lhs`, respectivamente, mantendo a mesma semântica, que será apresentado em maior detalhe. A solução para o conflito das especificações causadas pela inclusão condicional disponibilizada pelos DTDs foi resolvida movendo apenas a especificação em conflito em *Haskell* para os ficheiros `Partwise.lhs` e `Timewise.lhs`. Deste modo, a especificação de `score.mod` foi traduzida em `Score.lhs` exceptuando este caso. Desta forma, manteve-se a semântica da especificação e também a quase totalidade da sua arquitectura.

Também se manteve na especificação em *Haskell* a documentação apresenta nos ficheiros de especificação em DTD. Foi utilizado o seguinte *namespace* `Text.XML.MusicXML` para a especificação em *Haskell* por ser uma especificação sobre *XML* que tem o *namespace* em `Text.XML`.

A especificação em *Haskell* usa ainda um ficheiro `Util.lhs` com funções utilitárias que usam a biblioteca HaXml e outro ficheiro `MusicXML.lhs` que agrupa todos os ficheiros.

Este pacote de software disponibiliza dois tipos de funções. As funções que transformam o conteúdo da árvore DOM[50] na sua semântica, sendo as funções denominadas de leitura, tendo como prefixo `read_`. O outro grupo de funções que transforma a semântica na árvore DOM são denominadas de funções de escrita e tem como prefixo show_. Esta nomenclatura é inspirada no *Haskell* que chama à função de leitura `read` e às funções de escrita de show. Para facilitar a alteração de metadados, tal como o nome do software e a data em que o documento é criado/modificado, foram criadas funções para tal efeito.

No tratamento da árvore DOM, revelou ser necessário o uso de Monadas de estado quando é feita a importação da DOM. A necessidade do estado prende-se com a necessidade de se pretender que a identificação baseada nos DTDs dos documentos *XML* seja correcta. A implementação funcional pura seria muito extensa, dado que seria também necessário implementar não só os casos de sucesso mas também os casos de insucesso, além de preservar a semântica.

3.3.1 Implementação de funções utilitárias

Este módulo responde à necessidade de existir funções para tipos base, tais como: `PCDATA`. Este módulo contém funções que efectuam a leitura/escrita da árvore *XML* para um tipo específico. Contém funções com a semântica dos operadores usados no DTD ?, + e $\times$. Estas funções tem o nome de com o sufixo `MAYBE`, `LIST` e `LIST1`.

A função sobre `ELEMENT` corresponde à manipulação de elementos *XML*.

Este módulo inicia-se por definir tipos de dados e funções sobre monads, para facilitar a escrita de todo o software, diminuindo significativamente a complexidade do código. As *monads* implementadas, *monads* de estado e *monads* Result, que apresenta uma mensagem de erro caso a computação falhe.

Listagem 3.2: Módulo Util - tipos de dados

```
1  -- * Result
2  -- |
3  data Result a = Ok a | Error String
4      deriving (Eq, Show)
5  -- |
6  instance Monad Result where
7      (Ok a)      >>= b = b a
8      (Error msg) >>= _ = Error msg
9      return x = Ok x
10     fail msg = Error msg
11 -- |
12 instance Functor Result where
13     fmap f (Ok a)      = Ok (f a)
14     fmap _ (Error msg) = Error msg
15 -- |
16 instance MonadPlus Result where
17     mzero = Error "unknow error"
18     (Ok a) `mplus` _ = (Ok a)
19     (Error _) `mplus` b = b
20
21 -- |
22 isOK :: Result a -> Bool
23 isOK (Ok _) = True
24 isOK _      = False
25 -- |
26 isError :: Result a -> Bool
27 isError (Error _) = True
```

```
28 isError _          = False
29 -- |
30 fromOK :: Result a -> a
31 fromOK (Ok a)       = a
32 fromOK (Error msg) = throw (ErrorCall msg)
33 -- |
34 fromError :: Result a -> String
35 fromError (Ok _)      = []
36 fromError (Error msg) = msg
37
38 -- * ST
39 -- |
40 newtype ST s a = ST {state :: s -> (s,a)}
41 instance Monad (ST s) where
42     return x = ST (\s -> (s,x))
43     p >>= f  = ST (\s1 -> let (s2, r) = state p s1 in state (f r) s2)
44 instance Functor (ST s) where
45     fmap f st = ST (\s -> (\(x,y) -> (x, f y)) (state st s) )
46 -- |
47 liftST :: (s -> a) -> ST s a
48 liftST f = ST (\s -> (s,f s))
49
50 -- * STM
51 -- |
52 newtype STM m s a = STM {stateM :: s -> m (s, a)}
53 -- |
54 instance (Monad m) => Monad (STM m s) where
55     return x = STM (\s -> return (s,x))
56     p >>= f  = STM (\s -> do {
57         ; (s',l) <- stateM p s
58         ; stateM (f l) s'})
59     fail msg = STM (\_ -> fail msg)
60 -- |
61 instance MonadPlus m => MonadPlus (STM m s) where
62     mzero = STM (\_ -> mzero)
63     a `mplus` b = STM (\s -> (stateM a s) `mplus` (stateM b s))
64 -- |
65 instance Monad m => Functor (STM m s) where
66     fmap f stm = STM (\s -> stateM stm s >>= (\(s1, a) -> return (s1, f a)))
67 -- |
68 liftSTM :: Monad m => ST s (m a) -> STM m s a
69 liftSTM p = STM (\s -> do {
70     ; let (s', l) = (state p s)
71     ; lx <- l
72     ; return (s',lx)})
73 -- |
74 returnSTM :: Monad m => m a -> STM m s a
75 returnSTM x = STM (\s -> x>>=(\y -> return (s,y)))
```

De seguida, apresenta funções para manipulação de estruturas simples enunciadas no DTD, tal como os tipos `ID`, `PCDATA`, `CDATA` e construções como `IMPLIED` , `OPTIONAL`, e `REQUIRED`. Havendo no entanto algum relaxamento quanto aos tipos `NMTOKEN` que são tratados como `CDATA`.

Listagem 3.3: Módulo Util - construções simples

```
1 -- * Basic
2 -- |
3 type CDATA = Prelude.String
4 -- |
```

```
5  read_CDATA :: Prelude.String -> Result CDATA
6  read_CDATA = return
7  -- |
8  show_CDATA :: CDATA -> Prelude.String
9  show_CDATA = id
10 -- |
11 type ID = Prelude.String
12 -- |
13 read_ID :: Prelude.String -> Result ID
14 read_ID = return
15 -- |
16 show_ID :: ID -> Prelude.String
17 show_ID = id
18 -- * Attributes
19 -- |
20 read_IMPLIED' :: String -> (String -> Result a) -> [Attribute] -> Maybe a
21 read_IMPLIED' key func s = maybe Nothing
22             (result . func . unwords .
23                 map (either id (const "")) . (\(AttValue l) -> l))
24             (lookup key s)
25     where -- |
26           result :: Result a -> Maybe a
27           result (Ok x) = Just x
28           result (Error _) = Nothing
29 -- |
30 read_IMPLIED :: Monad m =>
31     String -> (String -> Result a) -> STM m [Attribute] (Maybe a)
32 read_IMPLIED key func =
33     STM (\s-> return (s, read_IMPLIED' key func s))
34 -- |
35 show_IMPLIED :: String -> (a -> String) -> Maybe a -> [Attribute]
36 show_IMPLIED key function = maybe [] (show_REQUIRED key function)
37 -- |
38 read_REQUIRED :: Monad m => String -> (String -> Result a) -> STM m [Attribute] a
39 read_REQUIRED key func =
40     read_IMPLIED key func >>=
41     maybe (fail ("I expect "++key++" as required attribute")) return
42 -- |
43 show_REQUIRED :: String -> (a -> String) -> a -> [Attribute]
44 show_REQUIRED key function =
45     (:[]) . (\x -> (key, x)) . AttValue . (:[]) . Left . function
46 -- |
47 read_DEFAULT :: Monad m =>
48     String -> (String -> Result a) -> a -> STM m [Attribute] a
49 read_DEFAULT key func def =
50     read_IMPLIED key func >>=
51     maybe (return def) return
52 -- |
53 show_DEFAULT :: String -> (a -> String) -> a -> [Attribute]
54 show_DEFAULT = show_REQUIRED
55 -- |
56 show_FIXED :: String -> (a -> String) -> a -> [Attribute]
57 show_FIXED = show_REQUIRED
58 -- |
59 read_FIXED :: Monad m =>
60     String -> (String -> Result a) -> a -> STM m [Attribute] a
61 read_FIXED key func def =
62     read_IMPLIED key func >>=
63     maybe (return def) return
```

Também apresenta funções de manipulação de elementos *XML* que tem

como sufixo `ELEMENT`. Estas funções permitem a importação da DOM e a sua exportação a partir de um tipo genérico. Para auxiliar estas funções foram criadas funções para extrair os atributos e os filhos com o nome de `attributes` e `childs`. Para complementar a manipulação de elementos, foi necessária a escrita de funções sobre `PCDATA`, que têm o nome no sufixo.

Listagem 3.4: Módulo Util - contrução ELEMENT

```
1  -- |
2  read_ELEMENT' :: String -> [Content i] -> ([Content i], Result (Element i))
3  read_ELEMENT' tag ((CElem (e@(Elem key _ _)) _):t) | key == tag = (t, Ok e)
4  read_ELEMENT' tag ((CString _ s _):t) | Prelude.and (map isSpace s) =
5      read_ELEMENT' tag t
6  read_ELEMENT' tag (((CMisc _ _):t)) = read_ELEMENT' tag t
7  read_ELEMENT' tag l                 =
8      (l, Error ("l␣expect␣" ++ tag ++ "␣element" ++ moreinfo))
9      where moreinfo :: String
10           moreinfo = ":␣[" ++ concat (map conts l) ++ "]"
11           -- |
12           conts :: Content i -> String
13           conts (CElem (Elem k _ _) _) = "<" ++ k ++ "/>"
14           conts (CString _ s _) = s
15           conts (CRef _ _) = "(ref)"
16           conts (CMisc _ _) = "(misc)"
17 -- |
18 read_ELEMENT :: String -> STM Result [Content i] (Element i)
19 read_ELEMENT tag = liftSTM (ST (\s -> read_ELEMENT' tag s))
20 -- |
21 show_ELEMENT :: String -> [Attribute] -> [Content ()] -> [Content ()]
22 show_ELEMENT tag attr contents = [CElem (Elem tag attr contents) ()]
23 -- |
24 attributes :: Element i -> [Attribute]
25 attributes (Elem _ x _) = x
26 -- |
27 childs :: Element i -> [Content i]
28 childs (Elem _ _ x) = x
```

Para permitir construções sobre elementos de *XML*, implementaram-se funções correspondentes à especificação do DTD sobre elementos opcionais, funções `MAYBE`, sobre elementos em lista, podendo ou não a lista ser vazia, funções `LIST` e `LIST1`, respectivamente.

Listagem 3.5: Módulo Util - construções MAYBE e LIST

```
1  -- |
2  read_MAYBE :: STM Result [Content i] a ->
3      STM Result [Content i] (Maybe a)
4  read_MAYBE st = STM (\s ->
5          ((stateM st s) >>= (\(z1,z2) -> return (z1, return z2)))
6          `mplus` return (s,Nothing) )
7  -- |
8  show_MAYBE :: (a -> [Content ()]) -> Maybe a -> [Content ()]
9  show_MAYBE f = maybe [] f
10 -- |
11 read_LIST :: Eq i => STM Result [Content i] a -> STM Result [Content i] [a]
12 read_LIST st = STM (\s ->
13     let x = stateM st s
14     in case x of
```

```
15          Ok (x1,x2) -> if s == x1 then return (s,[x2])
16                        else let y = stateM (read_LIST st) x1
17                             in case y of
18                                  Ok (y1,y2) -> return (y1, x2:y2)
19                                  Error _ -> return (x1, [x2])
20          Error _ -> return (s, [])
21      )
22  -- |
23  show_LIST :: (a -> [Content ()]) -> [a] -> [Content ()]
24  show_LIST f = concat . map f
25  -- |
26  read_LIST1 :: Eq i => STM Result [Content i] a -> STM Result [Content i] [a]
27  read_LIST1 st = STM (\s ->
28      let x = stateM st s
29      in case x of
30         Ok (x1,x2) -> if s == x1 then return (s,[x2])
31                       else let y = stateM (read_LIST1 st) x1
32                            in case y of
33                                 Ok (y1,y2) -> return (y1, x2:y2)
34                                 Error _ -> return (x1, [x2])
35          Error _ -> fail "empty list"
36      )
37  -- |
38  show_LIST1 :: (a -> [Content ()]) -> [a] -> [Content ()]
39  show_LIST1 = show_LIST
```

Por fim, devido à falta de construções genéricas em *Haskell*, foi necessário implementar funções de leitura de sequência determinadas de elementos, tendo como sufixo o número de elementos a ler. Foi necessário implementar instâncias `Show` e `Eq` para tuplos de 17 elementos.

3.3.2 Implementação de common

Este módulo *Haskell* é a implementação de funções que permitem a manipulação com elementos definidos no ficheiro `common.mod`.

Este módulo também engloba o módulo `Util`, para não ser necessária a sua importação por outros módulos além do `Common`. Esta decisão torna a arquitectura da biblioteca mais semelhante à especificação dado que apenas o módulo `Common` depende do módulo `Util` e todos os outros módulos dependem do módulo `Common`.

Na especificação do MusicXML em DTD são apresentados os vários documentos de forma independente. O uso de entidades presentes no documento `common.mod` em outros documentos é feita sem a importação do `documento.mod` pelos documentos que usam as entidades assumindo que existirá um documento que agrupe todas as definições. Desta forma garante-se que os módulos são autónomos sem dependências e garantindo que não existem entidades por definir.

No entanto, não é possível manter a abordagem de preservar a independência entre as implementações dos vários documentos. Deste modo,

criaram-se apenas as dependências necessárias para resolver a entidades não-declaradas. Para evitar adicionar a dependência da implementação do módulo `Util` nas outras implementações que também dependem da implementação do módulo `Common`, o módulo `Common` agrega as definições do módulo `Util`. Esta arquitectura é intuitiva porque todas as implementações dependem da implementação de `common.mod` e esta depende da implementação de funções utilitárias.

3.3.3 Implementação de outros documentos

A implementações dos documentos `barline.mod`, `link.mod`, `identity.mod`, `direction.mod`, `layout.mod`, `attributes.mod` e `note.mod` dependem das implementações anteriormente referidas.

Identity Na implementação de `identity.mod` é implementada um conjunto de funções para facilitar a alteração da data e nome do software que manipulou/criou o documento em MusicXML.

Listagem 3.6: Módulo Identity - Actualização de data e nome do software

```
1 -- |
2 update_Identification :: ([Software], Encoding_Date) -> Identification ->
      Identification
3 update_Identification x (a,b,c,d,e,f) = (a, b, fmap (update_Encoding x) c, d, e, f)
4 -- |
5 update_Encoding :: ([Software], Encoding_Date) -> Encoding -> Encoding
6 update_Encoding (s,d) _ = (Encoding_1 d) : (fmap Encoding_3 s)
```

Attributes A implementação de `attributes.mod` é feita pelo módulo `Attributes`. Neste módulo estão presentas as definições da clave, tempo, tipo de compasso entre outras. Na especificação do MusicXML não existe a estruturação destes atributos tal como se pressupõe na notação musical.

Note A implementação de `note.mod` é feita pelo módulo Note. Neste módulo está presente a entidade note responsável pela definição das notas. Devido à diferença de expressividade em *Haskell* e em DTD, é necessário criar um nodo Note_ para representar o co-produto presente na entidade Note. O método que se usa para representar co-produtos em *Haskell* é sempre da forma apresentada a seguir, em que se adiciona uma entidade com o mesmo nome com o sufixo _ e os construtores tem o nome da entidade com o sufixo _#, onde # é o número do construtor no co-produto.

Listagem 3.7: Módulo Note

```
1  -- * Note
2  -- |
3  type Note = ((Print_Style, Printout, Maybe CDATA, Maybe CDATA,
4       Maybe CDATA, Maybe CDATA, Maybe CDATA, Maybe Yes_No),
5      (Note_, Maybe Instrument, Editorial_Voice, Maybe Type, [Dot],
6      Maybe Accidental, Maybe Time_Modification, Maybe Stem, Maybe Notehead,
7      Maybe Staff, [Beam], [Notations], [Lyric]))
8  -- |
9  read_Note :: Eq i => STM Result [Content i] Note
10 read_Note = do
11     y <- read_ELEMENT "note"
12     y1 <- read_8 read_Print_Style read_Printout
13                  (read_IMPLIED "dynamics" read_CDATA)
14                  (read_IMPLIED "end-dynamics" read_CDATA)
15                  (read_IMPLIED "attack" read_CDATA)
16                  (read_IMPLIED "release" read_CDATA)
17                  (read_IMPLIED "time-only" read_CDATA)
18                  (read_IMPLIED "pizzicato" read_Yes_No) (attributes y)
19     y2 <- read_13 read_Note_ (read_MAYBE read_Instrument)
20                   read_Editorial_Voice (read_MAYBE read_Type)
21                   (read_LIST read_Dot) (read_MAYBE read_Accidental)
22                   (read_MAYBE read_Time_Modification)
23                   (read_MAYBE read_Stem) (read_MAYBE read_Notehead)
24                   (read_MAYBE read_Staff) (read_LIST read_Beam)
25                   (read_LIST read_Notations) (read_LIST read_Lyric)
26                   (childs y)
27     return (y1,y2)
28
29 show_Note :: Note -> [Content ()]
30 show_Note ((a,b,c,d,e,f,g,h),(i,j,k,l,m,n,o,p,q,r,s,t,u)) =
31     show_ELEMENT "note" (show_Print_Style a ++ show_Printout b ++
32                          show_IMPLIED "dynamics" show_CDATA c ++
33                          show_IMPLIED "end-dynamics" show_CDATA d ++
34                          show_IMPLIED "attack" show_CDATA e ++
35                          show_IMPLIED "release" show_CDATA f ++
36                          show_IMPLIED "time-only" show_CDATA g ++
37                          show_IMPLIED "pizzicato" show_Yes_No h)
38                         (show_Note_ i ++ show_MAYBE show_Instrument j ++
39                          show_Editorial_Voice k ++
40                          show_MAYBE show_Type l ++
41                          show_LIST show_Dot m ++
42                          show_MAYBE show_Accidental n ++
43                          show_MAYBE show_Time_Modification o ++
44                          show_MAYBE show_Stem p ++
45                          show_MAYBE show_Notehead q ++
46                          show_MAYBE show_Staff r ++
47                          show_LIST show_Beam s ++
48                          show_LIST show_Notations t ++
49                          show_LIST show_Lyric u)
50 -- ** Note_
51 -- |
52 data Note_ = Note_1 (Grace, Full_Note, Maybe (Tie, Maybe Tie))
53            | Note_2 (Cue, Full_Note, Duration)
54            | Note_3 (Full_Note, Duration, Maybe (Tie, Maybe Tie))
55            deriving (Eq, Show)
56 -- |
57 read_Note_ :: STM Result [Content i] Note_
58 read_Note_ =
59     (read_Note_aux1 >>= return . Note_1) `mplus`
60     (read_Note_aux2 >>= return . Note_2) `mplus`
```

```
61      (read_Note_aux3 >>= return . Note_3)
62  read_Note_aux1 ::
63      STM Result [Content i] (Grace, Full_Note, Maybe (Tie, Maybe Tie))
64  read_Note_aux1 = do
65      y1 <- read_Grace
66      y2 <- read_Full_Note
67      y3 <- read_MAYBE read_Note_aux4
68      return (y1,y2,y3)
69  read_Note_aux2 :: STM Result [Content i] (Cue, Full_Note, Duration)
70  read_Note_aux2 = do
71      y1 <- read_Cue
72      y2 <- read_Full_Note
73      y3 <- read_Duration
74      return (y1,y2,y3)
75  read_Note_aux3 ::
76      STM Result [Content i] (Full_Note, Duration, Maybe (Tie, Maybe Tie))
77  read_Note_aux3 = do
78      y1 <- read_Full_Note
79      y2 <- read_Duration
80      y3 <- read_MAYBE read_Note_aux4
81      return (y1,y2,y3)
82  read_Note_aux4 :: STM Result [Content i] (Tie, Maybe Tie)
83  read_Note_aux4 = do
84      y1 <- read_Tie
85      y2 <- read_MAYBE read_Tie
86      return (y1,y2)
87  -- |
88  show_Note_ :: Note_ -> [Content ()]
89  show_Note_ (Note_1 (a,b,c)) =
90      show_Grace a ++ show_Full_Note b ++ show_MAYBE show_Note_aux1 c
91  show_Note_ (Note_2 (a,b,c)) =
92      show_Cue a ++ show_Full_Note b ++ show_Duration c
93  show_Note_ (Note_3 (a,b,c)) =
94      show_Full_Note a ++ show_Duration b ++ show_MAYBE show_Note_aux1 c
95  -- |
96  show_Note_aux1 :: (Tie, Maybe Tie) -> [Content ()]
97  show_Note_aux1 (a,b) = show_Tie a ++ show_MAYBE show_Tie b
```

3.3.4 Implementação de score

Devido à falta de expressividade em *Haskell* não é possível implementar o documento `score.mod` da mesma forma que é especificado em DTD. Isto deve-se a que usa definições de expressões condicionais, o que não é possível em *Haskell*. A implementação em *Haskell* num mesmo módulo gera conflito, por as duas implementações não serem isomorfas.

Este problema encontra-se na definição das entidades `score-partwise` e `score-timewise`. A primeira define-se uma sequência de `part` que contém uma sequência de `measure` sendo definida por `%music-data`. A segunda define-se por uma sequência de `measure` que contém uma sequência de `part` sendo definida por `%music-data`. As entidades `measure` e `part` tem informação adicional diferente.

A solução para resolver este problema passa por efectuar a definição das entidades em conflito em módulos independentes. Desta forma os módulos `Partwise` e `Timewise` contém estas definições que não estão na especificação do MusicXML. Desta forma garante-se a definição das entidades presentes na especificação, sem entrarem em conflito.

3.3.5 Implementação de partwise

A implementação de `partwise.dtd` como foi falado anteriormente contém a definição da entidade `score-partwise`. Além da implementação das entidades presentes na especificação é implementada a sugestão dada para o cabeçalho pela função `doctype`.

Listagem 3.8: Módulo Partwise - DocTypeDecl

```
1 -- |
2 doctype :: DocTypeDecl
3 doctype = DTD "score-partwise"
4   (Just (PUBLIC (PubidLiteral "-//Recordare//DTD MusicXML 2.0 Partwise//EN")
5                 (SystemLiteral "http://www.musicxml.org/dtds/partwise.dtd")))
6     []
```

A implementação da entidade `score-partwise` é a raiz de documentos MusicXML que são válidos perante a especificação em `partwise.dtd`.

3.3.6 Implementação de timewise

A implementação de `timewise.dtd` como foi falado anteriormente contém a definição da entidade `score-timewise`. Além da implementação das entidades presentes na especificação é implementada a sugestão dada para o cabeçalho pela função `doctype`.

Listagem 3.9: Módulo Timewise - DocTypeDecl

```
1 -- |
2 doctype :: DocTypeDecl
3 doctype = DTD "score-timewise"
4   (Just (PUBLIC (PubidLiteral "-//Recordare//DTD MusicXML 2.0 Timewise//EN")
5                 (SystemLiteral "http://www.musicxml.org/dtds/timewise.dtd")))
6     []
```

A implementação da entidade `score-timewise` é a raiz de documentos MusicXML que são válidos perante a especificação em `timewise.dtd`.

3.3.7 Implementação de opus

A sugestão para o cabeçalho dos documentos em MusicXML que se pretende que sejam válidos por `opus.dtd`, implementou-se com a função `doctype`, como se mostra.

Listagem 3.10: Módulo Opus

```
1 -- |
2 doctype :: DocTypeDecl
3 doctype = DTD "opus"
4   (Just (PUBLIC (PubidLiteral "-//Recordare//DTD MusicXML 2.0 Opus//EN")
5                 (SystemLiteral "http://www.musicxml.org/dtds/opus.dtd")))
6     []
```

A implementação das entidades presentes em `opus.dtd` e das funções sobre as entidades seguem os mesmos parâmetros, das implementações descritas anteriormente. A entidade raiz do documento é opus, implementada pelo tipo de dados Opus.

Não é apresentada uma função que manipule os documentos para os quais este documento referencia.

3.3.8 Implementação de container

A sugestão para o cabeçalho dos documentos em MusicXML que se pretende que sejam válidos por `opus.dtd`, implementou-se com a função `doctype`, como se mostra.

Listagem 3.11: Módulo Container

```
1 -- |
2 doctype :: DocTypeDecl
3 doctype = DTD "container"
4   (Just (PUBLIC (PubidLiteral "-//Recordare//DTD MusicXML 2.0 Container//EN")
5                 (SystemLiteral "http://www.musicxml.org/dtds/container.dtd")))
6     []
```

A implementação das entidades presentes em `opus.dtd` e das funções sobre as entidades seguem os mesmos parâmetros, das implementações descritas anteriormente. A entidade raiz do documento é opus, implementada pelo tipo de dados Opus.

Não é apresentada uma função que manipule os documentos para os quais este documento referencia.

3.4 Implementação da API

São definidos tipos de dados que correspondem à especificação do MusicXML. A especificação engloba a especificação de música orientada à partitura, definida em `partwise.dtd`, de música orientada ao compasso, definida em

timewise.dtd, de agrupamento de documentos MusicXML, não comprimidos, definido em opus.dtd e de agrupamento comprimido de documentos MusicXML, definido em container.dtd.

