Lenguaje de programación para la composición automática de música

Programming language for automatic music composition

Erick G.G. de Paz[1,*], Perfecto Malaquías Quintero Flores[1], Xavier Quiñones Solís[2]

[1]Posgrado e Investigación, Tecnológico Nacional de México, Instituto Tecnológico de Apizaco
Av. Instituto Tecnológico S/N; 90300, Apizaco, Tlaxcala, MÉXICO
[2]Academia de materias teóricas, Escuela de Música del Estado de Tlaxcala
Fernando Solana N° 1, Col. San Luis Apizaquito, 90401, Apizaco, Tlaxcala, MÉXICO
[*]erickggdepaz@yandex.com

PALABRAS CLAVE:

Composición musical automática, Lenguajes de programación, Cadenas de Markov

RESUMEN

Este artículo presenta un lenguaje de programación que organiza fragmentos de un corpus musical en nuevas y coherentes composiciones de acuerdo con las instrucciones codificadas por el usuario. Para este fin, la sintaxis del lenguaje consiste en una colección de operadores de alto nivel para controlar sistemáticamente los aspectos armónicos y rítmicos de la música compuesta automáticamente. Los operadores principales están basados en un modelo de cadenas de Markov; lo que ocasiona que el proceso de composición sea resultado de varias decisiones aleatorias. Esto permite que para el mismo código, múltiples composiciones diferentes puedan ser producidas. El artículo da una perspectiva sobre los aspectos teóricos del lenguaje y sus sintaxis. Adicionalmente, algunos ejemplos de código y sus composiciones producidas son incluidos y analizados.

KEYWORDS:

Automatic music composition, Programming languages, Markov chain

ABSTRACT

This paper introduces a programming language which arranges fragments of a music corpus into new and coherent compositions according to the directives coded by the user. For this purpose, the syntax of the language consists in a collection of high-level operators for driving systematically the harmonic and rhythmic aspects of automatically composed music. The main operators are based on a Markov chains model; reason why the process of composition is result of several random decisions. This allows that for the same code, a lot of different compositions can be produced. The paper gives an overview on the theoretical aspects of language and its syntax. Additionally, some examples of code and their produced composition are included and analysed.

• **Recibido: 29 de mayo de 2024** • **Aceptado: 15 de octubre de 2024** • **Publicado en línea: 1 de febrero de 2025**

1. INTRODUCCIÓN

El presente artículo presenta un lenguaje de programación que permite restringir de manera esquemática las características rítmicas y armónicas de composiciones que se generan automáticamente. El objetivo es brindar un medio para que la composición automática satisfaga de mejor manera las expectativas del usuario. Adicionalmente, el lenguaje incluye ciertas facilidades para producir variaciones y extensiones a partir de un tema musical; esto permite que a diferencia de muchos otros enfoques de

DOI: 10.30973/progmat/2025.17.1/1
Copyright:© 2025 por los autores
Creative Commons Atribución 4.0 CC-BY

composición automática, las obras generadas resulten musicalmente coherentes.

El lenguaje es de tipo *imperativo* y está basado en operadores diseñados con un alto nivel de abstracción para controlar cómo se unen *fragmentos* extraídos de un *corpus*. La relación *corpus*-operadores se consigue a través de cadenas de *Markov*. El *corpus*, compuesto por obras de grandes compositores, es utilizado para producir un modelo matricial. Posteriormente, las matrices son empleadas para definir operaciones que primordialmente permiten obtener un conjunto de posibles opciones para extender un *fragmento* musical inicial.

El lenguaje se ha diseñado para componer música *polifónica tonal*, es decir, música que involucra la manipulación de al menos dos *voces* (instrumentos), los cuales presentan simultáneamente diferentes *melodías* regidas bajo las leyes de la *armonía* [1]. Como primera evaluación del enfoque propuesto, el presente trabajo se limita a la composición de música *polifónica* de dos *voces*. Sin embargo, de manera contraria a lo que se podría pensar, la *polifonía* con más de dos *voces* relaja progresivamente las restricciones armónicas [2].

