Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-3661 Taller de Lenguajes de Programación

Proyecto 1: Generación y recuperación de secuencias musicales en base a modelos de contexto

1. Requisitos preliminares

Este proyecto debe ser desarrollado en lenguaje Haskell para el compilador *ghc*. Este compilador puede ser instalado en Ubuntu con el comando

```
apt-get install ghc
```

Adicionalmente, el presente proyecto requiere la biblioteca Euterpea, un lenguaje de representación de música en alto nivel basado en Haskell. Siga las instrucciones en `http://haskell.cs.yale.edu/euterpea/download/` para instalarla.

Por último, para la lectura de archivos MusicXML es necesario instalar las bibliotecas *HaXml* y *directory*, y para la generación de números aleatorios se requiere la biblioteca *random*. Puede instalar estas bibliotecas ejecutando el comando:

```
cabal install directory random HaXml
```

Para poder reproducir las secuencias musicales, es necesario que usted instale un sintetizador en su sistema. En Ubuntu, se recomienda que usted instale *qsynth*, mediante el comando:

```
apt-get install qsynth
```

Para que qsynth pueda producir sonidos es necesario haber cargado un *SoundFont*. En `http://ldc.usb.ve/~cgomez/lenguajes/` está disponible un SoundFont que usted puede utilizar con *qsynth*.

2. Introducción

Una tarea común en la actualidad consiste en utilizar algoritmos para componer automáticamente piezas musicales. Un enfoque particular a este problema es la composición automática basada en modelos estadísticos. Bajo este enfoque, la computadora analiza un cuerpo numeroso de ejemplos musicales para extraer de él cierta información estadística, como cuál es la frecuencia (el número de apariciones) de cada nota, y cuál es la frecuencia de cada par de notas consecutivas.

Esta información computada a partir del análisis del cuerpo musical constituye un *modelo*. Este modelo puede ser luego empleado para generar nuevas piezas musicales, que idealmente deberían reflejar el estilo de música de la colección original. Este es el esquema general de la composición musical en base a modelos estadísticos, el cual será ampliado en las secciones que siguen.

Los modelos estadísticos que han sido descritos no sólo pueden usarse para la generación, sino también para la clasificación de música. Por ejemplo, en general no es trivial definir una medida de distancia entre dos melodías A y B. Sin embargo, una estrategia para resolver este problema consiste en obtener un modelo de A y un modelo de B y computar la distancia entre ellos. De esta manera, la distancia entre las dos melodías no se calcula directamente, sino como la distancia entre los dos modelos. De este modo, los principios enunciados en el párrafo anterior pueden ser usados no sólo para generar melodías nuevas, sino para computar el grado de similitud entre dos melodías.

En este proyecto se requiere que usted programe un algoritmo para extraer un modelo estadístico de una melodía dada. Este algoritmo será usado por usted para analizar un cuerpo numeroso de ejemplos musicales, y obtener de él un modelo estadístico, que será utilizado para generar melodías nuevas. Igualmente, los modelos estadísticos serán usados para computar la distancia entre dos melodías.

3. Secuencias musicales

Para continuar, es necesario presentar parte de la notación que será usada en este texto. El término $\overline{e_n}$ se utiliza como abreviatura para la secuencia $(e_1, \ldots, e_n)$. El *espacio de eventos* es el conjunto de todos los eventos representables, y se denota por el símbolo ξ. El conjunto de todas las secuencias

que se pueden construir a partir de elementos de un conjunto S se denota S^*. La concatenación de un evento e a una secuencia c se denota por $c :: e$.

En Música, una melodía se define como una sucesión de sonidos con altura y duración definidas. Las melodías también pueden incluir silencios, es decir, intervalos de tiempo en los que no se produce ningún sonido, sin embargo, por simplicidad en este proyecto las melodías no podrán contener silencios.