Listagem 3.12: Módulo MusicXML - Tipos de dados

```
1  -- * MusicXML
2  -- |
3  data ScoreDoc = Partwise Score_Partwise
4                | Timewise Score_Timewise
5                  deriving (Eq, Show)
6  data MusicXMLDoc = Score ScoreDoc
7                   | Opus Opus
8                   | Container Container
9                     deriving (Eq, Show)
10 -- |
11 data MusicXMLRec = MusicXMLRec (Map.Map FilePath MusicXMLDoc)
12                    deriving (Eq, Show)
```

Existem tipos de dados equivalentes à especificação descrita e funções para os mesmos, tendo como o sufixo o nome do tipo de documento. As funções sobre a especificação em partwise.dtd tem como sufixo _Partwise e as funções sobre a especificação em timewise.dtd tem como sufixo _Timewise. O mesmo acontece para os ficheiros opus.dtd e container.dtd, que tem sufixos _Opus e _Container, respectivamente.

Listagem 3.13: Módulo MusicXML - Funções básicas

```
1  -- |
2  read_DOCUMENT :: STM Result [Content Posn] a -> Document Posn -> Result a
3  read_DOCUMENT r (Document _ _ x _) = stateM r [CElem x noPos] >>= (return.snd)
4  -- |
5  read_MusicXML_Partwise :: Document Posn -> Result Score_Partwise
6  read_MusicXML_Partwise = read_DOCUMENT read_Score_Partwise
7  -- |
8  read_MusicXML_Timewise :: Document Posn -> Result Score_Timewise
9  read_MusicXML_Timewise = read_DOCUMENT read_Score_Timewise
10 -- |
11 read_MusicXML_Opus :: Document Posn -> Result Opus
12 read_MusicXML_Opus = read_DOCUMENT read_Opus
13 -- |
14 read_MusicXML_Container :: Document Posn -> Result Container
15 read_MusicXML_Container = read_DOCUMENT read_Container
16 -- |
17 show_DOCUMENT :: DocTypeDecl -> (t -> [Content i]) -> t -> Result (Document i)
18 show_DOCUMENT doct f doc =
19     case f doc of
20         [(CElem processed _)] ->
21             return (Document (Prolog (Just xmldecl) []
22                     (Just doct) []) [] processed [])
23         _                     -> fail "internal error"
24 -- |
25 show_MusicXML_Partwise :: Score_Partwise -> Result (Document ())
26 show_MusicXML_Partwise =
27     show_DOCUMENT Partwise.doctype show_Score_Partwise
28 -- |
29 show_MusicXML_Timewise :: Score_Timewise -> Result (Document ())
30 show_MusicXML_Timewise =
```

```
31      show_DOCUMENT Partwise.doctype show_Score_Timewise
32  -- |
33  show_MusicXML_Opus :: Opus -> Result (Document ())
34  show_MusicXML_Opus x =
35      show_DOCUMENT Opus.doctype show_Opus x
36  -- |
37  show_MusicXML_Container :: Container -> Result (Document ())
38  show_MusicXML_Container x =
39      show_DOCUMENT Container.doctype show_Container x
40  -- |
41  update_MusicXML_Partwise :: ([Software], Encoding_Date) ->
42      Score_Partwise -> Score_Partwise
43  update_MusicXML_Partwise = update_Score_Partwise
44  -- |
45  update_MusicXML_Timewise :: ([Software], Encoding_Date) ->
46      Score_Timewise -> Score_Timewise
47  update_MusicXML_Timewise = update_Score_Timewise
```

Existem também funções que manipulam ficheiros, conteúdos de ficheiros e documentos. Estas funções usam o primeiro argumento para especificar a acção pretendida, sendo estas funções paramétricas.

Listagem 3.14: Módulo MusicXML - Interface com HaXml

```
1   -- |
2   read_CONTENTS :: (Document Posn -> Result a) ->
3       FilePath -> Prelude.String -> Result a
4   read_CONTENTS f filepath contents =
5       either fail f (xmlParse' filepath contents)
6   -- |
7   show_CONTENTS :: (a -> Result (Document i)) -> a -> Prelude.String
8   show_CONTENTS f musicxml =
9       maybe (fail "undefined error")
10          (renderStyle (Style LeftMode 100 1.5) . document)
11          ((toMaybe . f) musicxml)
12  -- |
13  read_FILE :: (Document Posn -> Result a) -> FilePath -> IO (Result a)
14  read_FILE f filepath = do
15      exists <- doesFileExist filepath
16      case exists of
17          True -> readFile filepath >>= return . (read_CONTENTS f) filepath
18          False -> (return . fail) ("no file: " ++ show filepath)
19  -- |
20  show_FILE :: (a -> Result (Document i)) -> FilePath -> a -> IO ()
21  show_FILE f filepath musicxml =
22      writeFile filepath (show_CONTENTS f musicxml)
```

Para facilitar a utilização das funções acima apresentadas foram criadas funções sobre `MusicXMLDoc`. Deste modo é possível ter funções para todo o MusicXML.

Listagem 3.15: Módulo MusicXML - Interface com MusicXMLDoc

```
1   -- |
2   read_MusicXMLDoc :: Document Posn -> Result MusicXMLDoc
3   read_MusicXMLDoc doc =
4     (read_DOCUMENT read_Score_Partwise doc >>= return .Score .Partwise) `mplus`
5     (read_DOCUMENT read_Score_Timewise doc >>= return .Score .Timewise) `mplus`
6     (read_DOCUMENT read_Opus doc >>= return . Opus) `mplus`
```

```
7   (read_DOCUMENT read_Container doc >>= return . Container) `mplus`
8   fail "<score-partwise> or <score-timewise> or <opus> or <container>"
9   -- |
10  show_MusicXMLDoc :: MusicXMLDoc -> Result (Document ())
11  show_MusicXMLDoc (Score (Partwise doc)) = show_MusicXML_Partwise doc
12  show_MusicXMLDoc (Score (Timewise doc)) = show_MusicXML_Timewise doc
13  show_MusicXMLDoc (Opus doc) = show_MusicXML_Opus doc
14  show_MusicXMLDoc (Container doc) = show_MusicXML_Container doc
15  -- |
16  update_MusicXMLDoc :: ([Software], Encoding_Date) ->
17      MusicXMLDoc -> MusicXMLDoc
18  update_MusicXMLDoc x (Score (Partwise doc)) =
19      Score (Partwise (update_MusicXML_Partwise x doc))
20  update_MusicXMLDoc x (Score (Timewise doc)) =
21      Score (Timewise (update_MusicXML_Timewise x doc))
22  update_MusicXMLDoc _ y = y
23  -- |
24  read_MusicXMLRec :: FilePath -> IO (Map.Map FilePath MusicXMLDoc)
25  read_MusicXMLRec f = do
26      x <- read_FILE read_MusicXMLDoc f >>= \a -> return (f, a)
27      case isOK (snd x) of
28          True -> do
29              xs <- mapM (\f' -> read_FILE read_MusicXMLDoc f'
30                          >>= \a -> return (f', a))
31                    (Text.XML.MusicXML.getFiles (fromOK (snd x)))
32              return (Map.map fromOK (Map.filter isOK (Map.fromList (x:xs))))
33          False -> return (Map.empty)
```

É declarado o cabeçalho dos documentos *XML* com a função `xmldecl`. Esta função apenas indica que o *XML* escrito está conforme a versão 1.0.

Listagem 3.16: Módulo MusicXML - *XML* Declaration

```
1  -- |
2  xmldecl :: XMLDecl
3  xmldecl = XMLDecl "1.0" Nothing Nothing
```

A especificação do MusicXML recorre a uma definição para as datas definidas de acordo com o ISO 8601, onde as datas são representadas no formato `yyyy-mm-dd`. Este formato divide a data em três áreas, onde a primeira tem 4 dígitos, a segunda tem dois dígitos e a última tem dois dígitos, utilizando o hífen como separador. Para simplificar a manipulação de datas, isto é, a criação de datas para indicar a data em que o ficheiro é criado/modificado, foi criada uma função que determina a data no formato especificado.

Listagem 3.17: Módulo MusicXML - getTime

```
1  -- | getTime uses old-time library. At future versions can be defined as:
2  --  @getTime :: IO Prelude.String@
3  --  @getTime = getCurrentTime >>= return . show . utctDay@
4  getTime :: IO Encoding_Date
5  getTime = getClockTime >>= toCalendarTime >>=
6      return . (\(CalendarTime yyyy mm dd _ _ _ _ _ _ _ _ _) ->
7          show4 yyyy ++ "-" ++ show2 (fromEnum mm + 1) ++ "-" ++ show2 dd)
8  -- |
9  show2, show3, show4 :: Int -> Prelude.String
10 show2 x | (x < 0) = show2 (-x)
11         | otherwise = case show x of; [a] -> '0':a:[]; y -> y
```

```
12 show3 x | (x < 0) = show3 (-x)
13         | otherwise = case show2 x of; [a,b] -> '0':a:b:[]; y -> y
14 show4 x | (x < 0) = show4 (-x)
15         | otherwise = case show3 x of; [a,b,c] -> '0':a:b:c:[]; y -> y
```

3.5 Exemplos de Utilização

A utilização da biblioteca MusicXML será feita com recurso ao módulo `Text.XML.MusicXML`. Nele são definidas funções que tornam possível a manipulação de documentos em MusicXML.

É apresentado um exemplo, o qual efectua a leitura de um documento MusicXML, actualizando os parâmetros referentes ao software e data de codificação do documento.

Listagem 3.18: Update: Exemplo de utilização do MusicXML

```
1  > module Main where
2  > import Text.XML.MusicXML
3  > filein = "Recordare/partwise/elite.xml"
4  > fileout = "out.xml"
5  > main = do
6  >   x <- read_FILE read_MusicXMLDoc filein
7  >   case isOK x of
8  >       True -> do
9  >           time <- getTime
10 >           let y = update_MusicXMLDoc (["My Software"], time) (fromOK x)
11 >           show_FILE show_MusicXMLDoc fileout y
12 >       False -> return ()
```

3.6 Casos de estudo

Foram aplicadas a função de importação de um ficheiro para a definição do tipo de dados de MusicXML implementada e a função de exportação que escreve num ficheiro *XML* de acordo com a especificação MusicXML dados elementos do tipo de dados de MusicXML implementado. Isto quer dizer que, nos testes, apenas são feitas execuções para uma análise da cobertura de código, usando a ferramenta HPC para a análise.

Foram usados três casos de estudo, com os nomes de *Recordare*, *Wikifonia* e *Gutenberg* que se refere respectivamente a fonte dos documentos utilizados. O primeiro caso de estudo apresenta documentos de tamanho diversificado, e nas formas `partwise` e `timewise`. Esta última forma foi obtida pela aplicação da transformação `parttime.xsl`. O segundo caso de estudo apresenta muitos ficheiros de pequenas dimensões, e no último caso

de estudo com o nome de Gutenberg são utilizados poucos documentos de grandes dimensões.

É apresentada uma aplicação que especifica o código a ser executado pelos vários casos de estudo. A aplicação recebe o nome de documentos em MusicXML de modo a que cada documento é importado para uma especificação em *Haskell* e de seguida exportado para outro documento MusicXML com informação equivalente. Apenas se notaram diferenças nos documentos de entrada e de saída para o *layout* do documento e os comentários presentes.

3.6.1 Recordare

Este caso de estudo utiliza os ficheiros disponibilizados pela Recordare, na sua especificação do MusicXML. Estes documentos são um conjunto diversificado de documentos, com variada notação musical de variadas dimensões.

Apresentamos o ficheiro que foi utilizado como parâmetro da aplicação para a execução de testes sobre este caso de estudo. Este ficheiro apresenta a lista de documentos em MusicXML.

Listagem 3.19: Lista de documentos MusicXML do caso de estudo Recordare

```
1  ../examples/Recordare/partwise/Echigo-Jishi.xml
2  ../examples/Recordare/partwise/elite.xml
3  ../examples/Recordare/partwise/ActorPreludeSample.xml
4  ../examples/Recordare/partwise/BeetAnGeSample.xml
5  ../examples/Recordare/partwise/Binchois.xml
6  ../examples/Recordare/partwise/BrahWiMeSample.xml
7  ../examples/Recordare/partwise/BrookeWestSample.xml
8  ../examples/Recordare/partwise/Chant.xml
9  ../examples/Recordare/partwise/DebuMandSample.xml
10 ../examples/Recordare/partwise/Dichterliebe01.xml
11 ../examples/Recordare/partwise/FaurReveSample.xml
12 ../examples/Recordare/partwise/MahlFaGe4Sample.xml
13 ../examples/Recordare/partwise/MozaChloSample.xml
14 ../examples/Recordare/partwise/MozaVeilSample.xml
15 ../examples/Recordare/partwise/MozartPianoSonata.xml
16 ../examples/Recordare/partwise/MozartTrio.xml
17 ../examples/Recordare/partwise/Saltarello.xml
18 ../examples/Recordare/partwise/SchbAvMaSample.xml
19 ../examples/Recordare/partwise/Telemann.xml
20 ../examples/Recordare/timewise/Echigo-Jishi.xml
21 ../examples/Recordare/timewise/elite.xml
22 ../examples/Recordare/timewise/ActorPreludeSample.xml
23 ../examples/Recordare/timewise/BeetAnGeSample.xml
24 ../examples/Recordare/timewise/Binchois.xml
25 ../examples/Recordare/timewise/BrahWiMeSample.xml
26 ../examples/Recordare/timewise/BrookeWestSample.xml
27 ../examples/Recordare/timewise/Chant.xml
28 ../examples/Recordare/timewise/DebuMandSample.xml
29 ../examples/Recordare/timewise/Dichterliebe01.xml
30 ../examples/Recordare/timewise/FaurReveSample.xml
31 ../examples/Recordare/timewise/MahlFaGe4Sample.xml
32 ../examples/Recordare/timewise/MozaChloSample.xml
33 ../examples/Recordare/timewise/MozaVeilSample.xml
```

```
34 ../examples/Recordare/timewise/MozartPianoSonata.xml
35 ../examples/Recordare/timewise/MozartTrio.xml
36 ../examples/Recordare/timewise/Saltarello.xml
37 ../examples/Recordare/timewise/SchbAvMaSample.xml
38 ../examples/Recordare/timewise/Telemann.xml
```

A execução da aplicação para os documentos MusicXML referidos obteve o resultado presente em apêndice. Neste resultado observa-se o tempo de execução global. Verificamos que a execução para a lista de documentos em MusicXML chega quase a um minuto[1].

Apresentamos de seguida os resultados obtidos pelo HPC. Esta ferramenta permite-nos analisar que funções são executadas e qual a sua frequência. O HPC devolve os resultados em perspectiva das funções, das alternativas possíveis e do total de expressões existentes.

	Percentagem da cobertura	Razão da cobertura
Funções	67 %	725/1077
Alternativas	36 %	208/568
Expressões	58 %	7708/13190

Tabela 3.1: Resultados do HPC para o caso de estudo Recordare com 38 documentos

Com recurso à aplicação `hp2ps` após a execução da aplicação com os parâmetros `+RTS -hT` é possível obter uma imagem referente ao uso da memória pela aplicação. Foi criada a seguinte imagem, onde se identifica a execução dos ficheiros referidos.

3.6.2 Wikifonia

Este caso de estudo utiliza os ficheiros disponibilizados pela Wikifonia[56]. Estes documentos são um conjunto diversificado sob a perspectiva de tema. No entanto tem uma limitada notação musical, sendo de referir que apenas existe uma voz representada por documento.

Apresentamos o ficheiro que foi utilizado como parâmetro da aplicação para a execução de testes sobre este caso de estudo. Este ficheiro em apêndice apresenta a lista de documentos em MusicXML.

A execução da aplicação para os documentos MusicXML referidos obteve o resultado presente no apêndice. Neste resultado observa-se o tempo de

[1] A máquina onde foram realizados os testes tem um processador com 1.5GHz e um memória RAM com 512MB, onde se usa a versão 6.8.2 do GHC num ambiente Windows XP com ferramentas GNU

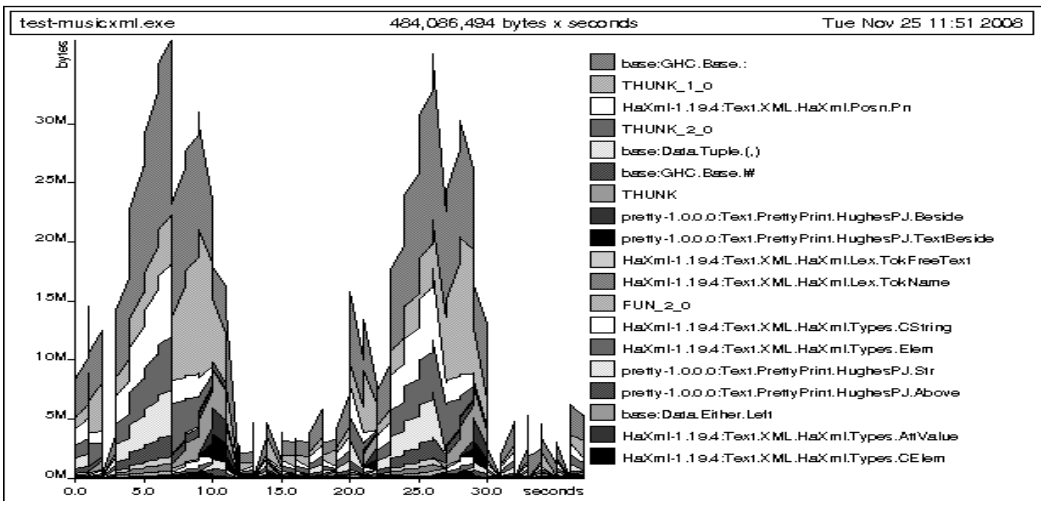


Figura 3.2: Heap no caso de estudo Recordare

execução global. Para os 221 documentos foi necessário quase dois minutos e meio de processamento.

Apresentamos de seguida os resultados obtidos pelo HPC. Esta ferramenta permite-nos analisar que funções são executadas e qual a sua frequência. O HPC devolve os resultados em perspectiva das funções, das alternativas possíveis e do total de expressões existentes.

	Percentagem da cobertura	Razão da cobertura
Funções	62 %	672/1075
Alternativas	27 %	155/568
Expressões	52 %	6881/13167

Tabela 3.2: Resultados do HPC para o caso de estudo Wikifonia com 221 documentos

Com recurso à aplicação `hp2ps` após a execução da aplicação com os parâmetros `+RTS -hT` é possível obter uma imagem referente ao uso da memória pela aplicação. Foi criada a seguinte imagem, onde se identifica a execução dos ficheiros referidos.

3.6.3 Gutenberg

Este caso de estudo utiliza os ficheiros disponibilizados pelo Gutenberg[9]. Estes documentos são um conjunto de documentos de grandes dimensões, o que necessita de maior capacidade de processamento.

Apresentamos o ficheiro que foi utilizado como parâmetro da aplicação para a execução de testes sobre este caso de estudo. Este ficheiro apresenta a lista de documentos em MusicXML.

Listagem 3.20: Lista de documentos MusicXML do caso de estudo Gutenberg

```
1 ../examples/Gutenberg/11755-Complete.utf8.xml
2 ../examples/Gutenberg/11001-Complete.utf8.xml
3 ../examples/Gutenberg/12149-Complete.utf8.xml
4 ../examples/Gutenberg/12695-complete.utf8.xml
5 ../examples/Gutenberg/13473-all.utf8.xml
```

A execução da aplicação para os documentos MusicXML referidos obteve o resultado presente no apêndice. Neste resultado observa-se o tempo de execução global. Na execução dos 5 documentos referidos foram necessários quase seis minutos e meio de processamento. Durante testes preliminares foi incluído um documento MusicXML, para o qual o computador necessitava de fazer *swaping*, por haver falta de memória.

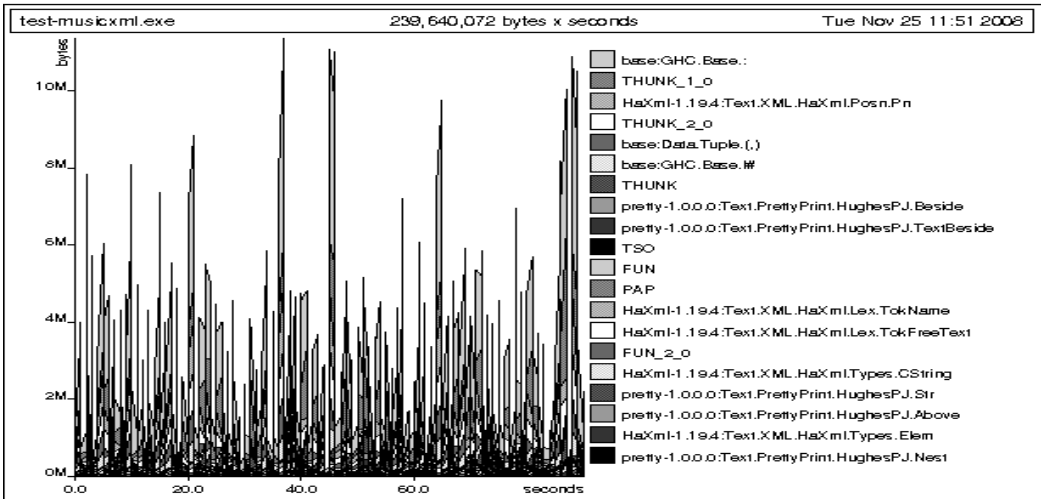


Figura 3.3: Heap no caso de estudo Wikifonia

Apresentamos de seguida os resultados obtidos pelo HPC. Esta ferramenta permite-nos analisar que funções são executadas e qual a sua frequência. O HPC devolve os resultados em perspectiva das funções, das alternativas possíveis e do total de expressões existentes.

	Percentagem da cobertura	Razão da cobertura
Funções	39 %	423/1076
Alternativas	20 %	115/564
Expressões	30 %	4001/13098

Tabela 3.3: Resultados do HPC para o caso de estudo Gutenberg com 5 documentos

Com recurso à aplicação `hp2ps` após a execução da aplicação com os parâmetros `+RTS -hT` é possível obter uma imagem referente ao uso da memória pela aplicação. Foi criada a seguinte imagem, onde se identifica a execução dos ficheiros referidos.

3.6.4 Escalabilidade

Foram selecionados alguns documentos, com o objectivo de testar a biblioteca com seis exemplos reais, dos quais se apresentam as primeiras páginas de três a seguir, e em apêndice as primeiras páginas dos restantes. O primeiro exemplo refere-se a uma partitura tradicional japonesa, onde é incluída a letra, tendo apenas uma página. O segundo exemplo tem a restrição de ter apenas uma voz. O terceiro exemplo é uma pequena obra musical com várias vozes e instrumentos, com quatro páginas. O quarto exemplo é um extracto com quatro páginas de uma obra musical com 22 instrumentos. O quinto e sexto elementos são obras musicais longas com vários instrumentos, tendo a 40 e 47 páginas respectivamente. Devido a que o MusicXML não determina o número de páginas, os mesmos documentos quando importados pelo Finale obtem-se cerca de 70 páginas para os dois últimos documentos.