Si bien existen lenguajes de programación destinados a la composición musical, los más populares *OpenMusic* e *Impromptu* están basados en el paradigma de programación funcional [3][4]. El uso de tales lenguajes exige un amplio dominio de la sintaxis de *Lisp*; hecho que dificulta su manipulación por parte de músicos. Por otro lado también se han desarrollado lenguajes imperativos, con sintaxis familiares para los usuarios de C/C++; ejemplos de esto son *Chuck* y *Csound*. Sin embargo el último, *Csound*, se centra en el control de la emisión de sonidos; en términos prácticos, resulta ser una partitura digital con múltiples opciones adicionales pero al igual que una partitura reserva para el usuario la tarea de componer [5]. Por su parte, los lenguajes *Chuck* e *Impromptu* están enfocados en el performance musical en vivo; por ello, su sintaxis ha sido centrada al esquema de programación *LiveCoding* [6]. De manera que, el enfoque propuesto en este artículo resulta novedoso al estado actual de la cuestión.

En la siguiente sección se realiza un breve bosquejo de los principios teóricos que rigen la composición. Posteriormente, en la tercera sección se describe el modelo numérico empleado; se define brevemente qué es un proceso de *Markov*, se expone cómo es empleado para producir un modelo matricial y qué operaciones se derivan de los modelos matriciales. La cuarta sección describe detalladamente cada aspecto de la sintaxis del lenguaje propuesto. La quinta sección incluye cuatro códigos de ejemplo y dos composiciones resultantes por cada código. Finalmente, la sexta sección contiene observaciones sobre los ejemplos de la quinta sección y un breve esbozo sobre las perspectivas del lenguaje.

2. FUNDAMENTOS DE LA COMPOSICIÓN TONAL

La música occidental dispone de 12 sonidos denominados *semitonos*; cada uno de ellos cuenta con versiones equivalentes más *agudas* y más *graves*. La disposición de dichos sonidos es fácil de visualizar a través de las teclas de un piano. En la Figura 1, se ilustra un piano etiquetado con los nombres de cada *semitono*. Obsérvese que, el nombre de *semitono* es periódico cada 12 teclas; y que, el subíndice incrementa después de 12 teclas. El nombre $\{\mathrm{do}, \mathrm{do\#}, \mathrm{re}, \ldots, \mathrm{la\#}, \mathrm{si}\}$ es útil para identificar la identidad del *semitono*; por otro lado el subíndice ayuda a identificar la *altura* (cualidad de ser más *agudo* o más *grave*). En lo sucesivo,

con el fin de facilitar la representación numérica de los *semitonos*, el conjunto discreto $\{do_3, do\#_3, \ldots, la\#_5, si_5\}$ será remplazado por el subconjunto de los números enteros $S=\{0,1,\ldots,34,35\}$; asumiendo las equivalencias $0=do_3, 1=do\#_3$, etc.

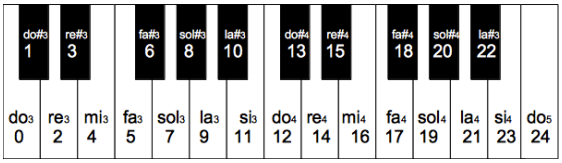


Figura 1. Teclas de un piano etiquetadas con el nombre de cada *semitono.*

Una pieza musical se desarrolla a través de la exposición de una serie de *semitonos* denominada *melodía*. Como restricción usual, se exige que los *semitonos* de la *melodía* estén relacionados *acústicamente*; propiedad que es facilitada por la agrupación de *semitonos* en *escalas diatónicas*. De acuerdo con Tchaikovsky [7], una *escala diatónica* es una jerarquía de sonidos centrada en un *semitono* denominado *tónica*; tal jerarquía es fácil de establecer por medio de ciertos *intervalos* (distancias en *semitonos*). Por ejemplo, la disposición de las teclas blancas/negras de un piano atiende a los *intervalos* propios de la *escala diatónica* desarrollada a partir de la *tónica* *Do* (o simplemente *escala mayor de* *Do*). Recurriendo a la Figura 1, la numeración de las teclas blancas del piano permite identificar los *intervalos* que conforman la *escala mayor* de *Do*. Conservando los mismos *intervalos* a partir de una *tónica* distinta, se pueden obtener *escalas diatónicas* para otros *semitonos*. Así, a partir del conjunto S, es posible derivar subconjuntos correspondientes a las *escalas diatónicas* de cada *semitono*, ver Tabla 1.

Tabla 1. *Escalas diatónicas* de do, do#, **...,** la# **y** *si*

Subconjuntos diatónicos de S	**Modo mayor de**	**Modo menor de**
$S_0=\{0,2,4,5,7,9,11,12,14,\ldots,31,33,35\}$	do	la
$S_1=\{1,3,5,6,8,10,12,13,15,\ldots,32,34\}$	do#	la#
$\ddots$		
$S_{10}=\{10,12,14,15,17,19,21,22,24,..,34\}$	la#	sol
$S_{11}=\{11,13,15,16,18,20,22,23,25,\ldots,35\}$	si	sol#

Considere además que los mismos *semitonos* que conforman la *escala mayor* de *Do* pertenecen alternativamente a la *escala menor* de *La*; de manera que, cada *escala diatónica* da origen a dos interpretaciones, denominadas *modos*, clasificadas como *mayor* o *menor*.