Una melodía será representada en este proyecto como una secuencia de eventos, donde cada evento es un par $(altura, duración)$. La altura se representará como un número entero, al igual que la duración.

```
type Evento = (Int, Int)
```

Cada nota musical tiene una altura específica denotada por su nombre y octava. Por ejemplo, el do central del piano se denota *do 4*, ya que es el do de la cuarta octava. En este proyecto, se utilizará la notación MIDI para las alturas, en la cual el do central se representa mediante el número 60. En esta representación los intervalos se cuentan en semitonos, por lo tanto, el re por encima del do central se representa mediante el número 62, el mi por 64, el fa por 65 y así sucesivamente.

Cada nota tiene también una duración específica. Dentro de este proyecto cada evento tendrá una duración expresada como un número entero, donde la figura de negra será equivalente a la duración 4, la corchea 2, la blanca 8, y así sucesivamente.

4. Modelos de contexto

Los *modelos de contexto* son un formalismo empleado para el modelado estadístico de secuencias. Los modelos de contexto se pueden motivar de la siguiente manera. Suponga que se tiene una secuencia de caracteres como sigue:

ABCABCBA

Una manera de modelar estadísticamente esta secuencia es computar la frecuencia (el número de apariciones) de cada evento. Dichas frecuencias se presentan en la tabla 1. Note que la secuencia vacía () es un contexto válido, y su frecuencia es igual a la longitud de la secuencia.

El problema con este método sencillo es que no reproduce el orden entre los eventos de la secuencia original. Por ejemplo, la siguiente secuencia tiene

():8	(A):3 (B):3 (C):2

Cuadro 1: Distribución de frecuencias de la secuencia ABCABCBA.

():8	(A):3 (B):3 (C):2	(AB):2 (BC):2 (BA):1 (CA):1 (CB):1

Cuadro 2: Distribución de frecuencias de la secuencia ABCABCBA, orden 1.

la misma distribución de frecuencias, pero mantiene poca similitud con la original:

CCBBBAAA

El método anterior se puede mejorar representando también la frecuencia de cada par de eventos consecutivos. Estas frecuencias, junto con las frecuencias de la tabla 1, se presentan en la tabla 2.

Esta distribución contiene más información acerca de la secuencia original, ya que toma en cuenta el orden de sucesión entre los eventos.

Un *modelo de contexto* es un mapa asociativo que asocia a cada tupla $c :: e$ un número entero que representa su frecuencia de aparición. El *orden* del modelo es la longitud máxima del contexto. El primer modelo era de orden 0, mientras que el segundo, de orden 1. Se pueden construir modelos de contexto de orden superior siguiendo el mismo procedimiento, mas por simplicidad esto no se hará en el proyecto.

Dado un modelo M, la probabilidad $p_M(e|c)$ de un evento e dado un contexto c se puede calcular como el número de apariciones de la secuencia $c :: e$ en la base de datos entre el número de apariciones del contexto c. El problema con este método es que el contexto podría no aparecer en la base de datos, en cuyo caso la probabilidad estaría indefinida. Para resolver este problema, la probabilidad $p_M(e_n|e_1 \dots e_{n-1})$ se calcula como la combinación lineal:

$$0{,}3 \cdot p_M(e_n|()) + 0{,}7 \cdot p_M(e_n|e_{n-1})$$

De esta manera, la probabilidad será siempre una cantidad definida.

5. Generación de secuencias

Una vez que se tiene un modelo de contexto, es posible utilizarlo de forma generativa para crear nuevas secuencias musicales. Esto se realiza con un procedimiento recursivo de izquierda a derecha, en otras palabras, primero se genera el primer evento, cuyo contexto es la secuencia vacía (). Seguidamente se genera el segundo evento, cuyo contexto es el primer evento generado. Este proceso continúa hasta alcanzar una longitud tope para la secuencia generada.

La elección del evento a generar en cada paso se realiza de manera exhaustiva. Es decir, dado un contexto c, para determinar el evento sucesor e se calcula la probabilidad $p(e^j|c)$ para cada posible evento e^j que pertenece al espacio de eventos. Esto resulta en una distribución de probabilidad sobre el espacio de eventos ξ, para un contexto c.

Recuerde que una propiedad de toda distribución de probabilidad es que la suma de las probabilidades de sus eventos debe ser igual a 1. Para que el vector de probabilidades obtenido en el paso anterior sea una distribución de probabilidad válida, generalmente es necesario normalizarlo de forma que la suma de sus elementos sea igual a 1.