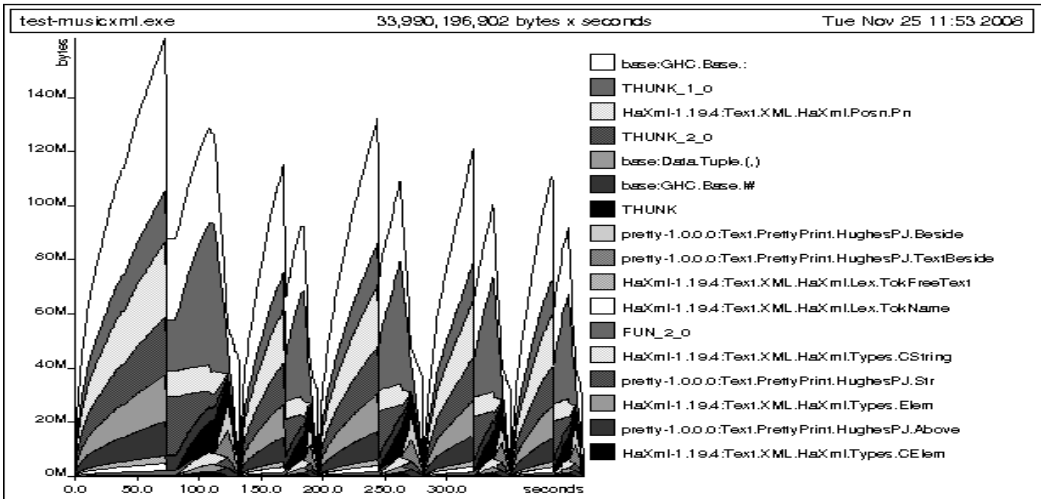
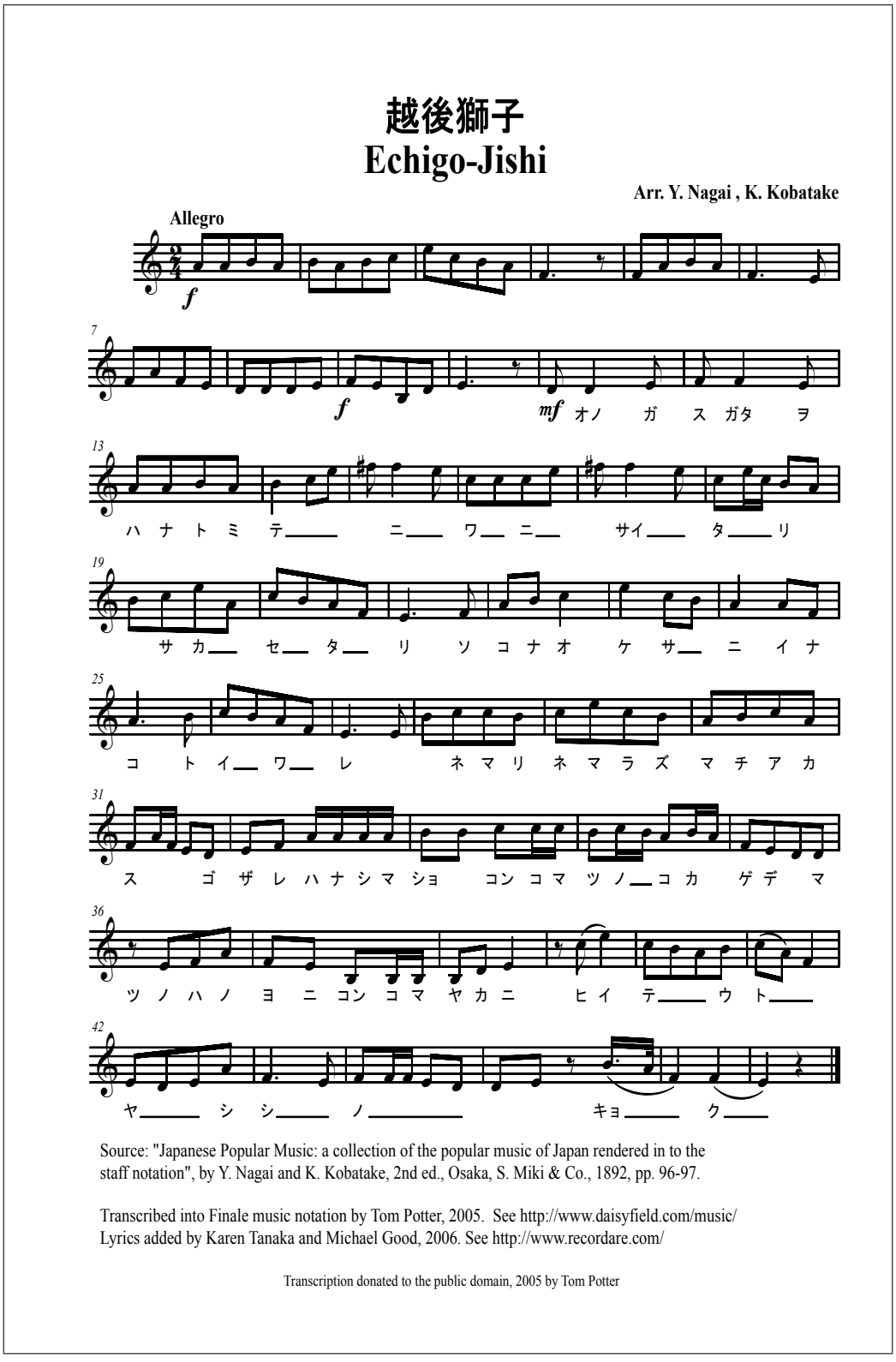
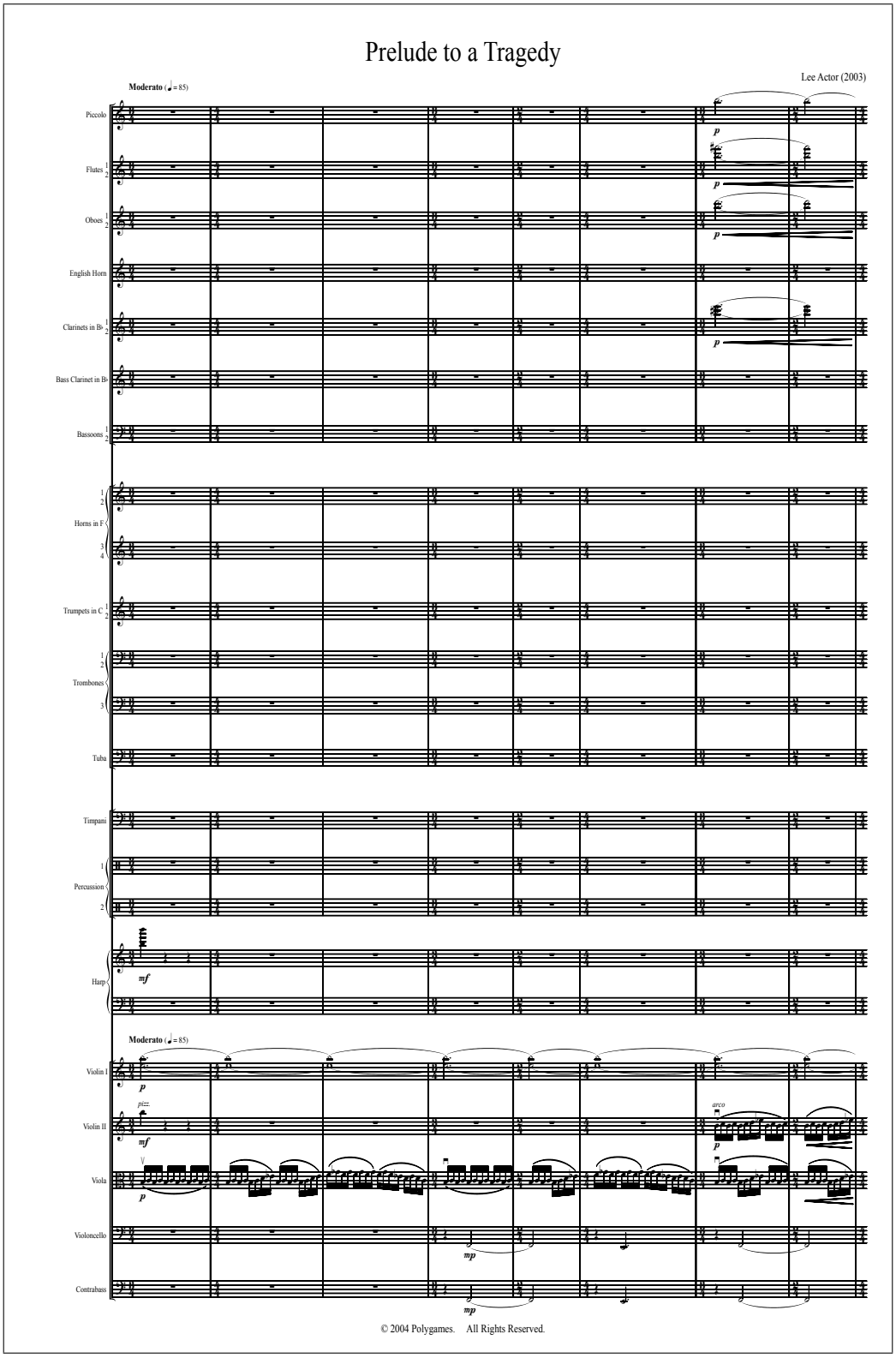
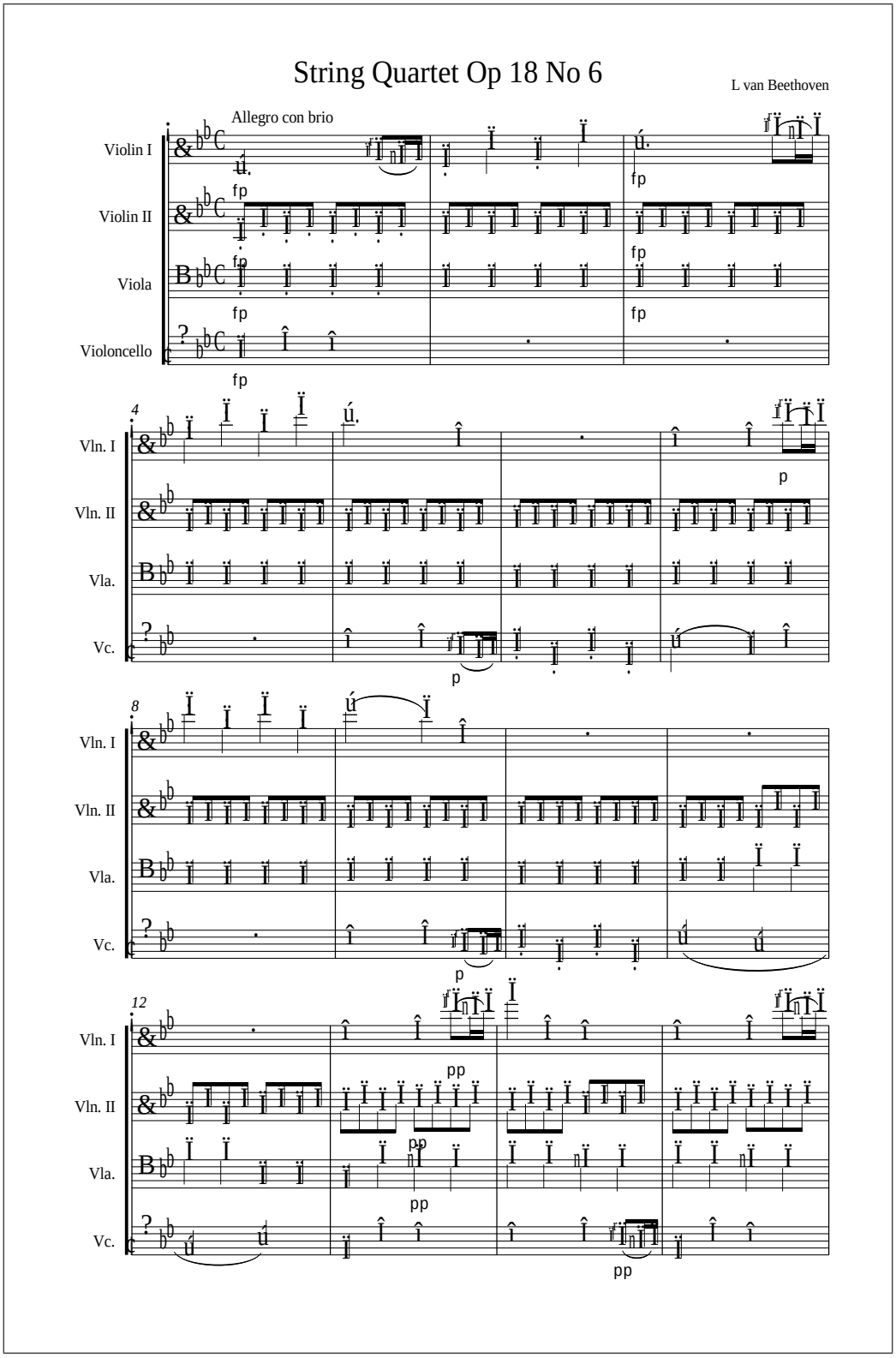


Figura 3.4: Heap no caso de estudo Gutenberg


越後獅子
Echigo-Jishi
Arr. Y. Nagai , K. Kobatake
Allegro
f
7
f
mf オノ ガ ス ガタ ヲ
13
ハ ナ ト ミ テ ニ ワ ニ サイ タ リ
19
サ カ セ タ リ ソ コ ナ オ ケ サ ニ イ ナ
25
コ ト イ ワ レ ネ マ リ ネ マ ラ ズ マ チ ア カ
31
ス ゴ ザ レ ハ ナ シ マ ショ コン コ マ ツ ノ コ カ ゲ デ マ
36
ツ ノ ハ ノ ヨ ニ コン コ マ ヤ カ ニ ヒ イ テ ウ ト
42
ヤ シ シ ノ キョ ク
Source: "Japanese Popular Music: a collection of the popular music of Japan rendered in to the staff notation", by Y. Nagai and K. Kobatake, 2nd ed., Osaka, S. Miki & Co., 1892, pp. 96-97.
Transcribed into Finale music notation by Tom Potter, 2005. See http://www.daisyfield.com/music/
Lyrics added by Karen Tanaka and Michael Good, 2006. See http://www.recordare.com/
Transcription donated to the public domain, 2005 by Tom Potter


Prelude to a Tragedy
Lee Actor (2003)
Moderato (♩= 85)
Piccolo
Flutes 1 2
Oboes 1 2
English Horn
Clarinets in B♭ 1 2
Bass Clarinet in B♭
Bassoons 1 2
Horns in F 1 2 3 4
Trumpets in C 1 2
Trombones 1 2 3
Tuba
Timpani
Percussion 1 2
Harp
Violin I
Violin II
Viola
Violoncello
Contrabass
p
mf
mp
pizz.
arco
© 2004 Polygames. All Rights Reserved.


String Quartet Op 18 No 6
L van Beethoven
Allegro con brio
Violin I
Violin II
Viola
Violoncello
Vln. I
Vln. II
Vla.
Vc.
fp
p
pp

A tabela apresentada apresenta uma pequena lista de documentos, com informação relativa ao tamanho, à memória utilizada, segundo o hpc, e ao tempo utilizado calculado pelo comando `time`. A última coluna da tabela refere-se à análise de cobertura isolada proporcionada por cada documento, medida em percentagem. O triplo contem a informação da percentagem em relação às funções, às alternativas e às expressões.

Após uma análise destes dados e criação das figuras 3.5 e 3.6, pode-se inferir que a relação entre o tamanho dos ficheiros e a memória utilizada é aproximadamente linear. No entanto, a relação entre o tamanho dos ficheiros e o tempo de processamento é exponencial. Quanto à análise da cobertura de código não é possível inferir conclusões. Um resultado que obtemos, quanto à dispersão de resultados na análise de cobertura, deve-se que o tamanho dos documentos é mais influenciado pela repetição das mesmas entidades do que pela existência de novas entidades.

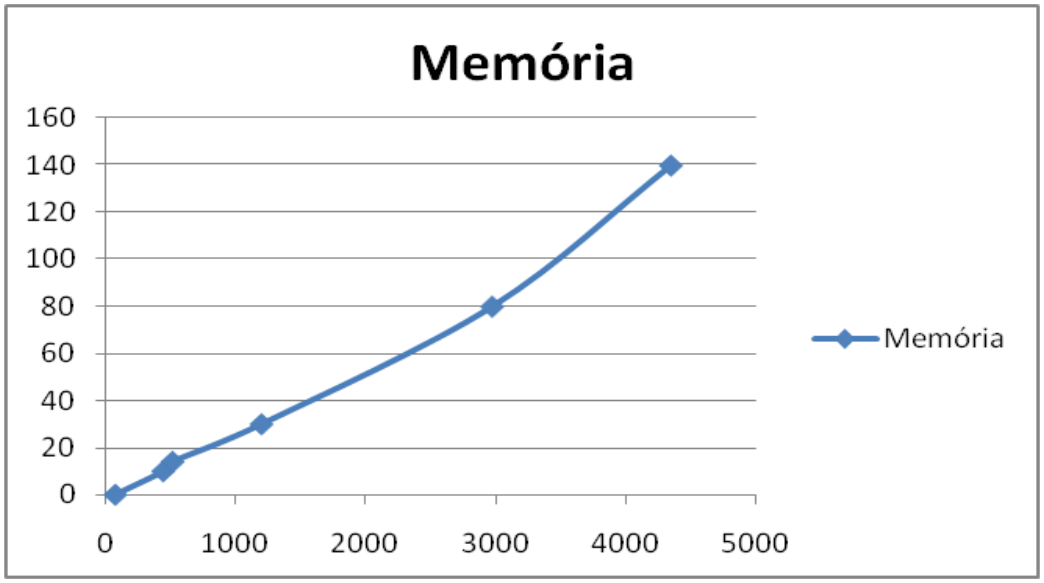


Figura 3.5: Relação entre tamanho(KB) e memória utilizada(MB)

Estes resultados foram obtidos numa máquina com 512MB de memória RAM e 1.5Gz correndo Windows XP com ferramentas da GNU. Devido a limitações de hardware, nomeadamente a falta de memória, não foi possível o processamento de documentos em MusicXML com tamanho superior a 4.5MB.

Nome documento	Tamanho	Memória usada	Tempo usado	Cobertura
Echigo-Jishi	74KB	1.6 MB	0.741s	61-29-50
Blue Danube	443KB	10MB	2.093s	46-17-36
elite	514KB	14MB	2.964s	47-20-38
ActorPreludeSample	1199KB	30MB	10.305s	59-26-47
13473-all.utf8	2974KB	80MB	58.494s	61-28-49
11755-Complete.utf8	4350KB	140MB	1m58.821s	61-29-50

Tabela 3.4: Memória e tempo utilizados na execução dos testes

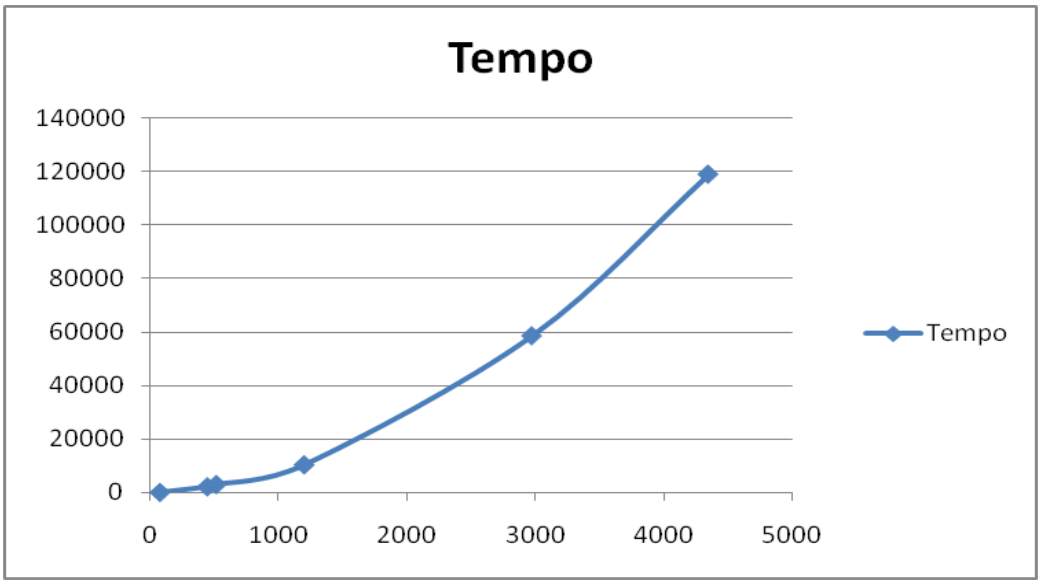


Figura 3.6: Relação entre tamanho(KB) e tempo de execução(ms)

3.7 Sumário

A biblioteca *MusicXML* permite ler e escrever documentos MusicXML na versão 2.0. A implementação foi sistematizada, não se interpretando de entidades presentes para garantir o isomorfismo entre a especificação em DTD e a implementação em *Haskell*. Com o propósito de manter a biblioteca eficiente redefiniram-se os tipos de dados com recurso ao construtor `type` em vez de se recorrer ao construtor `datatype`. A execução de testes obteve resultados satisfatórios, sendo processados documentos com mais de 40 páginas.

O MusicXML é um formato extenso, assim como a biblioteca musicxml pelo que propõe-se uma estratificação do mesmo facilitando a manipulação de documentos em MusicXML.

Capítulo 4

hamusic: Suporte à Análise Musical

Este capítulo apresenta uma biblioteca desenvolvida em *Haskell* que suporta a análise musical. Sobre o formato MusicXML é apresentada uma definição abstracta de notação musical e uma estratificação da notação MusicXML muito semelhante à notação abstracta. Também é apresentada uma linguagem para a análise musical independente da notação musical. Por fim é apresentada a arquitectura para um motor de expressões regulares usando notação musical.

Figura 4.1: Ferramentas em *hamusic*

4.1 Musica Abstracta

Apresenta-se uma notação de música abstracta. Nesta notação concebeu-se uma divisão conceptual entre a noção de tonalidade e a noção de ritmo.

Usa-se uma representação equivalente a uma sequência de notação musical, para a representação da variação da tonalidade e da variação rítmica.

Desta forma, além de se conceber a notação de Motivo, pela qual as várias camadas conceptuais de notação musical usam, também é possível definir a música de forma absoluta e/ou de forma relativa. A forma absoluta na definição da notação musical pode apresentar-se de forma simbólica, sendo a mais usual no tratamento da notação musical, e de forma numérica. A representação da notação musical em forma relativa apenas é possível numericamente. No entanto a representação numérica tem uma base presente na notação musical.

Figura 4.2: Música Abstracta

Esta última representação, a representação numérica, vai ter mais importância no estudo da notação musical por ser mais fácil a análise musical.

4.1.1 Motivo

Esta camada conceptual ajuda a definição formal da notação musical. Pretende-se com esta camada separar elementos que estão presentes ao longo da notação musical de elementos que atribuem e/ou modificam características.

Desta forma aparece uma camada para a representação musical que consiste num par entre propriedades aplicáveis a elementos presentes numa sequência.

4.1.2 Motivo Melódico

O motivo melódico consiste na sequência de variação de frequências. A forma absoluta para representação de frequência são designadas por notas musicais. Existem na notação ocidental sete notas musicais que em português tem o nome de Dó, Ré, Mi, Fá, Sol, Lá e Si. Ao intervalo de sete notas musicais chama-se oitava que tem 12 meios-tons.

É possível definir a sequência de frequências de três diferentes formas, em modo absoluto e duas em modo relativo. A mais usual é a definição de frequência equivalente ao nome da nota musical. A tradução da notação simbólica para um modelo matemático em base 7 traduz as notas numa notação numérica. Esta notação numérica induz em outra notação numérica de base 12 que facilita várias operações.

A tabela seguinte apresenta um quadro onde se apresentam as combinações válidas da representação da notação musical, segundo o modo simbólico ou numérico em base 7 ou 12 e de forma absoluta ou por variação.

	Simbólica	Numérica em base 7	Numérica em base 12
Absoluta	Sim	Sim	Sim
Variação	Não	Sim	Sim

Tabela 4.1: Matriz da representação do Motivo Melódico

Na representação da frequência existe a noção de um tom e de meio-tom, havendo a noção de acidentes, no caso de bemóis e sustenidos. A representação de frequências é o produto entre um número inteiro, na versão absoluta e um número real, na versão de variação e um número real. O primeiro elemento representa a frequência em tons ou a sua variação. No entanto o segundo elemento, tendo por unidade o meio-tom representa a ocorrência de acidentes.

No conjunto das propriedades esta camada tem a noção de Clave, tanto o nome como a linha, da tonalidade e da oitava em que se encontra a sequência

de frequências.

Listagem 4.1: Motivo Melódico

```
1  -- | Melodic node
2  type Pitch = Number
3  type MelodicNode = Maybe (Delta, Accident)
4  type MelodicRelative = Maybe (Delta, Accident)
5  type MelodicAbsolute = Maybe (Pitch, Accident)
6  type MelodicClass = Maybe (PitchClass, Accident)
7  -- | Accident is defined as number /provisional/.
8  --  It doesn't support natural (only supports flats and sharps)
9  -- To supports sharps, flats and natural, it will be @Maybe Number@
10 -- This number is number of half-tones.
11 type Accident = Maybe Number
12 -- |
13 data AccidentClass = DoubleSharp
14                    | Sharp
15                    | Natural
16                    | Flat
17                    | DoubleFlat
18                    | UnknowAccident Text
19                   deriving (Eq, Show)
20 -- | Pitch Class definition
21 data PitchClass = C -- ^ C
22                 | D -- ^ D
23                 | E -- ^ E
24                 | F -- ^ F
25                 | G -- ^ G
26                 | A -- ^ A
27                 | B -- ^ B
28                deriving (Eq, Show, Read)
29 -- | MelodicNode with PitchClass
30 type MelodicClassNode = Maybe (PitchClass, Accident)
31 -- | default settings
32 settings :: Settings
33 settings = fromList [
34                ("ClefName",    text "ClefG" priority),
35                ("ClefNumber", number 2    priority),
36                ("Key",         number 0    priority),
37                ("Mode",        text "Major" priority),
38                ("Octave",      number 0    priority)]
```

4.1.3 Motivo Rítmico

O motivo rítmico consiste na representação de durações. Existe, tal como para o motivo melódico, várias formas da representação da notação musical. A sequência pode representada de forma absoluta usando os nomes das figuras ou o tempo referente à sua duração. Também se pode representar a variação da duração. Cada elemento é definido por um par no qual o primeiro elemento apresenta a duração absoluta ou a variação e segundo elemento do par apresenta o número de pontos de aumentação.

A matriz do motivo rítmico é semelhante à matriz do motivo melódico, apesar de ter menos um tipo de representação.

	Simbólica	Numérica
Absoluta	Sim	Sim
Variação	Não	Sim

Tabela 4.2: Matriz da representação do Motivo Melódico

No conjunto das propriedades desta camada encontram-se a noção de tempo e compasso.

Listagem 4.2: Motivo Rítmico

```
1  -- * Types
2  -- | Rhythm node
3  type RhythmNode = (Delta, Dots)
4  --type RhythmAbsolute = (Number, Dots)
5  --type RhythmRelative = (Delta, Dots)
6  type RhythmAbsolute = (RatioNumber, Dots)
7  type RhythmRelative = (RatioNumber, Dots)
8  -- | Dots is defined by number.
9  -- Only Integers and positive numbers are allowed.
10 type Dots = IntegerNumber
11 type Duration = Number
12 data DurationClass = Whole
13                    | Half
14                    | Quarter
15                    | Eighth
16                    | Th16
17                    | Th32
18                    | Th64
19                    | UnkownDuration Text
20     deriving (Eq, Show)
21 -- | sefault settings
22 settings :: Settings
23 settings = fromList [
24                 ("TempoPitch", number 4     priority),
25                 ("TempoNumber", number 60    priority),
26                 ("CompassUp", number 4     priority),
27                 ("CompassDown", number 4     priority)]
```

4.1.4 Motivo Zip

O motivo zip[1] representa a fusão entre o motivo melódico e o motivo rítmico. Desta forma, este motivo contém todas as propriedades do motivo melódico e do motivo rítmico.