Los conceptos de *escala diatónica* y *modo* son los principios *fundamentales* que restringen la composición *melódica*. Además, con el fin de producir *polifonía*, es necesario introducir conceptos relacionados a la *armonía*, técnica musical que estudia las combinaciones de *semitonos* simultáneos que resultan gratos al oído. El presente trabajo se limita a la *polifonía de dos voces*, estilo musical que implica la ejecución de dos *melodías* simultáneas. Empleando pares ordenados del conjunto $S^2=S\times S$ para referirse a pares de *semitonos* simultáneos, es posible representar un *fragmento* de dos *voces* como una sucesión de pares *ordenados* [8]. De manera análoga, se puede obtener un producto cartesiano que sólo involucre los *semitonos* de una *escala* determinada, por ejemplo $S_0^2=S_0\times S_0$.

La *armonía* centra su estudio en la formación y manipulación de *acordes*, estructuras de 2 o más *semitonos* que resultan agradables al ser ejecutados simultáneamente [9]. Invariablemente todo *acorde* posee un *semitono fundamental*, con el cual el resto de los *semitonos* parece fundirse. Esta percepción de unión es ocasionada por las propiedades acústicas que presentan ciertos pares de *semitonos*, denominados pares *consonantes*.

A continuación, se define formalmente el concepto de *semitonos consonantes*: Sea $q=[s_x,s_y]$ un par ordenado del producto cartesiano S^2, adicionalmente de acuerdo con los valores de q, sean $s_G=min\,(s_x,s_y)$ y $s_A=max\,(s_x,s_y)$. Si la distancia en *semitonos* $d=[(s_A-s_G)\,mod\,12]$ es un valor contenido en:

- el conjunto $\{0,3,4,7\}$, se dice que el par de *semitonos* es *consonante* y que su *fundamental* es s_G;

- el conjunto $\{5,8,9\}$, se dice que el par de *semitonos* es *consonante* y que su *fundamental* es s_A;
- el conjunto $\{1,2,6,10,11\}$, se dice que el par es *disonante* y por tanto no posee *fundamental*.

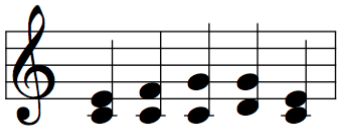
La definición anterior es la misma para cualquier subconjunto de S^2, por ejemplo S_0^2. En las subsecuentes secciones, se emplea la función $f(q)$ para referirse a la *fundamental* del par q; por ejemplo $f([0,7])$ equivale a 0, $f([0,4])$ a 4, etc.

Establecidos los principios más elementales relativos a la música, en la siguiente sección se expone la técnica matemática empleada para modelarlos.

3. MODELO NUMÉRICO

La manera más intuitiva de modelar la música es a través de un *proceso estocástico*; considerando a la sucesión de *semitonos* que conforman una *melodía* como los valores que toma una *variable estocástica* a lo largo del tiempo [10]. Para nuestro caso de estudio, la *polifonía de dos voces*, la *variable estocástica* asume valores del conjunto S^2 y el tiempo es representado por el conjunto discreto $t=1,2,\ldots,T$. Con fines ilustrativos, la Tabla 2 ejemplifica la representación de un *fragmento* musical de dos *voces* como un *proceso estocástico*.

Tabla 2. Ejemplo de un *fragmento* musical de dos *voces* y su representación como *proceso estocástico*

Partitura de una fragmento musical a dos voces	**Variable estocástica**
	$V_1=[12,16]$ $V_2=[12,17]$ $V_3=[12,19]$ $V_4=[14,19]$ $V_5=[12,16]$

