HARMONY: Sistema funcional de ayuda a la composición musical

Jose Emilio Labra Gayo
Luis Ángel Oliveira Rodríguez
Juan Manuel Cueva Lovelle

Departamento de Informática
Universidad de Oviedo
C/ Calvo Sotelo S/N, 33007
Oviedo, España
e-mail: labra@pinon.ccu.uniovi.es
Tlfno: +34-85103394

Resumen

Es este artículo se describe HARMONY: un sistema que integra las fases que intervienen en la creación de una composición musical. El sistema contempla la generación automática de melodías mediante fractales y algoritmos genéticos, la armonización de las melodías generadas, la asignación de instrumentos a cada voz y la ejecución en formato MIDI. El núcleo del sistema, desarrollado en lenguaje *Haskell*, permite armonizar melodías según las reglas de la teoría musical clásica.

HARMONY: Sistema funcional de ayuda a la composición musical

Resumen

Es este artículo se describe HARMONY: un sistema que integra las fases que intervienen en la creación de una composición musical. El sistema contempla la generación automática de melodías mediante fractales y algoritmos genéticos, la armonización de las melodías generadas, la asignación de instrumentos a cada voz y la ejecución en formato MIDI. El núcleo del sistema, desarrollado en lenguaje *Haskell*, permite armonizar melodías según las reglas de la teoría musical clásica.

1 Introducción

La utilización de computadoras como una herramienta útil para el desarrollo artístico está alcanzando una importancia considerable. Las facilidades ofrecidas para la creación de gráficos han permitido poner al alcance de usuarios no avanzados técnicas que les permiten crear composiciones de alto nivel. En el campo de la composición musical surgen herramientas que permiten al usuario componer música a partir de diversos algoritmos [1]. El objetivo del sistema HARMONY es integrar estos algoritmos en un sistema que permita proporcionar técnicas de alto nivel musical como la armonización de melodías a usuarios no avanzados.

La disciplina de armonía [7], [8], [18], [26] se estudia en últimos cursos de carreras musicales debido a su complejidad. Mediante la automatización del proceso de armonizar una melodía, se facilita la tarea compositiva permitiendo que el usuario se concentre en tareas más creativa. El sistema no pretende obligar al compositor a seguir unas determinadas reglas de armonía. Por el contrario, lo que se pretende es proporcionar al usuario las técnicas necesarias para que sea capaz de crear música siguiendo unas directrices armónicas determinadas. De esta forma, un requisito fundamental del sistema es permitir parametrizar las reglas a utilizar en la creación de acordes dando una mayor libertad de acción al compositor.

El sistema HARMONY integra las fases que intervienen en la creación de una composición musical. La implementación del núcleo del sistema se ha desarrollado en *Haskell* [19], un lenguaje puramente funcional con semántica no estricta. Los lenguajes funcionales ofrecen grandes posibilidades para la descripción de composiciones musicales debido a su carácter declarativo [6], [11], [16]. La evaluación perezosa permite definir composiciones conceptualmente infinitas y las funciones de orden superior aumentan la capacidad de abstracción del compositor. Con este sistema se puede comprobar la potencia descriptiva de los lenguajes funcionales que pueden utilizarse como medio de comunicación de ideas musicales de alto nivel.

2 Descripción del Sistema

En la figura 1 se pueden observar los distintos módulos que integran el sistema.

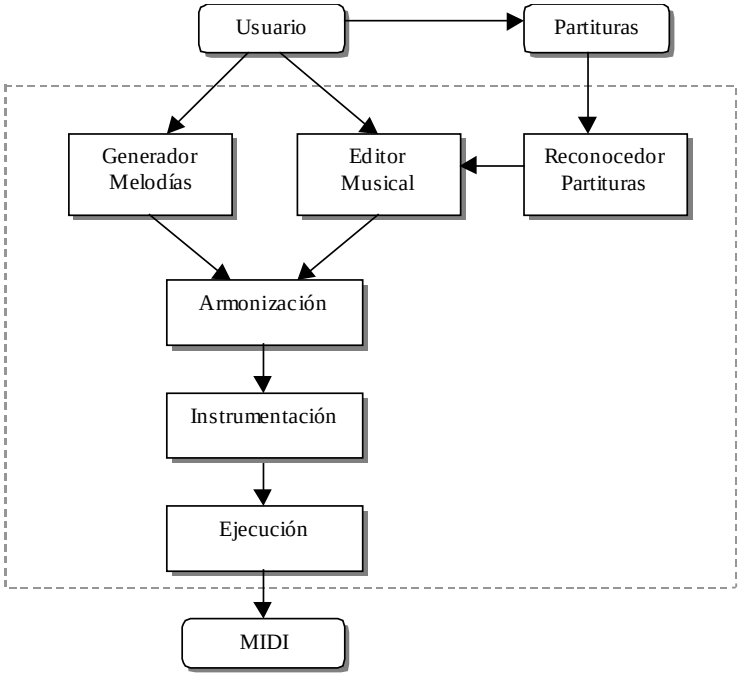


Figura 1: Esquema general del Sistema

a.- Generador de Melodías
Para la generación de melodías se desarrollaron las siguientes técnicas:

- **Fractales**: Se han implementado una serie de algoritmos de generación de señales aleatorias como la simulación del movimiento de Brown mediante cortes aleatorios o la utilización de sistemas de Lindenmayer. Los valores de la función de salida se asocian a las diversas tonalidades, generando melodías con diferentes grados de aleatoriedad.

- **Algoritmos genéticos**: La mayor dificultad en la utilización de algoritmos genéticos para la generación de composiciones musicales es la elección de la función de ajuste [2]. En [3] se describe la utilización de una función de ajuste que valora el juicio del usuario mediante redes neuronales. Actualmente se está desarrollando este módulo considerando como función de ajuste, el grado de seguimiento de las reglas clásicas de armonía implementadas en este sistema.

b.-Reconocedor de partituras
Permite reconocer partituras externas escaneadas[1]. Se han utilizado técnicas tradicionales de reconocimiento de caracteres, especializadas al tratamiento de signos musicales.

c.- Edición musical
Se está desarrollando un módulo[2] de edición de partituras que permita mostrar en pantalla las melodías generadas, las composiciones introducidas de forma directa por el usuario, las composiciones externas (partituras reconocidas) o los resultados de la armonización.

d.- Armonización

[1] Este módulo ha sido desarrollado en C++ [9]
[2] La primera versión del módulo se había realizado en Delphi y C++. Actualmente, se está construyendo una nueva versión en Haskell

El núcleo del sistema, desarrollado completamente en *Haskell*, consiste en tomar como entrada una melodía y generar un acompañamiento melódico basado en acordes siguiendo las normas de la armonía tradicional. En la siguiente sección se describirá este módulo en mayor profundidad.

e.- Instrumentación

Consiste en asignar un instrumento a cada voz de la composición. Aunque conceptualmente, las etapas de armonización e instrumentación son diferentes, a la hora de contemplar ciertas formas musicales es conveniente revisar los tipos de instrumentos para los cuales se está generando la armonía, ya que no todos alcanzan las mismas posibilidades.

f.- Ejecución

La ejecución de una composición musical consiste en la transformación de la notación simbólica en un formato reconocible por la computadora.

La separación entre las fases de armonización, instrumentación y ejecución persigue un mayor acercamiento al modo de funcionamiento humano. Así, las dos primeras fases se centrarían en la composición de la obra, mientras que la tercera fase se ocupa de su interpretación. Esta separación permitirá que una misma composición pueda ser interpretada con diferentes matices. La implementación actual resuelve las dos últimas fases mediante un simple traductor a lenguaje MIDI que asocia los instrumentos seleccionados por el usuario a cada voz.

3 Armonización

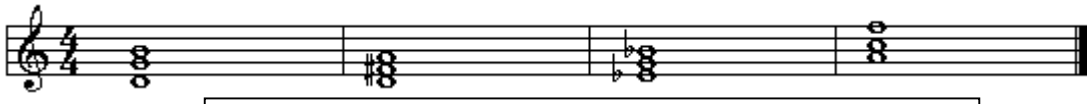
Como ya se ha indicado, el núcleo del sistema HARMONY consiste en la fase de armonización. Esta fase se ha desarrollado íntegramente en lenguaje *Haskell* y utiliza un lenguaje funcional intermedio para la definición de composiciones musicales.