A estrutura deste motivo é semelhante à estrutura dos motivos agregados, tendo uma secção para propriedades, na qual se encontram todas as propriedades de ambos os motivos. A sequência de elementos é representada por

[1] Atenção: não confundir a noção de zip nesta secção, que se refere ao agrupamento com a compressão zip.

um par entre o elemento do motivo melódico e o elemento do motivo rítmico.

4.1.5 Motivo Notacional, com Vozes, Instrumentos e Anotações

O motivo notacional corresponde a acrescentar notação musical ao motivo *zip*. Este motivo apresenta-se com a mesma estrutura, mas com alteração do elementos da sequência que tem presente a notação musical, além da notação musical base apresentada nos motivos melódicos e rítmico. O motivo notacional é dependente do motivo *zip*. Apesar de ser possível definir este motivo sem recurso ao motivo *zip*, torna-o sem informação relevante para o estudo musical. A notação musical só faz sentido quando aplicada a uma entidade musical, tal como a uma nota ou compasso.

É muito semelhante a definição de motivo com vozes e a definição de motivo com instrumentos. O motivo com vozes corresponde à agregação de vozes, às quais se adiciona um identificador. O mesmo acontece com o motivo com instrumentos, no qual se juntam vários instrumentos indexados por um identificador. Desta forma, é possível definir a estrutura musical sem perda da sua semântica. Os motivos com vozes e com instrumentos são a adição de perspectivas sobre a música. A obra musical é o conjunto das várias vozes e dos vários instrumentos que são as várias perspectivas sobre a obra musical.

O motivo com anotações agrega à notação musical a possibilidade de comentar a estrutura musical. Esta anotação preserva a estrutura de motivo. A necessidade da anotação musical deve-se a querer registar o estudo da análise musical na própria obra musical.

4.2 Estratificação de MusicXML

No presente capítulo é apresentada uma estratificação do formato MusicXML após se abordar a classificação do MusicXML perante a hierarquia de Chomsky. O formato MusicXML é estratificado em seis níveis, sendo o quinto nível equivalente ao próprio MusicXML. Os níveis são apresentados de forma decrescente, iniciando-se no nível 5. O nível 6 corresponde a uma extensão do MusicXML para anotações.

4.2.1 Nível 5

A análise efectuada neste nível apenas corresponde à versão `partwise` do MusicXML. Foi escolhida esta versão de representação da música por ser a

mais comum na representação de notação musical, em especial, em outros formatos para representação de notação musical, tal como o Lilypond e ABC. A análise com o foco nas várias partes musicais permite mais facilmente corresponder termos de cada nível a notações existentes. A análise da versão `timewise` em vários níveis é equivalente.

Este nível contém a implementação de um tipo de dados equivalente à especificação do MusicXML, mas interpretando a semântica de algumas entidades. Este nível pretende ser isomorfo à especificação do MusicXML. Devido a que a implementação feita no pacote MusicXML não faz a interpretação de algumas entidades, preservando-as na forma textual, é necessário que existam funções de interpretação e representação para tornar possível a sua manipulação com a semântica esperada.

Este nível será muito semelhante à implementação do pacote MusicXML de acordo com a especificação que usa *XML* Schemas.

Listagem 4.3: MusicXML - Nível 5

```
1  -- |
2  type Score_Partwise =
3      (MusicXML.Document_Attributes, (MusicXML.Score_Header, [Part]))
4  -- |
5  type Part = (MusicXML.ID, [Measure])
6  -- |
7  type Measure = ((MusicXML.CDATA, Maybe MusicXML.Yes_No,
8      Maybe MusicXML.Yes_No, Maybe MusicXML.Tenths), [Music_Data])
9  -- |
10 data Music_Data =
11       Music_Data_1 Note
12     | Music_Data_2 MusicXML.Backup
13     | Music_Data_3 MusicXML.Forward
14     | Music_Data_4 MusicXML.Direction
15     | Music_Data_5 Attributes
16     | Music_Data_6 MusicXML.Harmony
17     | Music_Data_7 MusicXML.Figured_Bass
18     | Music_Data_8 MusicXML.Print
19     | Music_Data_9 MusicXML.Sound
20     | Music_Data_10 MusicXML.Barline
21     | Music_Data_11 MusicXML.Grouping
22     | Music_Data_12 MusicXML.Link
23     | Music_Data_13 MusicXML.Bookmark
24       deriving (Eq, Show)
25 -- |
26 type Note =
27     ((MusicXML.Print_Style, MusicXML.Printout, Maybe MusicXML.CDATA,
28         Maybe MusicXML.CDATA, Maybe MusicXML.CDATA, Maybe MusicXML.CDATA,
29         Maybe MusicXML.CDATA, Maybe MusicXML.Yes_No),
30     (Note_, Maybe Instrument, Editorial_Voice,
31         Maybe Type, [Dot], Maybe Accidental,
32         Maybe Time_Modification, Maybe Stem,
33         Maybe Notehead, Maybe Staff, [Beam],
34         [Notations], [Lyric]))
35 -- |
36 data Note_ =
37           Note_1 (Grace, Full_Note, Maybe (Tie, Maybe Tie))
38         | Note_2 (Cue, Full_Note, Duration)
```

```
39          | Note_3 (Full_Note, Duration, Maybe (Tie, Maybe Tie))
40            deriving (Eq, Show)
41  -- |
42  type Grace = MusicXML.Grace
43  -- |
44  type Cue = MusicXML.Cue
45  -- |
46  type Tie = MusicXML.Tie
47  -- |
48  type Full_Note = (Maybe MusicXML.Chord, Full_Note_)
49  -- |
50  data Full_Note_ = Full_Note_1 Layer2.Pitch
51                  | Full_Note_2 Layer4.Unpitched
52                  | Full_Note_3 Layer4.Rest
53                    deriving (Eq, Show)
54  -- |
55  type Duration = IntegerNumber
56  -- |
57  type Editorial_Voice = MusicXML.Editorial_Voice
58  -- |
59  type Instrument = MusicXML.Instrument
60  -- |
61  type Type = (Maybe MusicXML.Symbol_Size, Layer1.Type_)
62  -- |
63  type Dot = MusicXML.Dot
64  -- |
65  type Accidental = ((Maybe MusicXML.Yes_No, Maybe MusicXML.Yes_No,
66      MusicXML.Level_Display, MusicXML.Print_Style), Layer1.Accidental_)
67  -- |
68  type Time_Modification = MusicXML.Time_Modification
69  -- |
70  type Stem = MusicXML.Stem
71  -- |
72  type Notehead = MusicXML.Notehead
73  -- |
74  type Beam = MusicXML.Beam
75  -- | positive number
76  type Staff = IntegerNumber
77  -- |
78  type Lyric = MusicXML.Lyric
79  -- |
80  type Notations = MusicXML.Notations
81  -- |
82  type Attributes = (Editorial, Maybe Divisions, [Key], [Time],
83      Maybe Staves, Maybe Part_Symbol, Maybe Instruments,
84      [Clef], [Staff_Details], Maybe Transpose, [Directive],
85      [Measure_Style])
86  -- |
87  type Editorial = MusicXML.Editorial
88  -- |
89  type Divisions = IntegerNumber
90  -- |
91  type Key = (
92      (Maybe MusicXML.CDATA, MusicXML.Print_Style, MusicXML.Print_Object),
93      (Key_, [Key_Octave]))
94  -- |
95  data Key_ = Key_1 (Maybe MusicXML.Cancel, Layer2.Fifths, Maybe Layer2.Mode)
96            | Key_2 [(Layer2.Key_Step, Layer2.Key_Alter)]
97              deriving (Eq, Show)
98  -- |
99  type Key_Octave = ((MusicXML.CDATA, Maybe MusicXML.Yes_No), Layer1.Octave)
100 -- |
```

```
101 type Time = ((Maybe MusicXML.CDATA, Maybe MusicXML.Time_A,
102     MusicXML.Print_Style, MusicXML.Print_Object), Layer3.Time_B)
103 -- |
104 type Staves = MusicXML.Staves
105 -- |
106 type Part_Symbol = MusicXML.Part_Symbol
107 -- |
108 type Instruments = MusicXML.Instruments
109 -- |
110 type Clef = (
111     (Maybe MusicXML.CDATA, Maybe MusicXML.Yes_No, Maybe MusicXML.Symbol_Size,
112      MusicXML.Print_Style, MusicXML.Print_Object),
113     (Layer2.Sign, Maybe Layer2.Line, Maybe Layer2.Clef_Octave_Change))
114 -- |
115 type Staff_Details = MusicXML.Staff_Details
116 -- |
117 type Transpose = MusicXML.Transpose
118 -- |
119 type Directive = MusicXML.Directive
120 -- |
121 type Measure_Style = MusicXML.Measure_Style
```

4.2.2 Nível 4

Este nível não contém a informação de *layout* em algumas entidades. Este nível é equivalente ao MusicXML com algumas perdas, como também é equivalente a outros formatos como Lilypond ou outros formatos de notação musical. Este nível na definição de música abstracta equivale aos níveis com notação, com vozes e com instrumentos.

O principal critério para obter este nível é o corte de informação especificada em alguns atributos do MusicXML. A selecção dos elementos em que a informação dos atributos é desprezada está relacionada com os próximos níveis.

Listagem 4.4: MusicXML - Nível 4

```
1  -- |
2  type Score_Partwise =
3      (MusicXML.Document_Attributes, (MusicXML.Score_Header, [Part]))
4  -- |
5  type Part = [Measure]
6  -- |
7  type Measure = [Music_Data]
8  -- |
9  data Music_Data =
10       Music_Data_1 Note
11     | Music_Data_2 MusicXML.Backup
12     | Music_Data_3 MusicXML.Forward
13     | Music_Data_4 MusicXML.Direction
14     | Music_Data_5 Attributes
15     | Music_Data_6 MusicXML.Harmony
16     | Music_Data_7 MusicXML.Figured_Bass
17     | Music_Data_8 MusicXML.Print
18     | Music_Data_9 MusicXML.Sound
```

```
19      | Music_Data_10 MusicXML.Barline
20      | Music_Data_11 MusicXML.Grouping
21      | Music_Data_12 MusicXML.Link
22      | Music_Data_13 MusicXML.Bookmark
23        deriving (Eq, Show)
24  -- |
25  type Note =
26      (Note_, Maybe Instrument, Editorial_Voice,
27          Maybe Type, [Dot], Maybe Accidental,
28          Maybe Time_Modification, Maybe Stem,
29          Maybe Notehead, Maybe Staff, [Beam],
30          [Notations], [Lyric])
31  -- |
32  data Note_ =
33            Note_1 (Grace, Full_Note, Maybe (Tie, Maybe Tie))
34          | Note_2 (Cue, Full_Note, Duration)
35          | Note_3 (Full_Note, Duration, Maybe (Tie, Maybe Tie))
36            deriving (Eq, Show)
37  -- |
38  type Grace = MusicXML.Grace
39  -- |
40  type Cue = MusicXML.Cue
41  -- |
42  type Tie = MusicXML.Tie
43  -- |
44  type Full_Note = (Maybe MusicXML.Chord, Full_Note_)
45  -- |
46  data Full_Note_ = Full_Note_1 Layer2.Pitch
47                  | Full_Note_2 Unpitched
48                  | Full_Note_3 Rest
49                    deriving (Eq, Show)
50  -- |
51  type Unpitched = MusicXML.Unpitched
52  -- |
53  type Rest = MusicXML.Rest
54  -- |
55  type Duration = IntegerNumber
56  -- |
57  type Editorial_Voice = MusicXML.Editorial_Voice
58  -- |
59  type Instrument = MusicXML.Instrument
60  -- |
61  type Type = Layer1.Type_
62  -- |
63  type Dot = MusicXML.Dot
64  -- |
65  type Accidental = Layer1.Accidental_
66  -- |
67  type Time_Modification = MusicXML.Time_Modification
68  -- |
69  type Stem = MusicXML.Stem
70  -- |
71  type Notehead = MusicXML.Notehead
72  -- |
73  type Beam = MusicXML.Beam
74  -- | positive number
75  type Staff = IntegerNumber
76  -- |
77  type Lyric = MusicXML.Lyric
78  -- |
79  type Notations = MusicXML.Notations
80  -- |
```

```
81  type Attributes = (Editorial, Maybe Divisions, [Key], [Time],
82      Maybe Staves, Maybe Part_Symbol, Maybe Instruments,
83      [Clef], [Staff_Details], Maybe Transpose, [Directive],
84      [Measure_Style])
85  -- |
86  type Editorial = MusicXML.Editorial
87  -- |
88  type Divisions = IntegerNumber
89  -- |
90  type Key = (Key_, [Layer2.Key_Octave])
91  -- |
92  data Key_ = Key_1 (Maybe MusicXML.Cancel, Layer2.Fifths, Maybe Layer2.Mode)
93            | Key_2 [(Layer2.Key_Step, Layer2.Key_Alter)]
94              deriving (Eq, Show)
95  -- |
96  type Time = Layer3.Time_B
97  -- |
98  type Staves = MusicXML.Staves
99  -- |
100 type Part_Symbol = MusicXML.Part_Symbol
101 -- |
102 type Instruments = MusicXML.Instruments
103 -- |
104 type Clef = (Layer2.Sign, Maybe Layer2.Line, Maybe Layer2.Clef_Octave_Change)
105
106 -- |
107 type Staff_Details = MusicXML.Staff_Details
108 -- |
109 type Transpose = MusicXML.Transpose
110 -- |
111 type Directive = MusicXML.Directive
112 -- |
113 type Measure_Style = MusicXML.Measure_Style
```

4.2.3 Nível 3

Esta é o nível mais simples que mantém a estrutura do MusicXML. Este nível é equivalente a uma representação simples do Lilypond, assim como à notação musical abc. Este nível, na definição da notação de música abstracta, é equivalente à não presença do nível com notação. Assim este nível é mais complexo, ou ter informação sobre os instrumentos ou vozes não presentes no motivo `Zip`, ao qual se encaixa melhor.

Neste nível descartou-se a informação sobre notação musical tal como ornamentos, informação técnica e outra. Um objectivo nesta definição foi encontrar um nível mais simples que ainda preserve a estrutura básica do MusicXML.

Listagem 4.5: MusicXML - Nível 3

```
1  -- |
2  type Score_Partwise =
3      (MusicXML.Document_Attributes, (MusicXML.Score_Header, [Part]))
4  -- |
5  type Part = [Measure]
```

```
6  -- |
7  type Measure = [Music_Data]
8  -- |
9  data Music_Data =
10       Music_Data_1 Note
11     | Music_Data_2 MusicXML.Backup
12     | Music_Data_3 MusicXML.Forward
13     | Music_Data_5 Attributes
14     | Music_Data_10 Barline
15       deriving (Eq, Show)
16 -- |
17 type Barline = MusicXML.Barline
18 -- |
19 type Note =
20     (Note_, Maybe Instrument, Editorial_Voice,
21         Maybe Type, [Dot], Maybe Accidental,
22         Maybe Staff)
23 -- |
24 data Note_ =
25           Note_3 (Full_Note, Duration)
26           deriving (Eq, Show)
27 -- |
28 type Full_Note = (Maybe MusicXML.Chord, Full_Note_)
29 -- |
30 data Full_Note_ = Full_Note_1 Layer2.Pitch
31                | Full_Note_3 Rest
32                  deriving (Eq, Show)
33 -- |
34 type Rest = ()
35 -- |
36 type Duration = IntegerNumber
37 -- |
38 type Editorial_Voice = MusicXML.Editorial_Voice
39 -- |
40 type Instrument = MusicXML.Instrument
41 -- |
42 type Type = Layer1.Type_
43 -- |
44 type Dot = MusicXML.Dot
45 -- |
46 type Accidental = Layer1.Accidental_
47 -- | positive number
48 type Staff = IntegerNumber
49 -- |
50 type Attributes = (Maybe Divisions, [Key], [Time],
51     Maybe Staves, Maybe Instruments,
52     [Clef], Maybe Transpose)
53 -- |
54 type Editorial = MusicXML.Editorial
55 -- |
56 type Divisions = IntegerNumber
57 -- |
58 type Key = (Key_, [Layer2.Key_Octave])
59 -- |
60 data Key_ = Key_1 (Maybe MusicXML.Cancel, Layer2.Fifths, Maybe Layer2.Mode)
61           | Key_2 [(Layer2.Key_Step, Layer2.Key_Alter)]
62             deriving (Eq, Show)
63 -- |
64 type Time = Time_B
65 -- |
66 data Time_B = Time_5 [(Beats, Beat_Type)]
67            | Time_6 MusicXML.Senza_Misura
```

```
68                deriving (Eq, Show)
69  -- | MusicXML Schema specify "xs:string"
70  type Beats = (IntegerNumber, Maybe IntegerNumber)
71  -- | MusicXML Schema specify "xs:string"
72  type Beat_Type = IntegerNumber
73  -- |
74  type Staves = MusicXML.Staves
75  -- |
76  type Part_Symbol = MusicXML.Part_Symbol
77  -- |
78  type Instruments = MusicXML.Instruments
79  -- |
80  type Clef = Layer2.Clef
81  -- |
82  type Staff_Details = MusicXML.Staff_Details
83  -- |
84  type Transpose = MusicXML.Transpose
85  -- |
86  type Directive = MusicXML.Directive
87  -- |
88  type Measure_Style = MusicXML.Measure_Style
```

4.2.4 Nível 2

Este nível apenas permite música para uma parte. Esta redução afecta a estrutura da notação musical. Desta forma este nível é equivalente a subconjuntos de notação musical tal como Lilypond ou ABC. Este nível além de toda a informação presente no primeiro nível, ainda tem a noção de compasso e a indicação da tonalidade e de compasso da partitura.

Listagem 4.6: MusicXML - Nível 2

```
1   -- |
2   type Score_Partwise = [Measure]
3   -- |
4   type Measure = [Music_Data]
5   -- |
6   data Music_Data =
7         Music_Data_1 Note
8       | Music_Data_5 Attributes
9         deriving (Eq, Show)
10  -- |
11  type Note = (Note_, Maybe Type, [Dot], Maybe Accidental)
12  -- |
13  data Note_ =
14            Note_3 (Full_Note, Duration)
15            deriving (Eq, Show)
16  -- |
17  type Full_Note = Full_Note_
18  -- |
19  data Full_Note_ = Full_Note_1 Pitch
20                  | Full_Note_3 Layer1.Rest
21                    deriving (Eq, Show)
22  -- |
23  type Pitch = (Layer1.Step, Maybe Layer1.Alter, Layer1.Octave)
24  -- |
```

```
25 type Duration = IntegerNumber
26 -- |
27 type Type = Layer1.Type_
28 -- |
29 type Dot = ()
30 -- |
31 type Accidental = Layer1.Accidental_
32 -- |
33 type Attributes = (Maybe Divisions, [Key], [Time], [Clef])
34 -- |
35 type Divisions = IntegerNumber
36 -- |
37 type Key = (Key_, [Key_Octave])
38 -- |
39 data Key_ = Key_1 (Fifths, Maybe Mode)
40           | Key_2 [(Key_Step, Key_Alter)]
41             deriving (Eq, Show)
42 -- |
43 type Fifths = IntegerNumber
44 -- |
45 data Mode = Major | Minor |
46     Dorian | Phrygian | Lydian | Mixolydian |
47     Aeolian | Ionian | Locrian
48     deriving (Eq, Show)
49 -- |
50 type Key_Step = Layer1.Step
51 -- |
52 type Key_Alter = Layer1.Alter
53 -- |
54 type Key_Octave = Layer1.Octave
55 -- |
56 type Time = Time_B
57 -- |
58 data Time_B = Time_5 [(Beats, Beat_Type)]
59               deriving (Eq, Show)
60 -- | MusicXML Schema specify "xs:string"
61 type Beats = (IntegerNumber, Maybe IntegerNumber)
62 -- | MusicXML Schema specify "xs:string"
63 type Beat_Type = IntegerNumber
64 -- |
65 type Clef = (Sign, Maybe Line, Maybe Clef_Octave_Change)
66 -- |
67 data Sign =
68     Clef_Sign_G | Clef_Sign_F | Clef_Sign_C |
69     Clef_Sign_Percussion | Clef_Sign_TAB |
70     Clef_Sign_None
71     deriving (Eq, Show, Enum)
72 -- |
73 type Line = IntegerNumber
74 -- |
75 type Clef_Octave_Change = IntegerNumber
```

4.2.5 Nível 1

Este nível define uma obra musical como sendo uma sequência de notas musicais. Não existe informação sobre a clave, o tempo nem sobre a divisão em compassos. A definição de nota musical quase se limita ao par entre a

frequência e a duração. Neste nível já não existe informação sobre a estruturação da notação musical.

Listagem 4.7: MusicXML - Nível 1

```
1  -- |
2  type Score_Partwise = [Music_Data]
3  -- |
4  data Music_Data =
5       Music_Data_1 Note
6       deriving (Eq, Show)
7  -- |
8  type Note = (Note_, Maybe Type, [Dot], Maybe Accidental)
9  -- |
10 data Note_ =
11           Note_3 Full_Note
12           deriving (Eq, Show)
13 -- |
14 type Full_Note = Full_Note_
15 -- |
16 data Full_Note_ = Full_Note_1 Pitch
17                 | Full_Note_3 Rest
18                   deriving (Eq, Show)
19 -- |
20 type Pitch = (Step, Maybe Alter, Octave)
21 -- |
22 data Step = C | D | E | F | G | A | B
23     deriving (Eq, Show, Ord, Enum)
24 -- |
25 type Alter = Number
26 -- |
27 type Octave = IntegerNumber
28 -- |
29 type Rest = ()
30 -- |
31 type Type = Type_
32 -- |
33 data Type_ = Long | Breve |
34     Whole | Half | Quarter | Eighth |
35     Th16 | Th32 | Th64 | Th128 | Th256
36     deriving (Eq, Show, Ord, Enum)
37 -- |
38 type Dot = ()
39 -- |
40 type Accidental = Accidental_
41 -- |
42 data Accidental_ =
43     Sharp | Natural | Flat |
44     Double_Sharp | Sharp_Sharp | Flat_Flat |
45     Natural_Sharp | Natural_Flat |
46     Quarter_Sharp | Quarter_Flat |
47     Three_Quarters_Sharp | Three_Quarters_Flat
48     deriving (Eq, Show, Ord, Enum)
```

4.2.6 Nível 6

Este nível é equivalente ao MusicXML com a interpretação da entidade `grouping` para anotação após a análise musical[29]. A anotação musical é uma anota-

ção com informação útil sobre a análise musical efectuada. Usa-se a entidade `grouping`, que está destinada a ser utilizada na análise musical.

A anotação é feita a um intervalo ou a um elemento, sendo possível o aninhamento de anotações. Também é possível agrupar várias anotações. Recomenda-se que o atributo `member-of` do elemento `grouping` seja `annotation`, para desambiguar outras possíveis utilizações destes elementos.

De seguida apresentamos a implementação do sexto nível.

Listagem 4.8: MusicXML - Nível 6

```
1  -- |
2  type Score_Partwise =
3      (MusicXML.Document_Attributes, (MusicXML.Score_Header, [Part]))
4  -- |
5  type Part = (MusicXML.ID, [Measure])
6  -- |
7  type Measure = ((MusicXML.CDATA, Maybe MusicXML.Yes_No,
8      Maybe MusicXML.Yes_No, Maybe MusicXML.Tenths), [Music_Data])
9  -- |
10 data Music_Data =
11       Music_Data_1 Note
12     | Music_Data_2 MusicXML.Backup
13     | Music_Data_3 MusicXML.Forward
14     | Music_Data_4 MusicXML.Direction
15     | Music_Data_5 Attributes
16     | Music_Data_6 MusicXML.Harmony
17     | Music_Data_7 MusicXML.Figured_Bass
18     | Music_Data_8 MusicXML.Print
19     | Music_Data_9 MusicXML.Sound
20     | Music_Data_10 MusicXML.Barline
21     | Music_Data_11 MusicXML.Grouping
22     | Music_Data_12 MusicXML.Link
23     | Music_Data_13 MusicXML.Bookmark
24     | Music_Data_14 Annotation
25       deriving (Eq, Show)
26 -- |
27 type Annotation = (Maybe MusicXML.Start_Stop, MusicXML.PCDATA)
28 -- |
29 type Note = Layer5.Note
30 -- |
31 type Attributes = Layer5.Attributes
```

4.3 MusicCount

Esta aplicação é análogo ao `wc`, mas destinado à notação musical. Neste momento aceita a notação musical apenas no formato MusicXML e conta o número de partes, compassos e notas. Recebe como parâmetros *flags* que indicam se efectua a contagem de partes, com a presença de `-p` ou `-part`, a contagem de compasso, com a presença de `-m` ou `-measure` e a contagem de notas, com a presença de `-n` ou `-note`.

Para a contagem de partes, compassos e notas é possível executar das seguintes formas:

Listagem 4.9: Exemplos de utilização do MusicCount

```
1 MusicCount -pmn file.xml
2 MusicCount --part --measure --note file.xml
```

4.4 MusicTranslate

Esta aplicação destina-se a converter documentos com notação musical entre diferentes formatos. Recebe como parâmetros obrigatórios uma opção `-i` ou `-input` e uma opção `-o` ou `-output` que tem como argumento o formato dos ficheiros com notação musical.

Apresentam-se de seguida alguns exemplos de utilização da aplicação:

Listagem 4.10: Exemplos de utilização do MusicTranslate

```
1 MusicTranslate -i musicxml -o abc file.xml
2 MusicTranslate -o abc -i musicxml file.xml
```

Neste momento apenas estão disponíveis as traduções entre os seguintes formatos:

input x output	abc	abstract	haskore	lilypond	musicxml
abc	-	Não	Não	Não	Não
abstract	Sim	-	Não	Não	Não
haskore	Não	Não	-	Não	Não
lilypond	Não	Não	Não	-	Não
musicxml	Sim	Sim	Sim	Não	-

Tabela 4.3: Matriz das traduções entre formatos de notação musical

4.5 HaMusic

Existe uma ferramenta com o mesmo nome do pacote que se processa as instruções presentes num documento *XML*. Serão apresentadas as operações disponíveis e a especificação de documentos *XML*.