El evento sucesor e se obtiene utilizando el algoritmo de selección de la ruleta sobre esta distribución de probabilidad.

Sea $e^1 \cdots e^m$ un conjunto de eventos y $p^1 \dots p^m$ sus probabilidades asociadas, el algoritmo de selección de la ruleta permite obtener un evento e^j al azar de acuerdo con esta distribución de probabilidad. El algoritmo consiste en dividir un segmento de recta unitario en m subsegmentos cada uno del tamaño p^j. Luego, se genera un número aleatorio r con distribución uniforme en el intervalo $[0, 1)$. El valor de retorno del algoritmo es el evento e^j correspondiente al subsegmento de recta en la posición r. Este algoritmo garantiza que cada evento e^j será elegido con probabilidad p^j.

6. Distancia entre secuencias musicales

Un mapa asociativo A es una estructura de datos que permite almacenar un conjunto de pares (k_i, v_i) donde cada k_i se denomina una clave, y v_i se denomina el valor asociado a k_i. Sea A un mapa asociativo, el conjunto de claves de A se denomina claves_A. La función $\text{valor}_A(k_i)$ devuelve el valor asociado a la clave k_i en el mapa A.

En el presente proyecto, un modelo de contexto puede ser representado como un mapa asociativo de ξ^* a $\mathcal{N}$. Sean A y B dos modelos de contexto representados de este modo, la distancia entre A y B se define como:

$$\text{distancia}(A, B) = \sqrt{\sum_{k_i \in \text{claves}_A \cup \text{claves}_B} \left(\text{valor}^*_B(k_i) - \text{valor}^*_A(k_i)\right)^2}$$

donde la función valor^*_A se define como

$$\text{valor}^*_A(k_i) = \begin{cases} \text{valor}_A(k_i) & \text{si } k_i \in A \\ 0 & \text{si } k_i \notin A \\ 0 & \text{si } k_i = () \end{cases}$$

A partir de estos conceptos es posible representar la distancia entre dos secuencias musicales $\overline{s_1}$ y $\overline{s_2}$. Esta distancia se define como la distancia entre los modelos de contexto construidos a partir de cada una de las secuencias individuales.

7. Colección musical

Para el desarrollo de este proyecto usted dispone de una colección de corales de Bach publicadas en la dirección `http://ldc.usb.ve/~cgomez/lenguajes/`. Los corales de Bach se encuentran en formato MusicXML.

El módulo Input, publicado en la misma dirección, dispone de funciones que permiten leer archivos en formato MusicXML. La función *loadMusicXmls* toma como parámetro la trayectoria de un directorio y devuelve un par `([[Evento]], [String])` que representa las secuencias musicales leídas y sus nombres de archivo respectivos.

El espacio de eventos para los corales de Bach se define como sigue, donde A es el conjunto de todas las alturas válidas y D el conjunto de todas las duraciones válidas.

$$A = \{60, \ldots, 69\}$$
$$B = \{1, 2, 3, 4, 6, 8, 12, 16\}$$
$$\xi = A \times B$$

Note que las alturas están limitadas al rango de una octava y medio por encima del do central, es decir, de la altura 60 a la 69. Igualmente, no se consideran todas las duraciones posibles de la 1 a la 16, sino solamente las más comunes. Esto se hace para reducir el tamaño del espacio de eventos.