Los *procesos aleatorios*, o *cadenas*, *de Markov* son *modelos estocásticos* ampliamente utilizados para tratar con composiciones musicales [11]. La conjetura distintiva de tales modelos es que el valor actual de la variable *estocástica* V_t es suficiente para estimar cual será el valor próximo V_{t+1}. Los procesos de *Markov*, en los que la variable sólo asume valores propios de un conjunto finito, pueden ser consensados en una *matriz de transición* [12]. Por definición, la *matriz de transición* M de una *variable estocástica* es una matriz que posee una fila por cada valor posible de la variable V_t y una columna por cada valor posible en la siguiente unidad de tiempo V_{t+1}. Por tanto, una *matriz de transición* siempre es cuadrada $(n\times n)$; para el caso particular de las composiciones de dos *voces*, se tiene que $n=|S^2|$, o bien, n es igual al número de pares ordenados contenidos en el conjunto finito S^2. La *matriz de transición* en la celda $M_{i,j}$ contiene la probabilidad de que en un *fragmento* se ejecute el i-ésimo par ordenado del conjunto S^2, seguido por el j-ésimo par; formalmente:

$$M=\begin{pmatrix} p\left(V_{t+1}=S^2_1 \mid V_t=S^2_1\right) & \ldots & p\left(V_{t+1}=S^2_n \mid V_t=S^2_1\right) \\ \vdots & \ddots & \vdots \\ p\left(V_{t+1}=S^2_1 \mid V_t=S^2_n\right) & \ldots & p\left(V_{t+1}=S^2_n \mid V_t=S^2_n\right) \end{pmatrix},$$

$$\text{donde } S^2=\begin{pmatrix} S^2_1 \\ \vdots \\ S^2_n \end{pmatrix}=\begin{pmatrix} [0,0] \\ \vdots \\ [35,35] \end{pmatrix}$$

La matriz anterior se puede estimar aislando *fragmentos* musicales de dos *voces* provenientes de un *corpus* de composiciones; tales *fragmentos* son transformados a sucesiones de pares ordenados, tal como ocurre en la Tabla 2. Así, cada probabilidad $M_{i,j}$ se puede estimar dividiendo $|i,j|$ (el número de veces que la transición del par i-ésimo al j-ésimo ocurre) entre $|i|$ (el número de veces que el par i-ésimo ocurre).

Adicionalmente, se puede estimar una matriz T que modele la versión *retrógrada* (en sentido temporal inverso) de los *fragmentos*, tal que la celda $T_{i,j}$ contenga $|j,i|/|j|$, formalmente:

$$T=\begin{pmatrix} p\left(V_{t-1}=S^2_1 \mid V_t=S^2_1\right) & \ldots & p\left(V_{t-1}=S^2_n \mid V_t=S^2_1\right) \\ \vdots & \ddots & \vdots \\ p\left(V_{t-1}=S^2_1 \mid V_t=S^2_n\right) & \ldots & p\left(V_{t-1}=S^2_n \mid V_t=S^2_n\right) \end{pmatrix},$$

$$\text{donde } S^2=\begin{pmatrix} S^2_1 \\ \vdots \\ S^2_n \end{pmatrix}=\begin{pmatrix} [0,0] \\ \vdots \\ [35,35] \end{pmatrix}$$

Analizando detalladamente las filas de las matrices M y T:

1) M_i contiene en su j-ésima entrada la probabilidad de que el par ordenado siguiente sea el j-ésimo, asumiendo que el par ordenado actual es el i-ésimo.
2) T_i contiene en su j-ésima entrada la probabilidad de que el par ordenado anterior sea el j-ésimo, asumiendo que el par ordenado actual es el i-ésimo.

Ambas matrices son útiles para realizar las siguientes operaciones:

1) Obtener el conjunto $A \subseteq S^2$ de pares ordenados que podrían anteceder a algún par ordenado perteneciente al conjunto dado $B \subseteq S^2$. Por ejemplo, suponga que se desea encontrar el conjunto A de pares ordenados que pueden anteceder a algún par de *semitonos* con *fundamental* 0, es decir, un conjunto $B \subseteq S^2 \mid f(b)=0 \forall b \in B$. Para ello:
   1.a Se calcula el número $|B|$ (de elementos en B) y se establece $p_B=1/|B|$.
   1.b Se crea un vector auxiliar w tal que si el i-ésimo par ordenado de S^2 está contenido en B, $w_i=p_B$; en caso contrario $w_i=0$. Como resultado, w es un vector de dimensión $1 \times |S^2|$.
   1.c Se calcula el vector $r=wT$ que es considerado para obtener el conjunto A; si y sólo si $r_i \neq 0$ el i-ésimo par ordenado de S^2 pertenece al subconjunto A. En lo sucesivo, se usa la notación $antecesor(B)$ para referirse al vector r, estimado a partir del conjunto B.
2) Obtener el conjunto de pares ordenados B que pueden suceder después de algún par ordenado del conjunto dado A. Para ello, de manera análoga al proceso anterior, se estima el vector w a partir del conjunto A y se multiplica con la matriz M. En lo sucesivo, se usa la notación $sucesor(B)$ para referirnos a $r=wM$.
3) Obtener el subconjunto $B \subseteq S^2$ de pares ordenados que pueden ir entre algún par ordenado del subconjunto $A \subseteq S^2$ y algún par ordenado del subconjunto $C \subseteq S^2$. Para ello:
   3.a Se obtienen los vectores $r=sucesor(A)$ y $t=antecesor(C)$.
   3.b Se calcula el vector $m=(r_1 t_1 \ \cdots \ r_n t_n)$; en lo sucesivo para referirnos a él, se empleará $medio(A,C)$.
   3.c El conjunto B se obtiene a través del vector m: si y sólo si $m_i \neq 0$ el i-ésimo par ordenado de S^2 pertenece al subconjunto B.