4.5.1 Operações

São apresentadas várias operações sobre documentos MusicXML. Entre as opções destacam-se a filtragem segundo condições específicas, a contagem de elementos presentes além de transformações para outros formatos ainda que em formato muito rudimentar. Estas operações encontram-se definidas de forma a ser possível a sua utilização como uma API. As operações apresentadas ajudam na análise musical de documentos MusicXML, o único que a biblioteca *hamusic* consegue importar.

A filtragem de entidades musicais permite excluir informação que se ache irrelevante para se efectuar posteriormente o estudo da análise musical. A existência de operações sobre os vários níveis do MusicXML permitem fazer uma análise de conteúdos que se achem irrelevantes. Este processo de simplificação da notação musical facilita a análise e ajuda a classificação das entidades musical.

A contagem de entidades musicais e a apresentação de algumas estatísticas ajuda a análise sobre uma perspectiva de frequências. Esta análise, por frequências ajuda a análise musical na detecção de padrões musicais.

A transformação de notação musical do formato MusicXML para o formato Haskore ou até para o formato MIDI são apresentados como uma forma de tornar visível e prática a análise musical.

4.5.2 Linguagem

Foi especificada uma linguagem de domínio específico correspondente às operações disponibilizadas, que pode ser integrada num ambiente de programação *Haskell* e um subconjunto da linguagem *XML* que corresponde à definição de uma linguagem para as operações.

De seguida apresenta-se uma lista de vantagens da utilização de um subconjunto do *XML* como a linguagem em vez da tradicional especificação de uma linguagem de domínio específico.

- A sintaxe base já está definida. Desta forma, não é necessário escrever um parser de raiz para a linguagem.
- Na sintaxe específica usa-se a anotação permitida pela tecnologia *XML*. Simplifica a compreensão da linguagem, tornando-a fácil de escrever.
- A possibilidade de após a especificação da linguagem se obter o código do parser. Neste caso específico, com a ajuda da biblioteca `HaXml` e da ferramenta `DTDtoHaskell`, apenas é necessário especificar a linguagem de anotação usando um DTD(Document Type Definition).

- Existe um controlo total no reconhecimento das expressões. É necessário escrever um parser para reconhecer a anotação.

Na definição da linguagem é utilizada a biblioteca `HaXml` com as ferramentas `Dtd2Haskell` e `Validate` que agilizam o processo. Com as referidas ferramentas após a especificação da linguagem usando um DTD obtém-se um módulo em *Haskell* que efectua a leitura e escrita de documentos em *XML*.

Especificação da linguagem A linguagem de comunicação apresenta-se como uma lista de acções/comandos a serem processados. Esta lista de comandos engloba acções que podem ser feitas *inline* na linguagem *Haskell*. No entanto, esta forma facilita a sua utilização.

Listagem 4.11: Especificação em DTD

```
<!DOCTYPE script SYSTEM "Script.dtd" [

    <!--
    This is my DTD to Abstract Music Combinators/Commands
    -->

    <!ELEMENT script (action*)>
    <!ATTLIST script
        author CDATA #IMPLIED
        date CDATA #IMPLIED
        description CDATA #IMPLIED>

    <!ELEMENT action ((filter | reification | stat |
        parttime | timepart | haskore | midi)*)>
    <!ATTLIST action
        input CDATA #REQUIRED
        output CDATA #IMPLIED
        warnings (yes | no) #IMPLIED>

    <!-- use stat instead -->
    <!--
    <!ELEMENT length EMPTY>
    <!ATTLIST length
        select (part | measure ) #REQUIRED
        mode (simple | map | concat) #IMPLIED>
        -->

    <!ELEMENT filter EMPTY>
    <!ATTLIST filter
        select (note | note-grace | note-cue | note-normal) #REQUIRED
        mode (yes | no) #IMPLIED>

    <!ELEMENT reification EMPTY>
    <!ATTLIST reification
        value (1 | 2 | 3 | 4 | 5) #REQUIRED>

    <!ELEMENT stat (count*)>
    <!ATTLIST stat
        verbose (yes | no) #IMPLIED>

    <!ELEMENT count EMPTY>
```

```
42      <!ATTLIST count
43          select (part | measure | music-data | note |
44                  note-grace | note-cue | note-normal) #REQUIRED>
45
46      <!ELEMENT parttime EMPTY>
47      <!ELEMENT timepart EMPTY>
48      <!ELEMENT haskore EMPTY>
49
50      <!ELEMENT midi EMPTY>
51      <!ATTLIST midi
52          play (yes | no) #IMPLIED>
53  ]>
```

O Programa O programa, com o nome de *HaMusic*, lê um conjunto de instruções de um ficheiro *XML* e processa-as. O seu processamento são instruções de transformação/computação. Esta aplicação tem por objectivo condensar as aplicações que são descritas nas próximas secções. Assim, a aplicação *HaMusic* equivale ao conjunto somado das ferramentas `MusicCount`, `MusicTranslate` e outras operações não implementadas separadamente.

4.6 MusicGrep

Esta secção apresenta um estudo para a implementação do *MusicGrep*. O *MusicGrep* deverá ter uma interface equivalente ao comando `grep`.

4.6.1 Representação da Música: FIM DO CAPITULO

Foi prototipada uma representação da música muito simples, que será apresentada. No entanto, seria muito útil que a representação da música fosse um formato musical existente, como por exemplo o formato MusicXML.

4.6.2 Expressões Regulares Musicais

Também no estudo da arquitectura para o desenvolvimento desta ferramenta, propõe-se que se use um motor de expressões regulares existente. Dado que os motores de expressões regulares existentes trabalham sobre textos, é necessária uma função de pré-processamento que transforme o formato musical para um formato textual equivalente, e uma função de pós-processamento que efectue a operação inversa.

De seguida apresenta-se um diagrama com as simplificações possíveis que permitem escrever uma expressão regular musical.

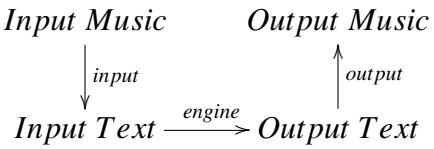


Figura 4.3: Arquitectura do MusicGrep (vista interna)

A existência de modificadores ajuda na expressividade das expressões regulares musicais, dado que os modificadores apresentados, `/d /D /p /P` originam subconjuntos da expressão regular original.

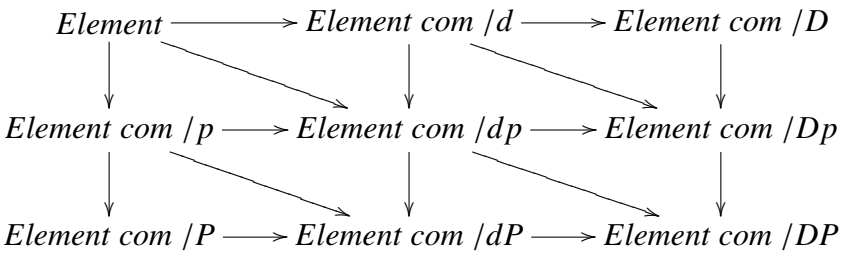


Figura 4.4: Element's translations

Uma expressão regular musical sem os modificadores mencionados refere-se a uma definição absoluta da notação musical quanto ao tempo e à tonalidade. Quando aplicado o modificador `/d` pretende-se ignorar a definição absoluta da duração, transformando-a numa definição relativa. A notação musical obtida através de pattern-matching de uma expressão regular com o modificador `/d` permite a identificação com idênticos intervalos de duração, não sendo exactamente iguais o que acontecia apenas no resultado da aplicação da expressão regular musical.

A aplicação do modificador `/p` tem resultado equivalente, pelo que permite identificar não só tonalidades iguais, mas também tonalidades relativas.

A noção de relatividade na tonalidade ou na duração não é igual. Enquanto que para obter a relatividade sobre a tonalidade se utiliza o binómio adição/subtracção, para a duração utiliza-se o binómio multiplicação/divisão. Na definição de relatividade da tonalidade existem duas possibilidades amplamente aceites, sendo a representação de ambas de forma numérica. A primeira é a divisão de uma oitava em sete notas musicais, e a segunda é a divisão da oitava em 12 meios-tons.

Nota-se que a aplicação da expressão regular musical sem modificadores identifica-se com um subconjunto da aplicação regular musical com os modificadores `/d` e `/p`. No primeiro caso, na aplicação do modificador `/d` permite várias durações que sejam equivalentes. No segundo caso, na aplicação do modificador `/p` permite várias tonalidades que sejam equivalentes.

Deve ter igual resultado a aplicação do modificador `/d` seguido do modificador `/p` à aplicação do modificador `/p` seguido do modificador `/d`. Desta forma, diz-se que o diagrama comuta na aplicação destes dois modificadores, havendo comutatividade na sua aplicação.

Os modificadores `/D` e `/P` indicam que se deve ignorar toda a informação relativa à duração e à tonalidade, respectivamente. Também estes modificadores são comutativos.

O âmbito da expressão regular musical apresentada apenas se cinge à informação sobre a tonalidade, duração e a existência ou não de pausas.

4.7 Sumário

Neste capítulo apresentam-se apresentada uma definição formal da notação musical em várias camadas. Também se apresentou uma estratificação do formato MusicXML comparando cada nível com a definição formal da notação musical apresentada previamente e com outros formatos existentes. Uma linguagem que permite a automatização de procedimentos de análise musical é apresentada na forma embutida e com recurso a um DTD que facilita a interoperabilidade entre procedimentos. São apresentadas ferramentas tais como um contador de entidades musicais e um tradutor de notação musical que são um subconjunto da aplicação *HaMusic*. Também é apresentada uma definição de expressão regular musical e a arquitectura para o motor de expressões regulares musicais. Na expressão musical apenas se considera uma notação musical muito simples.

No próximo capítulo apresentam-se as conclusões do trabalho desenvolvido assim como linhas de investigação.

Capítulo 5

Conclusão e Trabalho Futuro

Neste capítulo são apresentadas as conclusões deste documento e o que se prevê para trabalho futuro.

5.1 Conclusão

musicxml A biblioteca musicxml foi desenvolvida correspondendo à especificação do formato MusicXML. A tradução de DTD para Haskell foi metódica e sistematizada, sendo possível identificar padrões. Com este método de tradução são evitados os erros de tradução devido, também, à forte tipagem existente em Haskell. Para aumentar a segurança da tradução, foram executados vários casos de teste com ficheiros com partituras completas. Os resultados da execução de testes apresentam bons resultados em termos de performance da utilização da memória e tempo utilizado.

hamusic A biblioteca hamusic oferece algumas operações e estudo sobre música. A biblioteca apresenta uma definição de música abstracta, que se propõe a existência de vários níveis. Esta biblioteca apresenta um estudo inovador sobre a estratificação do MusicXML. É apresentada uma definição do MusicXML em seis níveis. São apresentadas ferramentas em estado experimental facilitando a análise musical. Uma das ferramentas permite a contagem de elementos musicais. A tradução entre vários formatos é apresentada de forma parcial, não cobrindo a totalidade dos formatos propostos, mas obtendo resultados satisfatórios.

repositório Durante o desenvolvimento foram coleccionados documentos MusicXML contendo partituras, algumas completas, que foram usados nal-

guns testes. Também foi criado um repositório de bibliografia, contendo os textos e as referências bibliográficas de forma estruturada.

5.2 Trabalho Futuro

Esta dissertação apresenta várias possibilidades para trabalho futuro, que algumas são apresentadas de seguida. Estas são possíveis orientações que se perspectivam no propósito de criar um ambiente em Haskell para a manipulação de música.

MusicXML O pacote *musicxml* que efectua a importação/exportação de documentos *XML* para o tipo de dados MusicXML definido em *Haskell* está conforme a especificação dos DTDs. No entanto, no momento da escrita deste documento foi apresentada uma versão usando *XML* Schemas do formato MusicXML pela Recordare. Esta redefinição foi feita, enquanto estava sendo discutida pela comunidade do MusicXML. A manutenção do pacote musicxml actualizado, conforme as especificações propostas, requer que se implementem as novas restrições nos tipos de dados e nas funções. Estas restrições além de ser necessária a implementação de novos tipos de dados e funções, é possível manter a mesma estrutura geral do pacote. É útil implementar o suporte a documentos em UTF-8 e UTF-16, além de permitir ler e escrever documentos comprimidos. Actualmente, apenas consegue ler o conteúdos dos documentos comprimidos.

Estratificação do formato MusicXML A estratificação do formato MusicXML está dependente da biblioteca de manipulação de MusicXML. Esta dependência é prejudicial quando a estrutura de dados do formato MusicXML for alterada, deve ser alterada pelo menos o nível 5. A definição dos vários níveis de MusicXML requer ainda estudo para se tornar clara a definição de cada nível e propriedades inerentes. Prevê-se que a estratificação seja sujeita a debate para se definir um padrão para a estratificação.

Música Abstracta A definição de música abstracta requer ainda um estudo aturado de operadores e funções algumas propriedades algébricas, para se obter uma definição clara dos níveis de definição e das propriedades associadas. A implementação de operações similares às existentes no Haskore é necessária numa primeira instância para manipular música. Operações como a transposição de uma pauta ou a contagem de elementos musicais fazem sempre sentido. Numa segunda instância, a implementação de uma análise

musical automática usando os princípios GTTM é necessária para a fazer análise a um repositório musical.

MusicCount O `MusicCount` apresenta-se ainda numa fase rudimentar, não focando a eficiência. Podem ser adicionadas mais entidades para serem indexadas na execução, além das partes, compassos e notas. Também podem ser adicionados mais formatos para análise, tal como o formato ABC ou Lilypond além do MusicXML.

MusicTranslate O `MusicTranslate` apresenta-se ainda numa fase rudimentar. A tradução entre vários formatos é incompleta, existindo ainda muitas traduções não implementadas. Nesta fase apenas se encontram, principalmente traduções de exportação de MusicXML para outros formatos. É de grande importância a importação de outros formatos para o MusicXML.

MusicAnnotate No contexto da notação musical a anotação de notação musical é de relevante importância. Deste modo uma aplicação que permita fazer anotações a vários formatos de notação musical vira a ser útil na análise musical. Deve haver uma interface em que seja fácil a visualização de anotação assim como a sua edição.

MusicGrep Nesta dissertação são lançadas as bases para a implementação de Expressões Regulares Musicais. A implementação requer estudo na definição da própria linguagem musical, possibilitando a pesquisa de elementos musicais. Deve haver um ferramenta que permita a comparação de notação musical com base num pesquisador de expressões regulares musicais.

DSL - Script xml Deve ser revista a definição da script que permite a manipulação de música, tornando-a mais expressiva e completa. A definição em DTD deve acompanhar as modificações que existam ao nível da DSL ou até das funções disponibilizadas.

GTTM Deve ser implementado um sistema automático para análise musical que utilize as regras propostas em Generative Theory of Tonal Music. Para este sistema deve haver uma interface amigável que permita visualizar os resultados, de modo que seja de fácil compreensão. Do mesmo modo, é útil a implementação de uma análise automática usando a Teoria Schenkeriana.

Bibliografia

[1] R. Cohn. The autonomy of motives in schenkerian accounts of tonal music. *Music Theory Spectrum*, 14(2):150–170, 1992.

[2] A. Forte. Schenker's conception of musical structure, 1959.

[3] A. Gill and C. Runciman. Haskell program coverage. *ACM*, 2007.

[4] G. Gonzato. Making music with ABC plus. 2007.

[5] M. Good. MusicXML: An internet-friendly format for sheet music. 2001.

[6] M. Good. MusicXML. Webpage, Recordare, 2008. http://www.musicxml.com.

[7] M. Good and G. Actor. Using MusicXML for file interchange. *IEEE*, 2003.

[8] E. Gregorio. The abc class. March 2006.

[9] Gutenberg. Gutenberg project. Webpage, 2008. http://www.gutenberg.org.

[10] Hackage. Hackage. Webpage, 2008. http://hackage.haskell.org.

[11] M. Hamanaka, K. Hirata, and S. Tojo. Automatic generation of grouping structure based on the gttm. In *ICMC 2004*, 2004.

[12] M. Hamanaka, K. Hirata, and S. Tojo. Atta: Automatic time-span tree analyzer based on extended gttm. In *ISMIR 2005*, 2005.

[13] M. Hamanaka, K. Hirata, and S. Tojo. Automatic generation of metrical structure based on gttm. In *ICMC 2005*, 2005.

[14] M. Hamanaka, K. Hirata, and S. Tojo. Melody expectation method based on gttm and tps. In *ISMIR 2008*, September 2008.

[15] M. Hamanaka, K. Hirata, and S. Tojo. Melody morphing method based on gttm. In *ICMC 2008*, 2008.

[16] Haskell. Ghc. Webpage, 2008. http://www.haskell.org/ghc.

[17] Haskell. Haskell. Webpage, 2008. http://www.haskell.org/hugs.

[18] P. Haudak, J. Hudges, S. P. Jones, and P. Wadler. A history of haskell: Being lazy with class. *ACM*, 2007.

[19] W. B. Hewlett and E. S. Field. Directory of computer assisted research in musicology. 1986.

[20] W. B. Hewlett and E. S. Field. Directory of computer assisted research in musicology. 1987.

[21] R. Hinze and J. Jeuring. Generic haskell: applications. 2002.

[22] H. H. Hoos and K. A. Hamel. The guido music notation format version 1.0 specification part 1: Basic guido. Technical report, Technische Universität Darmstadt, Fachbereich Informatik Technische Universität Darmstadt, 1997.

[23] H. H. Hoos, K. A. Hamel, K. Renz, and J. Kilian. The guido notation format a novel approach for adequately representing score-level music.

[24] H. H. Hoos, J. Kilian, K. Renz, and T. Helbich. Salieri: A general, interactive computer music system.

[25] P. Hudak. Haskore music tutorial. Technical report, Yale University, 2000.

[26] P. Hudak. *Haskore Music Tutorial*, 2005.

[27] H. D. III. *Yet Another Haskell Tutorial*, 2006.

[28] S. P. Jones. *Haskell 98 Language and Libraries: The Revised Report*, 2002.

[29] K. Kaji and K. Nagao. Mixa: A musical annotation system. 2004.

[30] G. R. Laboratory. *LibMusicXML Reference Manual*. Grame Research Laboratory, March 2004.

[31] F. Lerdahl. *Tonal Pitch Space*. Oxford University Press, USA, 2001.

[32] F. Lerdahl and R. Jackendoff. Toward a formal theory of tonal music. *Journal of Music Theory*, 21(1):111–171, 1977.

[33] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[34] D. Lewin. A formal theory of generalized tonal functions. *Journal of Music Theory*, 26(1):23–60, 1982.

[35] A. Löh. *Typesetting Haskell and more with lhs2TeX*, 2004.

[36] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, December 2005.

[37] M. Music. Finale. Webpage, Finale Music, 2008. http://finalemusic.com/finale.

[38] Myriad. Myriad music plugin. Webpage, 2008. http://www.myriad-online.com/en/products/mmplugin.htm.

[39] H. Nienhuys, J. Nieuwenhuizen, et al. Gnu lilypond: The music typesetter. *Free Software Foundation*, 2004.

[40] H. Nienhuys, J. Nieuwenhuizen, et al. Lilypond: Program usage. *Free Software Foundation*, 2007.

[41] M. Project. Mutopia project. Webpage, 2008. http://www.mutopiaproject.org.

[42] K. Renz. Algorithms and data structures for a music notation system based on guido music notation. Master's thesis.

[43] Sibelius. Scorch. Webpage, 2008. http://sibelius.com/scorch.

[44] Sibelius. Sibelius. Webpage, 2008. http://sibelius.com.

[45] S. Silva. Repositório do MusicXML e HaMusic. Webpage, Universidade do Minho, 2008. https://troglodita.di.uminho.pt/svn/musica.

[46] S. Sinclair, M. Droettboom, and I. Fujinaga. Lilypond for pyscore: Approaching a universal translator for music notation. 2006.

[47] P. THIEMANN. A typed representation for html and xml documents in haskell. pages 435–468, 2002.

[48] A. van Deursen, P. Klint, and J. Visser. Domain-specific languages. *ACM SIGPLAN Notices*, 2000.

[49] B. L. Vercoe. Csounds. Webpage, M.I.T. Media Laboratory, 2008. http://www.csounds.com.

[50] W3C. Dom. Webpage, 2008. http://www.w3c.org/DOM.

[51] W3C. W3c. Webpage, 2008. http://www.w3c.org.

[52] W3C. Xml. Webpage, 2008. http://www.w3c.org/XML.

[53] W3C. Xml schema. Webpage, 2008. http://www.w3c.org/XML/Schema.

[54] M. Wallace and C. Runciman. Haskell and xml: Generic combinators or type-based translation? *ACM*, 1999.

[55] M. Wallace and C. Runciman. Haskell and xml: Generic combinators or type-based translation? *ACM*, 1999.

[56] Wikifonia. Wikifonia. Webpage, 2008. http://www.wikifonia.org.

[57] D. B. Williams. MusicXML: The new link for sharing sibelius and finale files. 2008.

Apêndice A

Estrutura do repositório

O repositório[45] apresentado inclui documentos de desenvolvimento além de documentação. Na secção sobre Material para divulgação é feita uma descrição da informação mais relevante em todo o repositório além de uma descrição geral.

A.1 Software

São apresentas duas bibliotecas elaboradas com o propósito de apresentar um estudo sobre o MusicXML. A primeira biblioteca apresentada é o pacote musicxml que permite a leitura e escrita de ficheiros em MusicXML. A segunda biblioteca permite, ainda que de forma rudimentar, a análise musical através de algumas operações de análise, tal como estatísticas.

Também foi necessária a implementação de pequenas ferramentas utilitárias. No início do desenvolvimento foi criada uma `script` em Perl, com o propósito de ser mais fácil de testar. Também foi implementada uma `script` que transformava algumas anotações, para o mesmo documento ser usado em `Literate Programming` e documentação HTML. As ferramentas usam a mesma notação causando um conflito. Também foram criadas `scripts` para simplificar e automatizar a manipulação de bibliografia.

musicxml Este pacote de software encontra-se no caminho `/work/MusicXML`. Esta biblioteca tem uma arquitectura tradicional com as pastas `src`, `doc`, `examples`, `tests` e `script`, que tem, respectivamente, os ficheiros de código em *Haskell*, documentação usando `lhs2TeX`, os ficheiros referentes aos três casos de estudo apresentados, os ficheiros de execução dos testes em *Haskell* e ficheiros de manutenção da biblioteca, tal como o `Makefile` que se encontra na pasta `script`.

hamusic Este pacote de software encontra-se no caminho `/work/HaMusic`. Esta biblioteca tem uma arquitectura tradicional com as pastas `src`, `doc`, `examples`, `tests` e `script`, que tem, respectivamente, os ficheiros de código em *Haskell*, documentação usando `lhs2TeX`, os ficheiros referentes aos três casos de estudo apresentados, os ficheiros de execução dos testes em *Haskell* e ficheiros de manutenção da biblioteca, tal como o Makefile que se encontra na pasta `script`.

HaskellPP Este pré-processador cria um ficheiro de extensão `.lt` após o processamento das instruções, que tornavam possível um ficheiro com anotações para ser processado pelo `haddock` seja processado também pelo `lhs2TeX`. Deste modo a ferramenta `lhs2TeX` transforma os ficheiros com extensão `.lt` em ficheiros com extensão `.tex`.

A necessidade deste processador deve-se a que existem anotações que são usadas por ambos os processadores, `haddock` e `lhs2TeX`, com diferente semântica. Este caso deve-se a que o `hadock` geralmente é utilizado apenas para ficheiros em *Haskell*, não `literate programming`, sendo com a extensão `.hs`, e a que a utilização do `lhs2TeX` apenas tem sentido com `literate haskell`, isto é, com a extensão `.lhs`.

BibTeX A bibliografia foi armazenada de forma estruturada no caminho `/references` e encontram-se ficheiros de bibliografia para cada documento.

Desta forma foram criadas algumas ferramentas no auxílio da criação/manutenção de bibliografia em formato `bibtex`. Para facilitar a automatização, as referências bibliográficas têm uma chave similar ao nome de ficheiro, e a indicação de um ficheiro através de uma chave, em que o seu valor é o link para o documento referenciado.

Estre as `scripts` implementadas, com recurso à linguagem Perl, entre as quais, `scripts` que juntam a bibliografia num único ficheiro, com o nome `joinbibtex.pl`, que separam a bibliografia em vários ficheiros, efectuando a operação inversa à `script` enunciada anterior, tendo o nome de `splitbibtex.pl`. Também existem `scripts` para criar referências bibliográficas em falta, preenchendo o mínimo de informação, com o nome de `mkbibtex.pl`, além de `scripts` de transformação de referências bibliográficas, que transformam a chave link em pdf, chamando-se `bibtex_link2pdf.pl`, que criam a chave pdf em referências que não têm, com o nome `bibtex_mkpdf.pl`, que transforma a chave pdf em pdfs, chamando-se `bibtex_pdf2pdfs.pl`, e a `script` `bibtex_pdfbars.pl` que faz uma correcção das barras presentes no valor da chave que se refere à localização do documento.

A.2 Documentação

Para o pacote MusicXML e para o pacote *HaMusic* foi criada documentação, usando o `lhs2TeX`, de forma a auxiliar futuros desenvolvimentos. A documentação tem a particularidade de ter a totalidade de código com alguns comentários, descrevendo as decisões tomadas. A documentação do pacote MusicXML é mais extensiva contendo a documentação presente na especificação do formato MusicXML.