8. Requerimientos

En `http://haskell.cs.yale.edu/euterpea/download/` está disponible el esqueleto del módulo Main que deberá tener su programa. Este esqueleto se copia a continuación:

```
module Main where

import Prelude hiding (init)
import Input
import Euterpea hiding (Event)
import Data.List
import Data.Function

-- Directorio predeterminado
directorio :: String
directorio = "./xml/"

-- Longitud de las secuencias musicales generadas
longitud :: Int
longitud = 50

{- Induce un modelo de contexto a partir de la colección musical
   en el directorio por defecto, genera una secuencia musical
   nueva a partir de este modelo, la imprime por pantalla y la
   reproduce.
   -}
componer :: IO ()
componer = componer' directorio

componer' :: String -> IO ()
```

```
componer' dir = do
  (seqs, filenames) <- loadMusicXmls dir
  -- let modelo = ...
  -- let composicion = ...
  putStrLn $ show composicion
  play $ sequenceToMusic composicion

{- Recupera las diez secuencias más similares a la k-ésima secuencia
   de la colección musical en el directorio por defecto, donde la
   colección musical ha sido ordenada en orden alfabético por el
   nombre de archivo. Imprime una lista ordenada de las diez
   secuencias más similares. En cada fila de la lista se debe indicar
   el número de la secuencia (relativo al orden alfabético de la
   colección), el nombre de archivo y la distancia a la consulta.
   -}
buscar :: Int -> IO ()
buscar = buscar' directorio

buscar' :: String -> Int -> IO ()
buscar' dir = do
  seqfns <- loadMusicXmls dir
  let seqfns_ordenados = unzip $ sortBy (compare `on` snd) $ zip seqfns
  -- ...

tocar :: Int -> IO ()
tocar n = do
  seqfns <- loadMusicXmls directorio
  let (seqs, filenames) = unzip $ sortBy (compare `on` snd) $ (uncurry zip) seqfns
  if (n > 0) && (n <= length seqs) then
    putStrLn (filenames !! (n-1)) >>
    play (sequenceToMusic (seqs !! (n-1)))
    else
      putStrLn "Indice fuera de rango"

eventToNote :: Evento -> Music Note1
eventToNote e = note
  where
  d = (fromIntegral $ snd e) / 16
  p = Euterpea.pitch $ fst e
  note = Prim (Note d (p,[]))

sequenceToMusic :: [Evento] -> Music Note1
sequenceToMusic es = line $ map eventToNote es
```

La función *main* induce un modelo de contexto a partir de una colección de ejemplos musicales y genera una secuencia musical nueva a partir de este modelo. Seguidamente, imprime la secuencia resultante y la reproduce por medio de la función *play* de Euterpea. La longitud de la secuencia generada debe ser igual al valor de la función *longitud* que figura al comienzo del archivo. Usted debe completar la definición de la función *componer'*.

La función *buscar* recibe un entero que corresponde a la posición de una secuencia en la colección de ejemplos, estando esta colección en orden alfabético por el nombre de archivo. Esta función imprime una lista ordenada de las diez secuencias de la colección con menor distancia a la consulta (incluyendo la consulta, que debería aparecer en la primera posición). En cada fila de la lista se debe imprimir el número de la secuencia del resultado, su nombre de archivo y la distancia a la consulta. Usted debe completar la definición de la función *buscar'*. A continuación se proporciona un ejemplo de la salida que debe producir la función buscar:

```
37    015408B_.xml    23.2
4     024835B3.xml    25.6
68    024833B3.xml    32.3
20    015705B_.xml    48.9
23    006707B_.xml    54.5
79    036400B_.xml    72.8
4     033700B_.xml    96.3
12    031000B_.xml    125.8
67    033200B_.xml    150.7
90    040600B_.xml    198.3
```

Usted no debe modificar la firma de ninguna de las funciones facilitadas en el esqueleto.

9. Entrega

Usted debe entregar todos los archivos fuente de su programa en el directorio raíz que entregue. Incluya en este directorio un *Makefile* que permita compilar su programa. El nombre del ejecutable generado por el *Makefile* debe ser `music` (con todas las letras en minúscula).

Usted sólo debe entregar los archivos fuente y el *Makefile*, no incluya los archivos MusicXml.

Empaquete el código fuente en el archivo `p1gXX.tar.gz`, donde XX corresponde con su número de grupo en Aula Virtual. Los archivos fuente deben estar al nivel superior del archivo *tar.gz*, no debe haber un directorio de por medio.

Usted debe subir el *tar.gz* a la sección de Documentos de su grupo en Aula Virtual a más tardar el lunes 9 de diciembre (lunes de la semana 7) a las 11:00 pm. Adicionalmente, debe entregar el código impreso a su profesor de taller a más tardar el jueves de la misma semana. La entrega del código fuente impreso es un requisito para que su proyecto sea corregido.