La notación intermedia utilizada para definir composiciones musicales utiliza una sintaxis similar a las listas del lenguaje *Haskell*.

```
NombreNota ::= Do |Re |Mi |Fa |Sol |La |Si
Alteracion ::= N|B|S|D|X -- Normal,Bemol,Sostenido,Doble Bemol,Doble Sostenido
Octava ::= Integer
Duracion ::= Float
Nota ::= (NombreNota, Alteracion, Octava, Duracion)
Acorde ::= [Nota,...]
Melodia ::= [Nota,...]
ComposicionArmonica    ::= [Acorde,...]
ComposicionContrapunto ::= [Melodia,...]
```

Figura 2: Resumen de Notación simbólica empleada

Se consideran dos visiones de una composición: la visión armónica (una composición se considera una sucesión de acordes) y la visión contrapuntística (una composición se considera una lista de melodías). Para la representación de acordes, se considerará, además, la tonalidad y el primer y segundo intervalo del acorde.



```
comp=[[(Re,N,1,4.0),(Sol,N,1,4.0),(Si,N,1,4.0)],
      [(Re,N,1,4.0),(Fa,S,1,4.0),(La,N,1,4.0)],
      [(Mi,B,1,4.0),(Sol,N,1,4.0),(Si,B,1,4.0)]
      [(La,N,1,4.0),(Do,N,2,4.0),(Fa,N,2,4.0)]]
```

Figura 3: Ejemplo de composición y notación simbólica correspondiente

La notación simbólica se está modificando con el fin de adoptar el sistema *Haskore* [11] como lenguaje musical. Éste sistema permite definir composiciones musicales de alto nivel dentro del sistema Haskell. De esta forma, se pueden declarar composiciones parametrizadas, repeticiones, recursividad, evaluación perezosa, etc. Se pretende facilitar un subconjunto del lenguaje *Haskell* especializado para descripciones musicales como lenguaje de *Scripting*. De esta forma, se ofrece al usuario una herramienta descriptiva de alto nivel que proporciona una mayor normalización y evita ambigüedades.

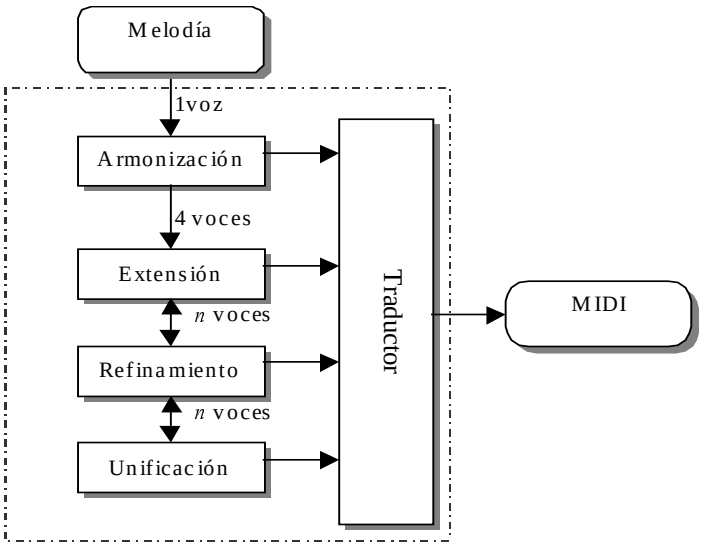


Figura 4: Núcleo del Sistema HARMONY

La entrada de la fase de armonización es una melodía de una voz. Esta melodía puede obtenerse a partir de una composición musical existente, mediante el editor musical o generada automáticamente. A partir de ella, las operaciones fundamentales son:

- **Armonización**: A partir de una melodía en el formato descrito, se extraen las notas base y se forma sobre cada una de ellas el acorde correspondiente según las reglas contempladas. La sucesión de acordes cumplirá ciertos requisitos como la no existencia de quintas y octavas ilegales, la construcción correcta de intervalos, los cruzamientos entre voces, las tesituras, etc. El usuario puede controlar la actuación del sistema mediante la especificación de ciertos parámetros como la tonalidad, las funciones de ponderación (para indicar qué voz se desea mover o qué voz se desea mantener fija), el tipo de posibles representaciones de un acorde o incluso el propio algoritmo de armonización.

- **Extensión a *n* voces**: Puesto que la armonización inicial se realiza a 4 voces, esta operación permite la ampliación a un número arbitrario de voces. Estas voces podrán refinarse posteriormente para ganar mayor riqueza musical.

- **Refinamiento**: Una vez creada una armonización, el resultado puede refinarse mediante la inclusión de notas de paso, floreos o combinaciones de ambos.

- **Unificación**: Dado que con los procesos anteriores se pueden perder excesivamente las melodías en el entramado armónico, esta operación permite la recuperación de las mismas. El proceso consiste en analizar las sucesivas notas dentro de una misma voz, uniendo las que sean iguales y preparando la composición para una visión melódica o contrapuntística.

La notación interna utilizada entre cada una de las operaciones son ficheros de texto con el formato indicado. Con el fin de permitir depurar el control sobre el proceso, se ha implementado un traductor de dicha notación al formato MIDI mediante combinadores de analizadores sintácticos [13].