A.3 Material para divulgação

O material sujeito a divulgação é apresentado com o recurso à sua localização, tendo por / a totalidade do repositório. Durante o desenvolvimento das bibliotecas apresentadas foi sendo colectado um repositório de artigos científicos presentes em `/references/pdfs/`, assim como exemplos de documentos MusicXML que se podem encontrar em `/work/examples/musicxml/`. Considera-se ainda mais importante as bibliotecas apresentadas. A biblioteca musicxml encontra-se em `/work/MusicXML/` e a biblioteca hamusic em `/work/HaMusic/`, sendo possível fazer download em `/download/`.

Como informação adicional, inclui-se a especificação do MusicXML em DTDs em `/work/dtds/` e em XML Schemas disponível em `/work/xsd/`. Algumas transformações utilitárias sobre documentos MusicXML estão em `/work/xslt` e a documentação referente a todo o desenvolvimento encontra-se disponível em `/docs/`.

Apêndice B

Documentos Diversos

São apresentados alguns documentos neste apêndice que se achem relevantes mas de média dimensão para se encontrarem presentes no texto.

B.1 MusicXML: Casos de estudo

São apresentados os ficheiros como resultado do output gerado na execução do caso de estudo.

B.1.1 Recordare: Output

Apresenta-se o ficheiro gerado com o resultado da execução do caso de estudo Recordare.

Listagem B.1: Output do caso de estudo Recordare

```
1
2  Number: 1
3  file: "../examples/Recordare/partwise/Echigo-Jishi.xml"
4  Reading [Ok]
5  Writing [Ok]
6
7  Number: 2
8  file: "../examples/Recordare/partwise/elite.xml"
9  Reading [Ok]
10 Writing [Ok]
11
12 Number: 3
13 file: "../examples/Recordare/partwise/ActorPreludeSample.xml"
14 Reading [Ok]
15 Writing [Ok]
16
17 Number: 4
18 file: "../examples/Recordare/partwise/BeetAnGeSample.xml"
19 Reading [Ok]
20 Writing [Ok]
```

```
21
22 Number: 5
23 file: "../examples/Recordare/partwise/Binchois.xml"
24 Reading [Ok]
25 Writing [Ok]
26
27 Number: 6
28 file: "../examples/Recordare/partwise/BrahWiMeSample.xml"
29 Reading [Ok]
30 Writing [Ok]
31
32 Number: 7
33 file: "../examples/Recordare/partwise/BrookeWestSample.xml"
34 Reading [Ok]
35 Writing [Ok]
36
37 Number: 8
38 file: "../examples/Recordare/partwise/Chant.xml"
39 Reading [Ok]
40 Writing [Ok]
41
42 Number: 9
43 file: "../examples/Recordare/partwise/DebuMandSample.xml"
44 Reading [Ok]
45 Writing [Ok]
46
47 Number: 10
48 file: "../examples/Recordare/partwise/Dichterliebe01.xml"
49 Reading [Ok]
50 Writing [Ok]
51
52 Number: 11
53 file: "../examples/Recordare/partwise/FaurReveSample.xml"
54 Reading [Ok]
55 Writing [Ok]
56
57 Number: 12
58 file: "../examples/Recordare/partwise/MahlFaGe4Sample.xml"
59 Reading [Ok]
60 Writing [Ok]
61
62 Number: 13
63 file: "../examples/Recordare/partwise/MozaChloSample.xml"
64 Reading [Ok]
65 Writing [Ok]
66
67 Number: 14
68 file: "../examples/Recordare/partwise/MozaVeilSample.xml"
69 Reading [Ok]
70 Writing [Ok]
71
72 Number: 15
73 file: "../examples/Recordare/partwise/MozartPianoSonata.xml"
74 Reading [Ok]
75 Writing [Ok]
76
77 Number: 16
78 file: "../examples/Recordare/partwise/MozartTrio.xml"
79 Reading [Ok]
80 Writing [Ok]
81
82 Number: 17
```

```
83  file: "../examples/Recordare/partwise/Saltarello.xml"
84  Reading [Ok]
85  Writing [Ok]
86
87  Number: 18
88  file: "../examples/Recordare/partwise/SchbAvMaSample.xml"
89  Reading [Ok]
90  Writing [Ok]
91
92  Number: 19
93  file: "../examples/Recordare/partwise/Telemann.xml"
94  Reading [Ok]
95  Writing [Ok]
96
97  Number: 20
98  file: "../examples/Recordare/timewise/Echigo-Jishi.xml"
99  Reading [Ok]
100 Writing [Ok]
101
102 Number: 21
103 file: "../examples/Recordare/timewise/elite.xml"
104 Reading [Ok]
105 Writing [Ok]
106
107 Number: 22
108 file: "../examples/Recordare/timewise/ActorPreludeSample.xml"
109 Reading [Ok]
110 Writing [Ok]
111
112 Number: 23
113 file: "../examples/Recordare/timewise/BeetAnGeSample.xml"
114 Reading [Ok]
115 Writing [Ok]
116
117 Number: 24
118 file: "../examples/Recordare/timewise/Binchois.xml"
119 Reading [Ok]
120 Writing [Ok]
121
122 Number: 25
123 file: "../examples/Recordare/timewise/BrahWiMeSample.xml"
124 Reading [Ok]
125 Writing [Ok]
126
127 Number: 26
128 file: "../examples/Recordare/timewise/BrookeWestSample.xml"
129 Reading [Ok]
130 Writing [Ok]
131
132 Number: 27
133 file: "../examples/Recordare/timewise/Chant.xml"
134 Reading [Ok]
135 Writing [Ok]
136
137 Number: 28
138 file: "../examples/Recordare/timewise/DebuMandSample.xml"
139 Reading [Ok]
140 Writing [Ok]
141
142 Number: 29
143 file: "../examples/Recordare/timewise/Dichterliebe01.xml"
144 Reading [Ok]
```

```
145 Writing [Ok]
146
147 Number: 30
148 file: "../examples/Recordare/timewise/FaurReveSample.xml"
149 Reading [Ok]
150 Writing [Ok]
151
152 Number: 31
153 file: "../examples/Recordare/timewise/MahlFaGe4Sample.xml"
154 Reading [Ok]
155 Writing [Ok]
156
157 Number: 32
158 file: "../examples/Recordare/timewise/MozaChloSample.xml"
159 Reading [Ok]
160 Writing [Ok]
161
162 Number: 33
163 file: "../examples/Recordare/timewise/MozaVeilSample.xml"
164 Reading [Ok]
165 Writing [Ok]
166
167 Number: 34
168 file: "../examples/Recordare/timewise/MozartPianoSonata.xml"
169 Reading [Ok]
170 Writing [Ok]
171
172 Number: 35
173 file: "../examples/Recordare/timewise/MozartTrio.xml"
174 Reading [Ok]
175 Writing [Ok]
176
177 Number: 36
178 file: "../examples/Recordare/timewise/Saltarello.xml"
179 Reading [Ok]
180 Writing [Ok]
181
182 Number: 37
183 file: "../examples/Recordare/timewise/SchbAvMaSample.xml"
184 Reading [Ok]
185 Writing [Ok]
186
187 Number: 38
188 file: "../examples/Recordare/timewise/Telemann.xml"
189 Reading [Ok]
190 Writing [Ok]
```

B.1.2 Wikifonia: Input

Apresenta-se o ficheiro criado com a lista de documentos em MusicXML presentes no caso de estudo Wikifonia.

Listagem B.2: Lista de documentos MusicXML do caso de estudo Wikifonia

```
1 ../examples/Wikifonia/A.C. Jobim - One Note Samba.xml
2 ../examples/Wikifonia/ALBERTO MARAFIOTI - TEMA PER PIANOFORTE.xml
3 ../examples/Wikifonia/Aaron - U-Turn (Lili).xml
4 ../examples/Wikifonia/Adrienne Anderson, Barry Manilow - Could It Be Magic.xml
```

```
5  ../examples/Wikifonia/Air - Playground Love.xml
6  ../examples/Wikifonia/Alan Menken, Howard Ashman - Skid Row (Downtown).xml
7  ../examples/Wikifonia/Alanis Morissette, Glen Ballard - Ironic.xml
8  ../examples/Wikifonia/Alberto Marafioti  - Solfeggio di difficile esecuzione.xml
9  ../examples/Wikifonia/Alberto Marafioti  - Solfeggio di media difficolt_ in do
       Maggiore.xml
10 ../examples/Wikifonia/Alberto Marafioti - Solfeggi in chiave di violino ad uso delle
       scuole medie.xml
11 ../examples/Wikifonia/Alberto Marafioti - Solfeggio di difficile esecuzione.xml
12 ../examples/Wikifonia/Alicia Keys - Fallin '.xml
13 ../examples/Wikifonia/All-American Rejects - It Ends tonight.xml
14 ../examples/Wikifonia/Amanda McBroom - The Rose.xml
15 ../examples/Wikifonia/Amy Winehouse - Back to black.xml
16 ../examples/Wikifonia/Andrew Lloyd Webber - All I ask of you.xml
17 ../examples/Wikifonia/Andrew Lloyd Webber - Any Dream Will Do.xml
18 ../examples/Wikifonia/Andrew Lloyd Webber - Don't cry for me argentina.xml
19 ../examples/Wikifonia/Andrew Lloyd Webber, Tim Rice - Close every door.xml
20 ../examples/Wikifonia/Anita Kerr - A House Is Not A Home.xml
21 ../examples/Wikifonia/Anonymous - The Highway To Limerick.xml
22 ../examples/Wikifonia/Anouk - Our own love.xml
23 ../examples/Wikifonia/Antonio Carlos Jobim - The Girl from Ipanema.xml
24 ../examples/Wikifonia/Barry Goldberg - Blues for Barry.xml
25 ../examples/Wikifonia/Bart Howard - Fly me to the moon.xml
26 ../examples/Wikifonia/Bee Gees - How deep is your love.xml
27 ../examples/Wikifonia/Bellini - Samba de Janeiro.xml
28 ../examples/Wikifonia/Benny Andersson, Stig Anderson, Bjorn Ulvaeus - Dancing Queen.
       xml
29 ../examples/Wikifonia/Benny Carter - When lights are low.xml
30 ../examples/Wikifonia/Benny Golson - Whisper Not.xml
31 ../examples/Wikifonia/Bernie Miller - Bernie's Tune.xml
32 ../examples/Wikifonia/Billy Joel - Just The Way You Are.xml
33 ../examples/Wikifonia/Billy Joel - Lullabye (Good night, my angel).xml
34 ../examples/Wikifonia/Billy Steinberg - Eternal Flame.xml
35 ../examples/Wikifonia/Billy Strayhorn - Take The 'A' Train.xml
36 ../examples/Wikifonia/Blink 182 - Mutt.xml
37 ../examples/Wikifonia/Bob Haggard - What's New.xml
38 ../examples/Wikifonia/Britney Spears - (You Drive Me) Crazy.xml
39 ../examples/Wikifonia/Bronislau Kaper - Invitation.xml
40 ../examples/Wikifonia/Bryan Adams - (Everything I Do) I Do It For You.xml
41 ../examples/Wikifonia/Bryan Adams, Jim Vallance - The Summer of 69.xml
42 ../examples/Wikifonia/Burt Bacharach, Hal David - I Say A Little Prayer.xml
43 ../examples/Wikifonia/Charles Fox, Norman Gimbel - Killing me softly.xml
44 ../examples/Wikifonia/Charlie Chaplin - Smile.xml
45 ../examples/Wikifonia/Charlie Parker (1920-1955) - Au Privave.xml
46 ../examples/Wikifonia/Charlie Parker - Anthropology.xml
47 ../examples/Wikifonia/Charlie Parker - Confirmation.xml
48 ../examples/Wikifonia/Charlie Parker - Cool Blues.xml
49 ../examples/Wikifonia/Charlie Parker - Moose the mooche.xml
50 ../examples/Wikifonia/Charlie Parker, Dizzy Gillespie - Groovin' High.xml
51 ../examples/Wikifonia/Chicago - If You Leave Me Now.xml
52 ../examples/Wikifonia/Chick Corea - 500 Miles High.xml
53 ../examples/Wikifonia/Christina Aguilera - Beautiful.xml
54 ../examples/Wikifonia/Christina Aguilera - Hurt.xml
55 ../examples/Wikifonia/Claude Joseph Rouget de Lisle - La Marseillaise.xml
56 ../examples/Wikifonia/Coldplay - God Put a Smile Upon Your Face.xml
57 ../examples/Wikifonia/Coldplay - What if.xml
58 ../examples/Wikifonia/Cole Porter - Night And Day.xml
59 ../examples/Wikifonia/Cy Coleman, Dorothy Fields - Big Spender.xml
60 ../examples/Wikifonia/Dario G - Carneval de Paris.xml
61 ../examples/Wikifonia/Dino Fekaris, Freddie Perren - I Will Survive.xml
62 ../examples/Wikifonia/Dizzy Gillespie - A Night In Tunesia.xml
63 ../examples/Wikifonia/Don McLean - American Pie.xml
```

```
64 ../examples/Wikifonia/Donald Fagen - Security joan.xml
65 ../examples/Wikifonia/Donald Kahn - A Beautiful Friendship.xml
66 ../examples/Wikifonia/Duke Ellington - Caravan.xml
67 ../examples/Wikifonia/Duke Ellington - In a Sentimental Mood.xml
68 ../examples/Wikifonia/Duke Ellington - It Don't Mean a Thing.xml
69 ../examples/Wikifonia/Duke Ellington - Mood Indigo.xml
70 ../examples/Wikifonia/Duke Ellington - Solitude.xml
71 ../examples/Wikifonia/Duke Ellington - Sophisticated Lady.xml
72 ../examples/Wikifonia/Duke Ellington, Billy Strayhorn - Satin Doll.xml
73 ../examples/Wikifonia/Eagle-Eye Cherry - Save Tonight.xml
74 ../examples/Wikifonia/Elton John - Can You Feel The Love Tonight.xml
75 ../examples/Wikifonia/Elton John, Bernie Taupin - Nikita.xml
76 ../examples/Wikifonia/Elvis Presley - Love me tender.xml
77 ../examples/Wikifonia/Four Seasons - Big Girls Don't Cry.xml
78 ../examples/Wikifonia/Four Seasons - Walk like a man.xml
79 ../examples/Wikifonia/Fran_ois van Campenhout - La Braban_onne.xml
80 ../examples/Wikifonia/Frank Churchill - Someday My Prince Will Come.xml
81 ../examples/Wikifonia/Frank Loesser - If I were a bell.xml
82 ../examples/Wikifonia/Franz Grubert - Silent Night.xml
83 ../examples/Wikifonia/Freddie Mercury - Bohemian Rhapsody.xml
84 ../examples/Wikifonia/Freddie Mercury - We are the Champions.xml
85 ../examples/Wikifonia/Genesis - In too deep.xml
86 ../examples/Wikifonia/George Gershwin - I Got Rhythm.xml
87 ../examples/Wikifonia/George Gershwin - Summertime.xml
88 ../examples/Wikifonia/Giacomo Mason - New Friends.xml
89 ../examples/Wikifonia/Giorgo Moroder, Irene Cara, Keith Forsey - Flashdance... What a
       Feeling.xml
90 ../examples/Wikifonia/Greg Johnston - Smooth Talk.xml
91 ../examples/Wikifonia/Guy Roche, Shelly Peiken - What A Girl Wants.xml
92 ../examples/Wikifonia/Guys and Dolls - Luck be a lady.xml
93 ../examples/Wikifonia/Gwen Stefani - Don't Speak.xml
94 ../examples/Wikifonia/Hair - Aquarius.xml
95 ../examples/Wikifonia/Heather Nova - Heaven Sent.xml
96 ../examples/Wikifonia/Heather Nova - Maybe an angel.xml
97 ../examples/Wikifonia/Henry Mancini - The Days of Wine and Roses.xml
98 ../examples/Wikifonia/Herbie Hancock - Cantaloupe Island.xml
99 ../examples/Wikifonia/Herbie Hancock - Dolphin Dance.xml
100 ../examples/Wikifonia/Herbie Hancock - Maiden Voyage.xml
101 ../examples/Wikifonia/Herbie Hancock - Watermelon Man.xml
102 ../examples/Wikifonia/Horace Silver - Song for My Father.xml
103 ../examples/Wikifonia/Hoyt Curtin - Meet The Flintstones.xml
104 ../examples/Wikifonia/J. Calvert, G. Hughes - We're Going To Ibiza.xml
105 ../examples/Wikifonia/Jack Strachey - These Foolish Things.xml
106 ../examples/Wikifonia/Jean-Baptiste Lully - God save the queen.xml
107 ../examples/Wikifonia/Jerome Kern, Johnny Mercer - I'm Old Fashioned.xml
108 ../examples/Wikifonia/Jerome Richardson - Groove Merchant.xml
109 ../examples/Wikifonia/Jerry Bock, Sheldon Harnick - To Life.xml
110 ../examples/Wikifonia/Jerry Leiber, Mike Stoller - Bernie's Tune.xml
111 ../examples/Wikifonia/Jiggs Whigham - Cape Clip So.xml
112 ../examples/Wikifonia/Joe Henderson - Mamacita.xml
113 ../examples/Wikifonia/Joe Henderson - No Me Esqueca.xml
114 ../examples/Wikifonia/Johann Strauss Jr. - Blue Danube.xml
115 ../examples/Wikifonia/John Coltrane - Mr. P.C..xml
116 ../examples/Wikifonia/John Denver - Annie's Song.xml
117 ../examples/Wikifonia/John Lennon - Imagine.xml
118 ../examples/Wikifonia/John Lennon, Paul McCartney - A Hard Day's Night.xml
119 ../examples/Wikifonia/John Lennon, Paul McCartney - Eleanor Rigby.xml
120 ../examples/Wikifonia/John Lennon, Paul McCartney - Lady Madonna.xml
121 ../examples/Wikifonia/John Reading - O Come All Ye Faithful.xml
122 ../examples/Wikifonia/Johnny Green - Out of Nowhere.xml
123 ../examples/Wikifonia/Jon Lind, Allee Willis - Boogie Wonderland.xml
124 ../examples/Wikifonia/Joseph Haydn - Deutschlandlied.xml
```

```
125 ../examples/Wikifonia/Joseph Kosma, Jacques Prevert - Autumn Leaves.xml
126 ../examples/Wikifonia/Juan Tizol - Perdido.xml
127 ../examples/Wikifonia/Jule Styne - Time After Time.xml
128 ../examples/Wikifonia/K. Dorham - Blue Bossa.xml
129 ../examples/Wikifonia/Kaiser Chiefs - Ruby.xml
130 ../examples/Wikifonia/Katrina and The Waves - Walking On Sunshine.xml
131 ../examples/Wikifonia/Kenny Dorham - Lotus Blossom.xml
132 ../examples/Wikifonia/Kris Wauters, Koen Wauters - Passie.xml
133 ../examples/Wikifonia/L.Cohen - Hallelujah.xml
134 ../examples/Wikifonia/Leon Russell - A Song For You.xml
135 ../examples/Wikifonia/Leonard Cohen - Hallelujah.xml
136 ../examples/Wikifonia/Lifehouse - You & Me.xml
137 ../examples/Wikifonia/Lou Reed - Perfect Day.xml
138 ../examples/Wikifonia/Luiz Bonfa - Black Orpheus.xml
139 ../examples/Wikifonia/Mark Knopfler - Romeo and Juliet.xml
140 ../examples/Wikifonia/Mars Bonfire - Born to be wild.xml
141 ../examples/Wikifonia/Max Martin - ... Baby One More time.xml
142 ../examples/Wikifonia/Mendez, Shakira - Underneath Your Clothes.xml
143 ../examples/Wikifonia/Meredith Willson - Till there was you.xml
144 ../examples/Wikifonia/Michael Bubl_ - Everything.xml
145 ../examples/Wikifonia/Michael Jackson - Beat it!.xml
146 ../examples/Wikifonia/Miles Davis - All Blues.xml
147 ../examples/Wikifonia/Miles Davis - Four.xml
148 ../examples/Wikifonia/Miles Davis - Solar.xml
149 ../examples/Wikifonia/Miles Davis - Tune Up.xml
150 ../examples/Wikifonia/Muse - Unintended.xml
151 ../examples/Wikifonia/Nelly Furtado - All good things.xml
152 ../examples/Wikifonia/Nick Cave - Into my Arms.xml
153 ../examples/Wikifonia/Nickolas Ash - Ain't No Mountain High Enough.xml
154 ../examples/Wikifonia/Noa - Beautiful that way.xml
155 ../examples/Wikifonia/Norah Jones - Come away with me.xml
156 ../examples/Wikifonia/Oasis - Wonderwall.xml
157 ../examples/Wikifonia/Pat Metheny - Bachelors III.xml
158 ../examples/Wikifonia/Paul Desmond - Take Five.xml
159 ../examples/Wikifonia/Peter Gabriel - Steam.xml
160 ../examples/Wikifonia/Phil Thornalley, Anne Previn, Scott Cutler - Torn.xml
161 ../examples/Wikifonia/Procol Harum - A Whiter Shade of Pale.xml
162 ../examples/Wikifonia/R. Kelly - I Believe I can Fly.xml
163 ../examples/Wikifonia/REM - Everybody Hurts continued.xml
164 ../examples/Wikifonia/Ray Charles - Hallelujah, I love her so.xml
165 ../examples/Wikifonia/Ray Noble - Cherokee.xml
166 ../examples/Wikifonia/Red Hot Chili Peppers - Under the Bridge.xml
167 ../examples/Wikifonia/Richard Rodgers, Lorenz Hart - The Lady Is A Tramp.xml
168 ../examples/Wikifonia/Richard Rodgers, Lorenz hart - My Funny Valentine.xml
169 ../examples/Wikifonia/Richard Rodgers, Oscar Hammerstein - Edelweiss.xml
170 ../examples/Wikifonia/Ricky Martin - Livin' La Vlda Loca.xml
171 ../examples/Wikifonia/Ritchie Valens - La Bamba.xml
172 ../examples/Wikifonia/Robbie Williams, Guy Chambers - Angels.xml
173 ../examples/Wikifonia/Robbie Williams, Guy Chambers - Feel.xml
174 ../examples/Wikifonia/Robbie Williams, Guy Chambers - Let Me Entertain You.xml
175 ../examples/Wikifonia/Robert Lopez, Jeff Marx - Avenue Q.xml
176 ../examples/Wikifonia/Sade Adu, Ray St. John - Smooth Operator.xml
177 ../examples/Wikifonia/Scott Auble - Scott's Tune.xml
178 ../examples/Wikifonia/Sergei Rachmaninov, Eric Carmen - All By Myself.xml
179 ../examples/Wikifonia/Seymour Simons - All Of Me.xml
180 ../examples/Wikifonia/Sigmund Romberg, Oscar Hammerstein II - Softly as in a morning
        sunrise.xml
181 ../examples/Wikifonia/Snow Patrol - Chasing Cars.xml
182 ../examples/Wikifonia/Sonny Rollins - Pent Up House.xml
183 ../examples/Wikifonia/Sonny Rollins - St. Thomas.xml
184 ../examples/Wikifonia/Sonny Rollins - Tenor Madness.xml
185 ../examples/Wikifonia/Sting - Every Little Thing She Does Is Magic.xml
```

```
186 ../examples/Wikifonia/Sting - Roxanne.xml
187 ../examples/Wikifonia/Sting, The Police - Every breath you take.xml
188 ../examples/Wikifonia/Tadd Dameron - Lady Bird.xml
189 ../examples/Wikifonia/The Fray - How to Save a Life.xml
190 ../examples/Wikifonia/The Knife, Jose Gonzalez - Heartbeats.xml
191 ../examples/Wikifonia/The Kooks - Naive.xml
192 ../examples/Wikifonia/The Rolling Stones - Paint it black.xml
193 ../examples/Wikifonia/Thelonious Monk - 'Round Midnight.xml
194 ../examples/Wikifonia/Thelonious Monk - Straight, No Chaser.xml
195 ../examples/Wikifonia/They might be Giants - Experimental Film.xml
196 ../examples/Wikifonia/Thomas Waller - Honeysuckle Rose.xml
197 ../examples/Wikifonia/Thomas Yorke - Karma Police.xml
198 ../examples/Wikifonia/Toots Thielemans - Bluesette.xml
199 ../examples/Wikifonia/Toto - Africa.xml
200 ../examples/Wikifonia/Toto - Hold the line.xml
201 ../examples/Wikifonia/Tracy Chapman - The Wedding Song.xml
202 ../examples/Wikifonia/Traditional  - Dag Sinterklaasje.xml
203 ../examples/Wikifonia/Traditional - Angels We Have Heard On High.xml
204 ../examples/Wikifonia/Traditional - De zak van Sinterklaas.xml
205 ../examples/Wikifonia/Traditional - O kom er eens kijken.xml
206 ../examples/Wikifonia/Traditional - Sinterklaas is jarig.xml
207 ../examples/Wikifonia/Traditional - Sinterklaas kapoentje.xml
208 ../examples/Wikifonia/Traditional - Sinterklaasje bonne bonne bonne.xml
209 ../examples/Wikifonia/Traditional - Sinterklaasje kom maar binnen met je knecht.xml
210 ../examples/Wikifonia/Traditional - We wish you a merry christmas.xml
211 ../examples/Wikifonia/Traditional - What child is this_.xml
212 ../examples/Wikifonia/Traditional - Zachtjes gaan de paardenvoetjes.xml
213 ../examples/Wikifonia/Traditional - Zie ginds komt de stoomboot.xml
214 ../examples/Wikifonia/Trent Reznor - Hurt.xml
215 ../examples/Wikifonia/U2 - With or without you.xml
216 ../examples/Wikifonia/Village People  - Y.M.C.A.xml
217 ../examples/Wikifonia/Walter Afanasieff, Mariah Carey - Hero.xml
218 ../examples/Wikifonia/Wayne Thomps - Always On My Mind.xml
219 ../examples/Wikifonia/Willem Vermandere - Blanche en zijn peird.xml
220 ../examples/Wikifonia/Zombies - Time of the Season.xml
221 ../examples/Wikifonia/anonymous - Wilhelmus van Nassouwe.xml
```

B.1.3 Wikifonia: Output

Apresenta-se o ficheiro gerado com o resultado da execução do caso de estudo Wikifonia.

Listagem B.3: Output do caso de estudo Wikifonia

```
1
2  Number: 1
3  file: "../examples/Wikifonia/A.C. Jobim - One Note Samba.xml"
4  Reading [Ok]
5  Writing [Ok]
6
7  Number: 2
8  file: "../examples/Wikifonia/ALBERTO MARAFIOTI - TEMA PER PIANOFORTE.xml"
9  Reading [Ok]
10 Writing [Ok]
11
12 Number: 3
13 file: "../examples/Wikifonia/Aaron - U-Turn (Lili).xml"
14 Reading [Ok]
```

```
15 Writing [Ok]
16
17 Number: 4
18 file: "../examples/Wikifonia/Adrienne␣Anderson,␣Barry␣Manilow␣-␣Could␣It␣Be␣Magic.xml
       "
19 Reading [Ok]
20 Writing [Ok]
21
22 Number: 5
23 file: "../examples/Wikifonia/Air␣-␣Playground␣Love.xml"
24 Reading [Ok]
25 Writing [Ok]
26
27 Number: 6
28 file: "../examples/Wikifonia/Alan␣Menken,␣Howard␣Ashman␣-␣Skid␣Row␣(Downtown).xml"
29 Reading [Ok]
30 Writing [Ok]
31
32 Number: 7
33 file: "../examples/Wikifonia/Alanis␣Morissette,␣Glen␣Ballard␣-␣Ironic.xml"
34 Reading [Ok]
35 Writing [Ok]
36
37 Number: 8
38 file: "../examples/Wikifonia/Alberto␣Marafioti␣␣␣-␣Solfeggio␣di␣difficile␣esecuzione.
       xml"
39 Reading [Ok]
40 Writing [Ok]
41
42 Number: 9
43 file: "../examples/Wikifonia/Alberto␣Marafioti␣␣-␣Solfeggio␣di␣media␣difficolt_␣in␣do
       ␣Maggiore.xml"
44 Reading [Ok]
45 Writing [Ok]
46
47 Number: 10
48 file: "../examples/Wikifonia/Alberto␣Marafioti␣-␣Solfeggi␣in␣chiave␣di␣violino␣ad␣uso
       ␣delle␣scuole␣medie.xml"
49 Reading [Ok]
50 Writing [Ok]
51
52 Number: 11
53 file: "../examples/Wikifonia/Alberto␣Marafioti␣-␣Solfeggio␣di␣difficile␣esecuzione.
       xml"
54 Reading [Ok]
55 Writing [Ok]
56
57 Number: 12
58 file: "../examples/Wikifonia/Alicia␣Keys␣-␣Fallin'.xml"
59 Reading [Ok]
60 Writing [Ok]
61
62 Number: 13
63 file: "../examples/Wikifonia/All-American␣Rejects␣-␣It␣Ends␣tonight.xml"
64 Reading [Ok]
65 Writing [Ok]
66
67 Number: 14
68 file: "../examples/Wikifonia/Amanda␣McBroom␣-␣The␣Rose.xml"
69 Reading [Ok]
70 Writing [Ok]
71
```

```
Number: 15
file: "../examples/Wikifonia/Amy Winehouse - Back to black.xml"
Reading [Ok]
Writing [Ok]

Number: 16
file: "../examples/Wikifonia/Andrew Lloyd Webber - All I ask of you.xml"
Reading [Ok]
Writing [Ok]

Number: 17
file: "../examples/Wikifonia/Andrew Lloyd Webber - Any Dream Will Do.xml"
Reading [Ok]
Writing [Ok]

Number: 18
file: "../examples/Wikifonia/Andrew Lloyd Webber - Don't cry for me argentina.xml"
Reading [Ok]
Writing [Ok]

Number: 19
file: "../examples/Wikifonia/Andrew Lloyd Webber, Tim Rice - Close every door.xml"
Reading [Ok]
Writing [Ok]

Number: 20
file: "../examples/Wikifonia/Anita Kerr - A House Is Not A Home.xml"
Reading [Ok]
Writing [Ok]

Number: 21
file: "../examples/Wikifonia/Anonymous - The Highway To Limerick.xml"
Reading [Ok]
Writing [Ok]

Number: 22
file: "../examples/Wikifonia/Anouk - Our own love.xml"
Reading [Ok]
Writing [Ok]

Number: 23
file: "../examples/Wikifonia/Antonio Carlos Jobim - The Girl from Ipanema.xml"
Reading [Ok]
Writing [Ok]

Number: 24
file: "../examples/Wikifonia/Barry Goldberg - Blues for Barry.xml"
Reading [Ok]
Writing [Ok]

Number: 25
file: "../examples/Wikifonia/Bart Howard - Fly me to the moon.xml"
Reading [Ok]
Writing [Ok]

Number: 26
file: "../examples/Wikifonia/Bee Gees - How deep is your love.xml"
Reading [Ok]
Writing [Ok]

Number: 27
file: "../examples/Wikifonia/Bellini - Samba de Janeiro.xml"
```

```
134 Reading [Ok]
135 Writing [Ok]
136
137 Number: 28
138 file: "../examples/Wikifonia/Benny Andersson, Stig Anderson, Bjorn Ulvaeus - Dancing
        Queen.xml"
139 Reading [Ok]
140 Writing [Ok]
141
142 Number: 29
143 file: "../examples/Wikifonia/Benny Carter - When lights are low.xml"
144 Reading [Ok]
145 Writing [Ok]
146
147 Number: 30
148 file: "../examples/Wikifonia/Benny Golson - Whisper Not.xml"
149 Reading [Ok]
150 Writing [Ok]
151
152 Number: 31
153 file: "../examples/Wikifonia/Bernie Miller - Bernie's Tune.xml"
154 Reading [Ok]
155 Writing [Ok]
156
157 Number: 32
158 file: "../examples/Wikifonia/Billy Joel - Just The Way You Are.xml"
159 Reading [Ok]
160 Writing [Ok]
161
162 Number: 33
163 file: "../examples/Wikifonia/Billy Joel - Lullabye (Good night, my angel).xml"
164 Reading [Ok]
165 Writing [Ok]
166
167 Number: 34
168 file: "../examples/Wikifonia/Billy Steinberg - Eternal Flame.xml"
169 Reading [Ok]
170 Writing [Ok]
171
172 Number: 35
173 file: "../examples/Wikifonia/Billy Strayhorn - Take The 'A' Train.xml"
174 Reading [Ok]
175 Writing [Ok]
176
177 Number: 36
178 file: "../examples/Wikifonia/Blink 182 - Mutt.xml"
179 Reading [Ok]
180 Writing [Ok]
181
182 Number: 37
183 file: "../examples/Wikifonia/Bob Haggard - What's New.xml"
184 Reading [Ok]
185 Writing [Ok]
186
187 Number: 38
188 file: "../examples/Wikifonia/Britney Spears - (You Drive Me) Crazy.xml"
189 Reading [Ok]
190 Writing [Ok]
191
192 Number: 39
193 file: "../examples/Wikifonia/Bronislau Kaper - Invitation.xml"
194 Reading [Ok]
```

```
Writing [Ok]

Number: 40
file: "../examples/Wikifonia/Bryan Adams - (Everything I Do) I Do It For You.xml"
Reading [Ok]
Writing [Ok]

Number: 41
file: "../examples/Wikifonia/Bryan Adams, Jim Vallance - The Summer of 69.xml"
Reading [Ok]
Writing [Ok]

Number: 42
file: "../examples/Wikifonia/Burt Bacharach, Hal David - I Say A Little Prayer.xml"
Reading [Ok]
Writing [Ok]

Number: 43
file: "../examples/Wikifonia/Charles Fox, Norman Gimbel - Killing me softly.xml"
Reading [Ok]
Writing [Ok]

Number: 44
file: "../examples/Wikifonia/Charlie Chaplin - Smile.xml"
Reading [Ok]
Writing [Ok]

Number: 45
file: "../examples/Wikifonia/Charlie Parker (1920-1955) - Au Privave.xml"
Reading [Ok]
Writing [Ok]

Number: 46
file: "../examples/Wikifonia/Charlie Parker - Anthropology.xml"
Reading [Ok]
Writing [Ok]

Number: 47
file: "../examples/Wikifonia/Charlie Parker - Confirmation.xml"
Reading [Ok]
Writing [Ok]

Number: 48
file: "../examples/Wikifonia/Charlie Parker - Cool Blues.xml"
Reading [Ok]
Writing [Ok]

Number: 49
file: "../examples/Wikifonia/Charlie Parker - Moose the mooche.xml"
Reading [Ok]
Writing [Ok]

Number: 50
file: "../examples/Wikifonia/Charlie Parker, Dizzy Gillespie - Groovin' High.xml"
Reading [Ok]
Writing [Ok]

Number: 51
file: "../examples/Wikifonia/Chicago - If You Leave Me Now.xml"
Reading [Ok]
Writing [Ok]

```

```
257 Number: 52
258 file: "../examples/Wikifonia/Chick Corea - 500 Miles High.xml"
259 Reading [Ok]
260 Writing [Ok]
261
262 Number: 53
263 file: "../examples/Wikifonia/Christina Aguilera - Beautiful.xml"
264 Reading [Ok]
265 Writing [Ok]
266
267 Number: 54
268 file: "../examples/Wikifonia/Christina Aguilera - Hurt.xml"
269 Reading [Ok]
270 Writing [Ok]
271
272 Number: 55
273 file: "../examples/Wikifonia/Claude Joseph Rouget de Lisle - La Marseillaise.xml"
274 Reading [Ok]
275 Writing [Ok]
276
277 Number: 56
278 file: "../examples/Wikifonia/Coldplay - God Put a Smile Upon Your Face.xml"
279 Reading [Ok]
280 Writing [Ok]
281
282 Number: 57
283 file: "../examples/Wikifonia/Coldplay - What if.xml"
284 Reading [Ok]
285 Writing [Ok]
286
287 Number: 58
288 file: "../examples/Wikifonia/Cole Porter - Night And Day.xml"
289 Reading [Ok]
290 Writing [Ok]
291
292 Number: 59
293 file: "../examples/Wikifonia/Cy Coleman, Dorothy Fields - Big Spender.xml"
294 Reading [Ok]
295 Writing [Ok]
296
297 Number: 60
298 file: "../examples/Wikifonia/Dario G - Carneval de Paris.xml"
299 Reading [Ok]
300 Writing [Ok]
301
302 Number: 61
303 file: "../examples/Wikifonia/Dino Fekaris, Freddie Perren - I Will Survive.xml"
304 Reading [Ok]
305 Writing [Ok]
306
307 Number: 62
308 file: "../examples/Wikifonia/Dizzy Gillespie - A Night In Tunesia.xml"
309 Reading [Ok]
310 Writing [Ok]
311
312 Number: 63
313 file: "../examples/Wikifonia/Don McLean - American Pie.xml"
314 Reading [Ok]
315 Writing [Ok]
316
317 Number: 64
318 file: "../examples/Wikifonia/Donald Fagen - Security joan.xml"
```

```
Reading [Ok]
Writing [Ok]

Number: 65
file: "../examples/Wikifonia/Donald Kahn - A Beautiful Friendship.xml"
Reading [Ok]
Writing [Ok]

Number: 66
file: "../examples/Wikifonia/Duke Ellington - Caravan.xml"
Reading [Ok]
Writing [Ok]

Number: 67
file: "../examples/Wikifonia/Duke Ellington - In a Sentimental Mood.xml"
Reading [Ok]
Writing [Ok]

Number: 68
file: "../examples/Wikifonia/Duke Ellington - It Don't Mean a Thing.xml"
Reading [Ok]
Writing [Ok]

Number: 69
file: "../examples/Wikifonia/Duke Ellington - Mood Indigo.xml"
Reading [Ok]
Writing [Ok]

Number: 70
file: "../examples/Wikifonia/Duke Ellington - Solitude.xml"
Reading [Ok]
Writing [Ok]

Number: 71
file: "../examples/Wikifonia/Duke Ellington - Sophisticated Lady.xml"
Reading [Ok]
Writing [Ok]

Number: 72
file: "../examples/Wikifonia/Duke Ellington , Billy Strayhorn - Satin Doll.xml"
Reading [Ok]
Writing [Ok]

Number: 73
file: "../examples/Wikifonia/Eagle-Eye Cherry - Save Tonight.xml"
Reading [Ok]
Writing [Ok]

Number: 74
file: "../examples/Wikifonia/Elton John - Can You Feel The Love Tonight.xml"
Reading [Ok]
Writing [Ok]

Number: 75
file: "../examples/Wikifonia/Elton John , Bernie Taupin - Nikita.xml"
Reading [Ok]
Writing [Ok]

Number: 76
file: "../examples/Wikifonia/Elvis Presley - Love me tender.xml"
Reading [Ok]
Writing [Ok]
```

```
381
382 Number: 77
383 file: "../examples/Wikifonia/Four␣Seasons␣-␣Big␣Girls␣Don't␣Cry.xml"
384 Reading [Ok]
385 Writing [Ok]
386
387 Number: 78
388 file: "../examples/Wikifonia/Four␣Seasons␣-␣Walk␣like␣a␣man.xml"
389 Reading [Ok]
390 Writing [Ok]
391
392 Number: 79
393 file: "../examples/Wikifonia/Fran_ois␣van␣Campenhout␣-␣La␣Braban_onne.xml"
394 Reading [Ok]
395 Writing [Ok]
396
397 Number: 80
398 file: "../examples/Wikifonia/Frank␣Churchill␣-␣Someday␣My␣Prince␣Will␣Come.xml"
399 Reading [Ok]
400 Writing [Ok]
401
402 Number: 81
403 file: "../examples/Wikifonia/Frank␣Loesser␣-␣If␣I␣were␣a␣bell.xml"
404 Reading [Ok]
405 Writing [Ok]
406
407 Number: 82
408 file: "../examples/Wikifonia/Franz␣Grubert␣-␣Silent␣Night.xml"
409 Reading [Ok]
410 Writing [Ok]
411
412 Number: 83
413 file: "../examples/Wikifonia/Freddie␣Mercury␣-␣Bohemian␣Rhapsody.xml"
414 Reading [Ok]
415 Writing [Ok]
416
417 Number: 84
418 file: "../examples/Wikifonia/Freddie␣Mercury␣-␣We␣are␣the␣Champions.xml"
419 Reading [Ok]
420 Writing [Ok]
421
422 Number: 85
423 file: "../examples/Wikifonia/Genesis␣-␣In␣too␣deep.xml"
424 Reading [Ok]
425 Writing [Ok]
426
427 Number: 86
428 file: "../examples/Wikifonia/George␣Gershwin␣-␣I␣Got␣Rhythm.xml"
429 Reading [Ok]
430 Writing [Ok]
431
432 Number: 87
433 file: "../examples/Wikifonia/George␣Gershwin␣-␣Summertime.xml"
434 Reading [Ok]
435 Writing [Ok]
436
437 Number: 88
438 file: "../examples/Wikifonia/Giacomo␣Mason␣-␣New␣Friends.xml"
439 Reading [Ok]
440 Writing [Ok]
441
442 Number: 89
```

```
443 file: "../examples/Wikifonia/Giorgo Moroder, Irene Cara, Keith Forsey - Flashdance...
        What a Feeling.xml"
444 Reading [Ok]
445 Writing [Ok]
446
447 Number: 90
448 file: "../examples/Wikifonia/Greg Johnston - Smooth Talk.xml"
449 Reading [Ok]
450 Writing [Ok]
451
452 Number: 91
453 file: "../examples/Wikifonia/Guy Roche, Shelly Peiken - What A Girl Wants.xml"
454 Reading [Ok]
455 Writing [Ok]
456
457 Number: 92
458 file: "../examples/Wikifonia/Guys and Dolls - Luck be a lady.xml"
459 Reading [Ok]
460 Writing [Ok]
461
462 Number: 93
463 file: "../examples/Wikifonia/Gwen Stefani - Don't Speak.xml"
464 Reading [Ok]
465 Writing [Ok]
466
467 Number: 94
468 file: "../examples/Wikifonia/Hair - Aquarius.xml"
469 Reading [Ok]
470 Writing [Ok]
471
472 Number: 95
473 file: "../examples/Wikifonia/Heather Nova - Heaven Sent.xml"
474 Reading [Ok]
475 Writing [Ok]
476
477 Number: 96
478 file: "../examples/Wikifonia/Heather Nova - Maybe an angel.xml"
479 Reading [Ok]
480 Writing [Ok]
481
482 Number: 97
483 file: "../examples/Wikifonia/Henry Mancini - The Days of Wine and Roses.xml"
484 Reading [Ok]
485 Writing [Ok]
486
487 Number: 98
488 file: "../examples/Wikifonia/Herbie Hancock - Cantaloupe Island.xml"
489 Reading [Ok]
490 Writing [Ok]
491
492 Number: 99
493 file: "../examples/Wikifonia/Herbie Hancock - Dolphin Dance.xml"
494 Reading [Ok]
495 Writing [Ok]
496
497 Number: 100
498 file: "../examples/Wikifonia/Herbie Hancock - Maiden Voyage.xml"
499 Reading [Ok]
500 Writing [Ok]
501
502 Number: 101
503 file: "../examples/Wikifonia/Herbie Hancock - Watermelon Man.xml"
```

```
504 Reading [Ok]
505 Writing [Ok]
506
507 Number: 102
508 file: "../examples/Wikifonia/Horace Silver - Song for My Father.xml"
509 Reading [Ok]
510 Writing [Ok]
511
512 Number: 103
513 file: "../examples/Wikifonia/Hoyt Curtin - Meet The Flintstones.xml"
514 Reading [Ok]
515 Writing [Ok]
516
517 Number: 104
518 file: "../examples/Wikifonia/J. Calvert, G. Hughes - We're Going To Ibiza.xml"
519 Reading [Ok]
520 Writing [Ok]
521
522 Number: 105
523 file: "../examples/Wikifonia/Jack Strachey - These Foolish Things.xml"
524 Reading [Ok]
525 Writing [Ok]
526
527 Number: 106
528 file: "../examples/Wikifonia/Jean-Baptiste Lully - God save the queen.xml"
529 Reading [Ok]
530 Writing [Ok]
531
532 Number: 107
533 file: "../examples/Wikifonia/Jerome Kern, Johnny Mercer - I'm Old Fashioned.xml"
534 Reading [Ok]
535 Writing [Ok]
536
537 Number: 108
538 file: "../examples/Wikifonia/Jerome Richardson - Groove Merchant.xml"
539 Reading [Ok]
540 Writing [Ok]
541
542 Number: 109
543 file: "../examples/Wikifonia/Jerry Bock, Sheldon Harnick - To Life.xml"
544 Reading [Ok]
545 Writing [Ok]
546
547 Number: 110
548 file: "../examples/Wikifonia/Jerry Leiber, Mike Stoller - Bernie's Tune.xml"
549 Reading [Ok]
550 Writing [Ok]
551
552 Number: 111
553 file: "../examples/Wikifonia/Jiggs Whigham - Cape Clip So.xml"
554 Reading [Ok]
555 Writing [Ok]
556
557 Number: 112
558 file: "../examples/Wikifonia/Joe Henderson - Mamacita.xml"
559 Reading [Ok]
560 Writing [Ok]
561
562 Number: 113
563 file: "../examples/Wikifonia/Joe Henderson - No Me Esqueca.xml"
564 Reading [Ok]
565 Writing [Ok]
```

```
566
567 Number: 114
568 file: "../examples/Wikifonia/Johann␣Strauss␣Jr.␣-␣Blue␣Danube.xml"
569 Reading [Ok]
570 Writing [Ok]
571
572 Number: 115
573 file: "../examples/Wikifonia/John␣Coltrane␣-␣Mr.␣P.C..xml"
574 Reading [Ok]
575 Writing [Ok]
576
577 Number: 116
578 file: "../examples/Wikifonia/John␣Denver␣-␣Annie's␣Song.xml"
579 Reading [Ok]
580 Writing [Ok]
581
582 Number: 117
583 file: "../examples/Wikifonia/John␣Lennon␣-␣Imagine.xml"
584 Reading [Ok]
585 Writing [Ok]
586
587 Number: 118
588 file: "../examples/Wikifonia/John␣Lennon,␣Paul␣McCartney␣-␣A␣Hard␣Day's␣Night.xml"
589 Reading [Ok]
590 Writing [Ok]
591
592 Number: 119
593 file: "../examples/Wikifonia/John␣Lennon,␣Paul␣McCartney␣-␣Eleanor␣Rigby.xml"
594 Reading [Ok]
595 Writing [Ok]
596
597 Number: 120
598 file: "../examples/Wikifonia/John␣Lennon,␣Paul␣McCartney␣-␣Lady␣Madonna.xml"
599 Reading [Ok]
600 Writing [Ok]
601
602 Number: 121
603 file: "../examples/Wikifonia/John␣Reading␣-␣O␣Come␣All␣Ye␣Faithful.xml"
604 Reading [Ok]
605 Writing [Ok]
606
607 Number: 122
608 file: "../examples/Wikifonia/Johnny␣Green␣-␣Out␣of␣Nowhere.xml"
609 Reading [Ok]
610 Writing [Ok]
611
612 Number: 123
613 file: "../examples/Wikifonia/Jon␣Lind,␣Allee␣Willis␣-␣Boogie␣Wonderland.xml"
614 Reading [Ok]
615 Writing [Ok]
616
617 Number: 124
618 file: "../examples/Wikifonia/Joseph␣Haydn␣-␣Deutschlandlied.xml"
619 Reading [Ok]
620 Writing [Ok]
621
622 Number: 125
623 file: "../examples/Wikifonia/Joseph␣Kosma,␣Jacques␣Prevert␣-␣Autumn␣Leaves.xml"
624 Reading [Ok]
625 Writing [Ok]
626
627 Number: 126
```

```
628 file: "../examples/Wikifonia/Juan Tizol - Perdido.xml"
629 Reading [Ok]
630 Writing [Ok]
631
632 Number: 127
633 file: "../examples/Wikifonia/Jule Styne - Time After Time.xml"
634 Reading [Ok]
635 Writing [Ok]
636
637 Number: 128
638 file: "../examples/Wikifonia/K. Dorham - Blue Bossa.xml"
639 Reading [Ok]
640 Writing [Ok]
641
642 Number: 129
643 file: "../examples/Wikifonia/Kaiser Chiefs - Ruby.xml"
644 Reading [Ok]
645 Writing [Ok]
646
647 Number: 130
648 file: "../examples/Wikifonia/Katrina and The Waves - Walking On Sunshine.xml"
649 Reading [Ok]
650 Writing [Ok]
651
652 Number: 131
653 file: "../examples/Wikifonia/Kenny Dorham - Lotus Blossom.xml"
654 Reading [Ok]
655 Writing [Ok]
656
657 Number: 132
658 file: "../examples/Wikifonia/Kris Wauters, Koen Wauters - Passie.xml"
659 Reading [Ok]
660 Writing [Ok]
661
662 Number: 133
663 file: "../examples/Wikifonia/L.Cohen - Hallelujah.xml"
664 Reading [Ok]
665 Writing [Ok]
666
667 Number: 134
668 file: "../examples/Wikifonia/Leon Russell - A Song For You.xml"
669 Reading [Ok]
670 Writing [Ok]
671
672 Number: 135
673 file: "../examples/Wikifonia/Leonard Cohen - Hallelujah.xml"
674 Reading [Ok]
675 Writing [Ok]
676
677 Number: 136
678 file: "../examples/Wikifonia/Lifehouse - You & Me.xml"
679 Reading [Ok]
680 Writing [Ok]
681
682 Number: 137
683 file: "../examples/Wikifonia/Lou Reed - Perfect Day.xml"
684 Reading [Ok]
685 Writing [Ok]
686
687 Number: 138
688 file: "../examples/Wikifonia/Luiz Bonfa - Black Orpheus.xml"
689 Reading [Ok]
```

```
690 Writing [Ok]
691
692 Number: 139
693 file: "../examples/Wikifonia/Mark Knopfler - Romeo and Juliet.xml"
694 Reading [Ok]
695 Writing [Ok]
696
697 Number: 140
698 file: "../examples/Wikifonia/Mars Bonfire - Born to be wild.xml"
699 Reading [Ok]
700 Writing [Ok]
701
702 Number: 141
703 file: "../examples/Wikifonia/Max Martin - ... Baby One More time.xml"
704 Reading [Ok]
705 Writing [Ok]
706
707 Number: 142
708 file: "../examples/Wikifonia/Mendez, Shakira - Underneath Your Clothes.xml"
709 Reading [Ok]
710 Writing [Ok]
711
712 Number: 143
713 file: "../examples/Wikifonia/Meredith Willson - Till there was you.xml"
714 Reading [Ok]
715 Writing [Ok]
716
717 Number: 144
718 file: "../examples/Wikifonia/Michael Bubl - Everything.xml"
719 Reading [Ok]
720 Writing [Ok]
721
722 Number: 145
723 file: "../examples/Wikifonia/Michael Jackson - Beat it!.xml"
724 Reading [Ok]
725 Writing [Ok]
726
727 Number: 146
728 file: "../examples/Wikifonia/Miles Davis - All Blues.xml"
729 Reading [Ok]
730 Writing [Ok]
731
732 Number: 147
733 file: "../examples/Wikifonia/Miles Davis - Four.xml"
734 Reading [Ok]
735 Writing [Ok]
736
737 Number: 148
738 file: "../examples/Wikifonia/Miles Davis - Solar.xml"
739 Reading [Ok]
740 Writing [Ok]
741
742 Number: 149
743 file: "../examples/Wikifonia/Miles Davis - Tune Up.xml"
744 Reading [Ok]
745 Writing [Ok]
746
747 Number: 150
748 file: "../examples/Wikifonia/Muse - Unintended.xml"
749 Reading [Ok]
750 Writing [Ok]
751
```

```
Number: 151
file: "../examples/Wikifonia/Nelly Furtado - All good things.xml"
Reading [Ok]
Writing [Ok]

Number: 152
file: "../examples/Wikifonia/Nick Cave - Into my Arms.xml"
Reading [Ok]
Writing [Ok]

Number: 153
file: "../examples/Wikifonia/Nickolas Ash - Ain't No Mountain High Enough.xml"
Reading [Ok]
Writing [Ok]

Number: 154
file: "../examples/Wikifonia/Noa - Beautiful that way.xml"
Reading [Ok]
Writing [Ok]

Number: 155
file: "../examples/Wikifonia/Norah Jones - Come away with me.xml"
Reading [Ok]
Writing [Ok]

Number: 156
file: "../examples/Wikifonia/Oasis - Wonderwall.xml"
Reading [Ok]
Writing [Ok]

Number: 157
file: "../examples/Wikifonia/Pat Metheny - Bachelors III.xml"
Reading [Ok]
Writing [Ok]

Number: 158
file: "../examples/Wikifonia/Paul Desmond - Take Five.xml"
Reading [Ok]
Writing [Ok]

Number: 159
file: "../examples/Wikifonia/Peter Gabriel - Steam.xml"
Reading [Ok]
Writing [Ok]

Number: 160
file: "../examples/Wikifonia/Phil Thornalley, Anne Previn, Scott Cutler - Torn.xml"
Reading [Ok]
Writing [Ok]

Number: 161
file: "../examples/Wikifonia/Procol Harum - A Whiter Shade of Pale.xml"
Reading [Ok]
Writing [Ok]

Number: 162
file: "../examples/Wikifonia/R. Kelly - I Believe I can Fly.xml"
Reading [Ok]
Writing [Ok]

Number: 163
file: "../examples/Wikifonia/REM - Everybody Hurts continued.xml"
```

```
814 Reading [Ok]
815 Writing [Ok]
816
817 Number: 164
818 file: "../examples/Wikifonia/Ray Charles - Hallelujah, I love her so.xml"
819 Reading [Ok]
820 Writing [Ok]
821
822 Number: 165
823 file: "../examples/Wikifonia/Ray Noble - Cherokee.xml"
824 Reading [Ok]
825 Writing [Ok]
826
827 Number: 166
828 file: "../examples/Wikifonia/Red Hot Chili Peppers - Under the Bridge.xml"
829 Reading [Ok]
830 Writing [Ok]
831
832 Number: 167
833 file: "../examples/Wikifonia/Richard Rodgers, Lorenz Hart - The Lady Is A Tramp.xml"
834 Reading [Ok]
835 Writing [Ok]
836
837 Number: 168
838 file: "../examples/Wikifonia/Richard Rodgers, Lorenz hart - My Funny Valentine.xml"
839 Reading [Ok]
840 Writing [Ok]
841
842 Number: 169
843 file: "../examples/Wikifonia/Richard Rodgers, Oscar Hammerstein - Edelweiss.xml"
844 Reading [Ok]
845 Writing [Ok]
846
847 Number: 170
848 file: "../examples/Wikifonia/Ricky Martin - Livin ' La Vlda Loca.xml"
849 Reading [Ok]
850 Writing [Ok]
851
852 Number: 171
853 file: "../examples/Wikifonia/Ritchie Valens - La Bamba.xml"
854 Reading [Ok]
855 Writing [Ok]
856
857 Number: 172
858 file: "../examples/Wikifonia/Robbie Williams, Guy Chambers - Angels.xml"
859 Reading [Ok]
860 Writing [Ok]
861
862 Number: 173
863 file: "../examples/Wikifonia/Robbie Williams, Guy Chambers - Feel.xml"
864 Reading [Ok]
865 Writing [Ok]
866
867 Number: 174
868 file: "../examples/Wikifonia/Robbie Williams, Guy Chambers - Let Me Entertain You.xml
        "
869 Reading [Ok]
870 Writing [Ok]
871
872 Number: 175
873 file: "../examples/Wikifonia/Robert Lopez, Jeff Marx - Avenue Q.xml"
874 Reading [Ok]
```

```
875 Writing [Ok]
876
877 Number: 176
878 file: "../examples/Wikifonia/Sade Adu, Ray St. John - Smooth Operator.xml"
879 Reading [Ok]
880 Writing [Ok]
881
882 Number: 177
883 file: "../examples/Wikifonia/Scott Auble - Scott's Tune.xml"
884 Reading [Ok]
885 Writing [Ok]
886
887 Number: 178
888 file: "../examples/Wikifonia/Sergei Rachmaninov, Eric Carmen - All By Myself.xml"
889 Reading [Ok]
890 Writing [Ok]
891
892 Number: 179
893 file: "../examples/Wikifonia/Seymour Simons - All Of Me.xml"
894 Reading [Ok]
895 Writing [Ok]
896
897 Number: 180
898 file: "../examples/Wikifonia/Sigmund Romberg, Oscar Hammerstein II - Softly as in a
      morning sunrise.xml"
899 Reading [Ok]
900 Writing [Ok]
901
902 Number: 181
903 file: "../examples/Wikifonia/Snow Patrol - Chasing Cars.xml"
904 Reading [Ok]
905 Writing [Ok]
906
907 Number: 182
908 file: "../examples/Wikifonia/Sonny Rollins - Pent Up House.xml"
909 Reading [Ok]
910 Writing [Ok]
911
912 Number: 183
913 file: "../examples/Wikifonia/Sonny Rollins - St. Thomas.xml"
914 Reading [Ok]
915 Writing [Ok]
916
917 Number: 184
918 file: "../examples/Wikifonia/Sonny Rollins - Tenor Madness.xml"
919 Reading [Ok]
920 Writing [Ok]
921
922 Number: 185
923 file: "../examples/Wikifonia/Sting - Every Little Thing She Does Is Magic.xml"
924 Reading [Ok]
925 Writing [Ok]
926
927 Number: 186
928 file: "../examples/Wikifonia/Sting - Roxanne.xml"
929 Reading [Ok]
930 Writing [Ok]
931
932 Number: 187
933 file: "../examples/Wikifonia/Sting, The Police - Every breath you take.xml"
934 Reading [Ok]
935 Writing [Ok]
```

```
936
937 Number: 188
938 file: "../examples/Wikifonia/Tadd␣Dameron␣-␣Lady␣Bird.xml"
939 Reading [Ok]
940 Writing [Ok]
941
942 Number: 189
943 file: "../examples/Wikifonia/The␣Fray␣-␣How␣to␣Save␣a␣Life.xml"
944 Reading [Ok]
945 Writing [Ok]
946
947 Number: 190
948 file: "../examples/Wikifonia/The␣Knife,␣Jose␣Gonzalez␣-␣Heartbeats.xml"
949 Reading [Ok]
950 Writing [Ok]
951
952 Number: 191
953 file: "../examples/Wikifonia/The␣Kooks␣-␣Naive.xml"
954 Reading [Ok]
955 Writing [Ok]
956
957 Number: 192
958 file: "../examples/Wikifonia/The␣Rolling␣Stones␣-␣Paint␣it␣black.xml"
959 Reading [Ok]
960 Writing [Ok]
961
962 Number: 193
963 file: "../examples/Wikifonia/Thelonious␣Monk␣-␣'Round␣Midnight.xml"
964 Reading [Ok]
965 Writing [Ok]
966
967 Number: 194
968 file: "../examples/Wikifonia/Thelonious␣Monk␣-␣Straight,␣No␣Chaser.xml"
969 Reading [Ok]
970 Writing [Ok]
971
972 Number: 195
973 file: "../examples/Wikifonia/They␣might␣be␣Giants␣-␣Experimental␣Film.xml"
974 Reading [Ok]
975 Writing [Ok]
976
977 Number: 196
978 file: "../examples/Wikifonia/Thomas␣Waller␣-␣Honeysuckle␣Rose.xml"
979 Reading [Ok]
980 Writing [Ok]
981
982 Number: 197
983 file: "../examples/Wikifonia/Thomas␣Yorke␣-␣Karma␣Police.xml"
984 Reading [Ok]
985 Writing [Ok]
986
987 Number: 198
988 file: "../examples/Wikifonia/Toots␣Thielemans␣-␣Bluesette.xml"
989 Reading [Ok]
990 Writing [Ok]
991
992 Number: 199
993 file: "../examples/Wikifonia/Toto␣-␣Africa.xml"
994 Reading [Ok]
995 Writing [Ok]
996
997 Number: 200
```

```
file: "../examples/Wikifonia/Toto - Hold the line.xml"
Reading [Ok]
Writing [Ok]

Number: 201
file: "../examples/Wikifonia/Tracy Chapman - The Wedding Song.xml"
Reading [Ok]
Writing [Ok]

Number: 202
file: "../examples/Wikifonia/Traditional - Dag Sinterklaasje.xml"
Reading [Ok]
Writing [Ok]

Number: 203
file: "../examples/Wikifonia/Traditional - Angels We Have Heard On High.xml"
Reading [Ok]
Writing [Ok]

Number: 204
file: "../examples/Wikifonia/Traditional - De zak van Sinterklaas.xml"
Reading [Ok]
Writing [Ok]

Number: 205
file: "../examples/Wikifonia/Traditional - O kom er eens kijken.xml"
Reading [Ok]
Writing [Ok]

Number: 206
file: "../examples/Wikifonia/Traditional - Sinterklaas is jarig.xml"
Reading [Ok]
Writing [Ok]

Number: 207
file: "../examples/Wikifonia/Traditional - Sinterklaas kapoentje.xml"
Reading [Ok]
Writing [Ok]

Number: 208
file: "../examples/Wikifonia/Traditional - Sinterklaasje bonne bonne bonne.xml"
Reading [Ok]
Writing [Ok]

Number: 209
file: "../examples/Wikifonia/Traditional - Sinterklaasje kom maar binnen met je
    knecht.xml"
Reading [Ok]
Writing [Ok]

Number: 210
file: "../examples/Wikifonia/Traditional - We wish you a merry christmas.xml"
Reading [Ok]
Writing [Ok]

Number: 211
file: "../examples/Wikifonia/Traditional - What child is this_.xml"
Reading [Ok]
Writing [Ok]

Number: 212
file: "../examples/Wikifonia/Traditional - Zachtjes gaan de paardenvoetjes.xml"
```

```
1059 Reading [Ok]
1060 Writing [Ok]
1061
1062 Number: 213
1063 file: "../examples/Wikifonia/Traditional - Zie ginds komt de stoomboot.xml"
1064 Reading [Ok]
1065 Writing [Ok]
1066
1067 Number: 214
1068 file: "../examples/Wikifonia/Trent Reznor - Hurt.xml"
1069 Reading [Ok]
1070 Writing [Ok]
1071
1072 Number: 215
1073 file: "../examples/Wikifonia/U2 - With or without you.xml"
1074 Reading [Ok]
1075 Writing [Ok]
1076
1077 Number: 216
1078 file: "../examples/Wikifonia/Village People - Y.M.C.A.xml"
1079 Reading [Ok]
1080 Writing [Ok]
1081
1082 Number: 217
1083 file: "../examples/Wikifonia/Walter Afanasieff , Mariah Carey - Hero.xml"
1084 Reading [Ok]
1085 Writing [Ok]
1086
1087 Number: 218
1088 file: "../examples/Wikifonia/Wayne Thomps - Always On My Mind.xml"
1089 Reading [Ok]
1090 Writing [Ok]
1091
1092 Number: 219
1093 file: "../examples/Wikifonia/Willem Vermandere - Blanche en zijn peird.xml"
1094 Reading [Ok]
1095 Writing [Ok]
1096
1097 Number: 220
1098 file: "../examples/Wikifonia/Zombies - Time of the Season.xml"
1099 Reading [Ok]
1100 Writing [Ok]
1101
1102 Number: 221
1103 file: "../examples/Wikifonia/anonymous - Wilhelmus van Nassouwe.xml"
1104 Reading [Ok]
1105 Writing [Ok]
```