4 Trabajos Relacionados

Existe una gran cantidad de trabajos que utilizan fractales o algoritmos genéticos para la creación musical [10], [15], [23], [25]. La mayoría de estos sistemas generan melodías extrañas y son escasos los que permiten al usuario seleccionar una melodía inicial y armonizarla según las reglas de la armonía clásica. Estos sistemas están implementados mediante técnicas de programación tradicionales y llama la atención la exigua utilización de lenguajes puramente funcionales en el desarrollo de este tipo de algoritmos. Sólo en [24] se describe una implementación general de algoritmos genéticos en PolyP [14], una extensión de *Haskell* que admite la definición de funciones *politípicas*. Sin embargo, en dicho trabajo, los ejemplos presentados se aplican a la generación de planes e imágenes.

Harmony surgió de forma paralela al sistema Haskore [11]. En dicho trabajo se definen una serie de módulos *Haskell* que permiten expresar estructuras musicales de alto nivel. La utilización del propio lenguaje *Haskell* como lenguaje de descripción musical se contemplaba ya en la primera versión del sistema Harmony (las diferentes notaciones no eran más que listas de notas), no obstante, las posibilidades que permite *Haskore*, han llevado a los autores del presente trabajo a plantear la utilización de *Haskore* como lenguaje intermedio. La intención es que ambos sistemas se complementen, mientras que *Haskore* permite definir un lenguaje musical, *Harmony* es un sistema completo de desarrollo de composiciones musicales. En [16], se define otro lenguaje puramente funcional basado en el cálculo lambda para la descripción de estructuras musicales. R. B. Dannenberg ha definido una serie de lenguajes basados en LISP [4], [5] y [6] para la descripción de sistemas de síntesis de sonidos. En estos trabajos, a pesar de que se utilizan sistemas basados en LISP, se subraya la necesidad de la evaluación perezosa.

Es conveniente mencionar la existencia de una serie de productos en el mercado [2][21][22] que facilitan la composición musical utilizando técnicas de programación convencionales. Estos sistemas se orientan principalmente a Jazz o formas similares que permiten una mayor libertad compositiva e interpretativa.

5 Conclusiones y Trabajos Futuros

Las ventajas de la utilización del lenguaje Haskell como lenguaje de desarrollo del núcleo del sistema se han puesto de relieve a la hora de modificar las reglas del sistema de armonización. La utilización de funciones de orden superior permite independizar el sistema de reglas de la función de ponderación particular. Asimismo, la evaluación perezosa permite definir composiciones conceptualmente *infinitas* con una mayor abstracción. Por otro lado, los combinadores de analizadores sintácticos han facilitado la definición del traductor entre la notación intermedia y el formato MIDI.

El sistema descrito se ha implementado en lenguaje Haskell 1.4 [19] utilizando la implementación HUGS 1.4 desarrollada en las universidades de Yale y Nottingham.

Actualmente, se está desarrollando el módulo dedicado a la generación de melodías mediante algoritmos genéticos y se están reescribiendo los algoritmos de generación mediante fractales (la versión anterior utilizaba el lenguaje C) en lenguaje *Haskell*. Asimismo, el núcleo del sistema se va a modificar para que admita como notación interna la notación descrita en [11].

Entre las futuras líneas de trabajo, caben destacar las siguientes metas a corto plazo. Por un lado, la integración de las distintas componentes del sistema en un entorno gráfico amigable es prioritaria. Se

espera que el usuario potencial del sistema no tenga altos conocimientos de informática y las tecnologías actuales en cuanto a desarrollo de interfaces gráficos con lenguajes puramente funcionales han avanzado considerablemente. Actualmente se está desarrollando una primera versión integradora de las componentes utilizando la librería gráfica de HUGS [20]. También se está investigando la posibilidad de utilizar herramientas de composición visual tanto para la edición musical como para la propia programación del sistema.

Respecto a la capacidad musical del sistema, surge la necesidad de investigar la incorporación de nuevas formas musicales y la influencia que pueden ejercer sobre los métodos de composición otros parámetros como el ritmo, el estilo o los instrumentos de ejecución.

Agradecimientos

La implementación de los diversos módulos se está llevando a cabo mediante los Proyectos Fin de Carrera[3] que han desarrollado o desarrollan Antonio Fernández Vidal [9], Mónica Freira Maira, Esther García Galán y Natividad Vilela Carral. Sin ellos, el presente sistema no habría sido desarrollado.

Referencias

[1] A. Alpern, Techniques for Algorithmic Composition of Music, Hampshire College Divisional Examination,Humanities and Arts.

[2] John A. Biles , GenJam: A Genetic Algorithm for Generating Jazz Solos, *International Computer Music Conference*, 1994

[3] John A. Biles, P. G. Anderson, L. W. Loggi, Neural Network Fitness Functions for a Musical IGA. *Rochestes Institute of Technology*, 1996

[4] R. B. Dannenberg, C. L. Fraley, P. Velikonja. A Functional Language for Sound Synthesis with behavioral abstraction and Lazy Evaluation. En *Computer Generated Music,* Denis Baggi (Editor). IEE Computer Society Press. 1992

[5] R. B. Dannenberg, The Implementation of Nyquist. A Sound Synthesis Language. En *Proceedings of the 1993 International Computer Music* Conference, International Computer Music Association, Sept. 1993, pp. 168-171

[6] R. B. Dannenberg, Expressing Temporal Behavior Declaratively. En *CMU Computer Science, A 25th Annyversary Commemorative.*R. F. Rashid (Editor), ACM Anthology Series (Chap. 3) pp. 47-68

[7] Dietter de la Motte. Armonía. *Ed. Labor S.A.* ISBN: 83-335-7859-6 1989.

[8] R. Franko Goldman, Harmony in Western Music, *W.W. Norton & Company Inc.* ISBN: 0 393 09746 3. 1965

[9] A. Fernández Vidal, J. E. Labra Gayo, Reconocedor Óptico de partituras. *Proyecto Fin de Carrera.* Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo, 1997

[10] R. Greenhouse. The Well-Tempered Fractal v3.0, A composers Tool for the Derivation of Musical Motifs, Phrases and Rhythms From the Beauty and Symmetry of Fractals, Chaotic Attractors and other Mathematical Functions.

[11] P. Hudak, T. Makucevich, S. Gadde, B. Whong. Haskore Music Notation – an Algebra of Music. *Journal of Functional Programming,* 6(3), Junio 1996.

[12] John Hugues. Why Functional Programming matters. *The Computer Journal*, 32(2), 98-107.

[3] Un Proyecto Fin de Carrera es un trabajo realizado por un alumno como requisito para obtener el titulo de Ingeniero en Informática

[13] G. Hutton, E. Meijer. Monadic Parser Combinators, 1996

[14] J. Jeuring, P. Janson. PolyP – a polytipic programming language extension. En POPL '97: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Laguages, pages 470-482. ACM Press, 1997-07-06

[15] G. Lee Nelson, "Sonomorphs: An Application of Genetic Algorithms to the Growth and Development of Musical Organisms". *Proceedings of the Fourth Biennial Art & Technology Symposium,* Connecticut College, 4-7 Marzo 1993, pp. 155-169.

[16] O. Orlarey, D. Fober, S. Letz, M. Bilton. Lambda calculus and music calculi. En *Proceedings of International Computer Music Conference.* International Computer Music Association, 1994

[17] L.A. Oliveira Rodríguez, J. E. Labra G. Sistema de Ayuda a la Composición Musical. Proyecto Fin de Carrera. Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo. 1996

[18] G. Pratt, The Dynamics of Harmony. Principles and Practice. Open University Press. ISBN: 0 335 10595 5, 1984

[19] Peterson, J. and Hammond, K. (editors). Report on the Programming Language Haskell 1.4, A Non-strict Purely Functional Language. Technical report. Yale University. Department of Computer Science. 1997 (April)

[20] A. Reid, The HUGS Grphics Library, Borrador incluido en la distibución del compilador HUGS 1.4

[21] The Symbolic Composer Language, Experimental Music Laboratory For The PowerMacintosh, http://www.xs4all.nl/~psto/index.html

[22] Nathan Tenny, Orfeo: Fractal Music, http://www.qualcomm.com/~ntenny/fmusic/

[23] Claus-Dietter Schulz, The Fractal Music Bibliography, Computer Center Univ. of Stuttgart (RUS) Dept. Communication Systems 70550 Stuttgart, Germany

[24] M.Vestin. Genetic Algorithms in Haskell with polytypic programming. Febrero de 1997. Master's Thesis. Göteborg University. 1997. Disponible por ftp en: http://www.cs.chalmers.se/~johanj/polytipism/genetic.ps

[25] Wentian Li, A Bibliography on 1/f Noise, *Rockefeller university*, http://linkage.rockefeller.edu/wli/1fnoise/

[26] J. Zamacois. Tratado de Armonía. Libros I, II y III. *Ed. Labor*, 1948