B.1.4 Gutenberg: Output

Apresenta-se o ficheiro gerado com o resultado da execução do caso de estudo Gutenberg.

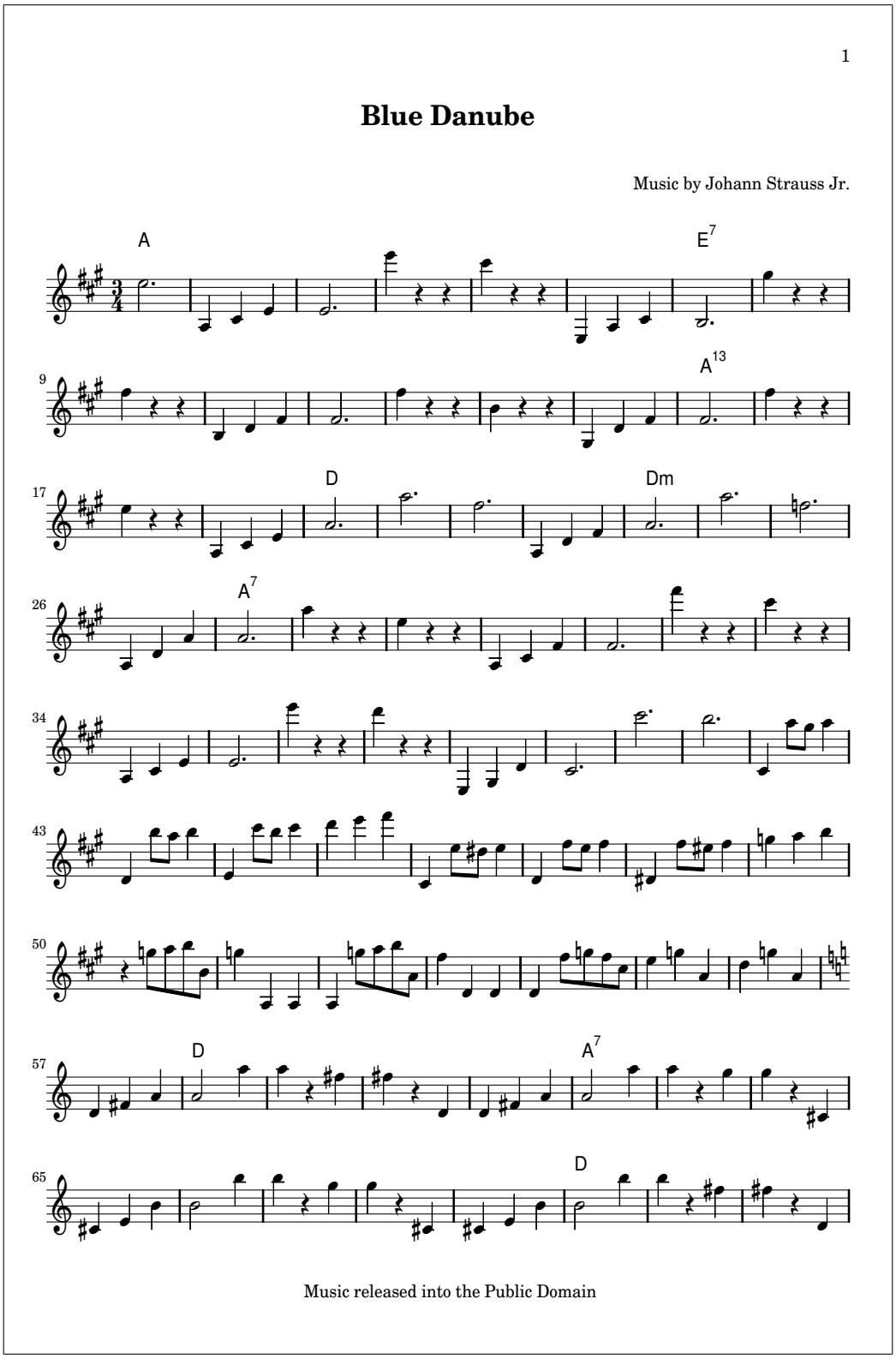
Listagem B.4: Output do caso de estudo Gutenberg

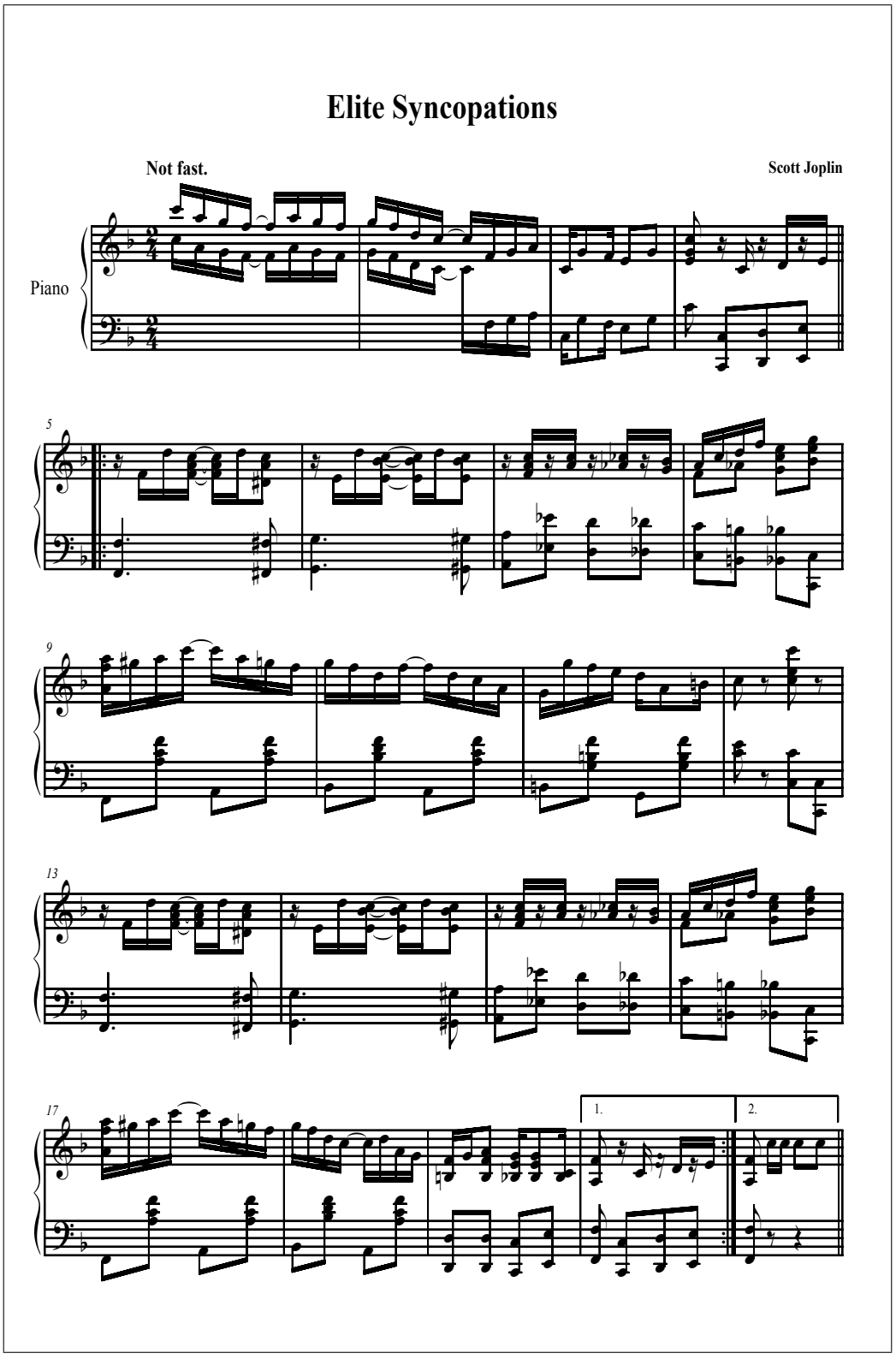
```
1
2 Number: 1
3 file: "../examples/Gutenberg/11755-Complete.utf8.xml"
```

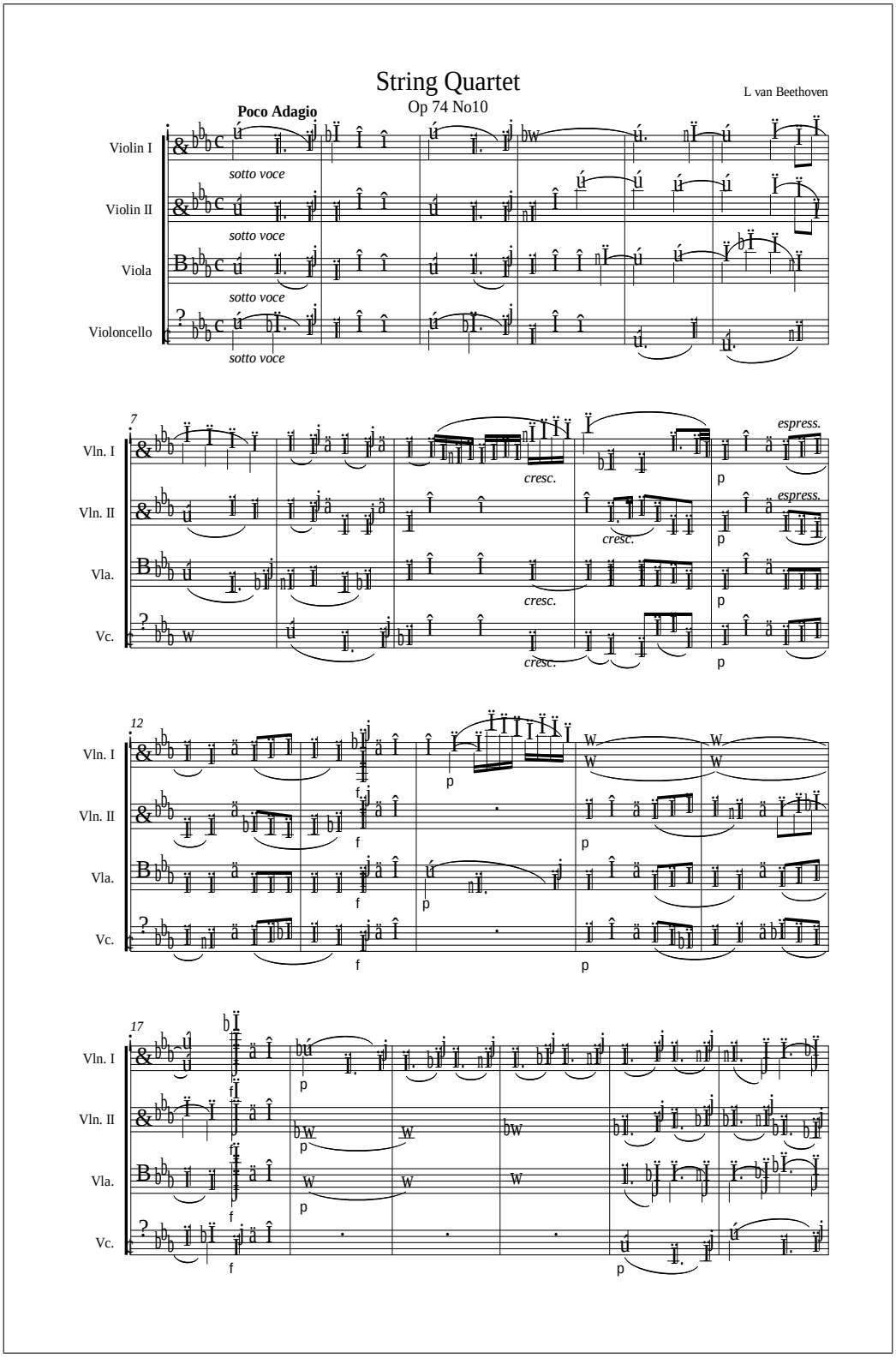
```
4  Reading [Ok]
5  Writing [Ok]
6
7  Number: 2
8  file: "../examples/Gutenberg/11001-Complete.utf8.xml"
9  Reading [Ok]
10 Writing [Ok]
11
12 Number: 3
13 file: "../examples/Gutenberg/12149-Complete.utf8.xml"
14 Reading [Ok]
15 Writing [Ok]
16
17 Number: 4
18 file: "../examples/Gutenberg/12695-complete.utf8.xml"
19 Reading [Ok]
20 Writing [Ok]
21
22 Number: 5
23 file: "../examples/Gutenberg/13473-all.utf8.xml"
24 Reading [Ok]
25 Writing [Ok]
```

B.2 Primeiras páginas de documentos MusicXML

Nesta secção são apresentadas as primeiras páginas dos documentos selecionados para uma análise mais detalhada sobre a escalabilidade da biblioteca MusicXML. O primeiro documento é a transcrição de uma música popular japonesa, que inclui a letra. A segunda é a obra Blue Danube de Strauss. O terceiro documento é o Elite Syncopations com quatro páginas, onde apenas se apresenta a primeira. O quarto documento é um extrato do Prelúdio para a Tragédia, contendo apenas quatro páginas. O quinto documento contém 40 páginas sendo um quarteto de cordas, tal como o sexto documento que contém 47 páginas.


1
Blue Danube
Music by Johann Strauss Jr.
A
E7
A13
D
Dm
A7
D
A7
D
Music released into the Public Domain


Elite Syncopations
Not fast.
Scott Joplin
Piano
5
9
13
17
1.
2.


String Quartet
Op 74 No10
L van Beethoven
Poco Adagio
Violin I
Violin II
Viola
Violoncello
sotto voce
7
Vln. I
Vln. II
Vla.
Vc.
cresc.
espress.
p
12
f
17

Índice