Considérese que los vectores obtenidos a través de las funciones $sucesor$, $antecesor$ y $medio$ no sólo indican cuales pares ordenados de S^2 pertenecen al correspondiente subconjunto; sino además señalan las probabilidades de encontrar cada secuencia de pares en los *fragmentos* usados para estimar las matrices M y T. En la siguiente sección dichas funciones son empleadas (a manera de operadores) para construir *fragmentos* musicales.

4. EL LENGUAJE

A continuación se describen los elementos de un lenguaje de programación desarrollado a partir de las funciones $sucesor$, $antecesor$ y $medio$ que las matrices M y T permiten.

1 ***Identificadores***: Los *identificadores* son conjuntos de letras $\{A,a,B,b,\ldots,Z,z\}$ y dígitos $\{0,1,\ldots,9\}$ empleados para hacer referencia a un *fragmento*. Como únicas restricciones, un *identificador* siempre empieza con una letra (no con un número) y no puede contener espacios, tabulaciones o caracteres especiales.

2 ***Fragmentos***: Dentro del lenguaje, un *fragmento* es una sucesión de pares ordenados (par de *semitonos*) estrictamente pertenecientes al subconjunto tonal S_0^2. Como extensión mínima todo *fragmento* posee al menos un par ordenado. Por ejemplo, el *fragmento* correspondiente a la partitura de la Tabla 2

puede ser formalmente representado como la siguiente tupla:

$$fragmentoTabla2=\{[12,16],[12,17],[12,19],[14,19],[12,16]\}$$

Para hacer referencia al primer par ordenado ($[12,16]$) se ocupará la notación simplificada $fragmentoTabla2_P$; análogamente para referirse al último ($[12,16]$), se usará $fragmentoTabla2_U$.

3 ***Referencias tonales***: Una *referencia tonal* sirve para crear un *fragmento* que contiene sólo un *par de semitonos consonantes* con una *fundamental* específica. La tabla 3 muestra las tres referencias tonales que existen en el lenguaje.

Tabla 3. Lista de referencias tonales y su interpretación

Referencia tonal	Un fragmento con un par ordenado seleccionado aleatoriamente del conjunto R
@	$R \subset S_0^2 \vee [f(q) \text{ mód } 12]=0 \forall q \in R$, *fundamental* 0.
!!	$R \subset S_0^2 \vee [f(q) \text{ mód } 12]=5 \forall q \in R$, *fundamental* 5.
!	$R \subset S_0^2 \vee [f(q) \text{ mód } 12]=7 \forall q \in R$, *fundamental* 7.

4 ***Referencias numéricas***: Cualquier número natural x dentro del lenguaje es considerado una referencia numérica. Para fines prácticos, cualquier número x equivale a una referencia tonal basada en el conjunto:

$$R \subset S_0^2 \mid [f(q) \, mod \, 12]=[x \, mod \, 12] \forall q \in R.$$

De manera que, dentro del lenguaje, $frag=!!$ es equivalente a $frag=5$.

5 ***Operadores musicales***: Son la familia de operadores más importante, pues están directamente basados en las funciones $sucesor$, $antecesor$ y $medio$. En la Tabla 4 se listan tales operadores.

6 ***Operadores de extensión***: Los *operadores de extensión* sirven para crear nuevos *fragmentos* basados en los ya existentes. La Tabla 5 lista las maneras apropiadas de emplearlos.

7 ***Operadores genéricos***: Estos operadores reciben el nombre de genéricos pues se trata de operadores presentes en la mayoría de lenguajes: asignación e impresión. La Tabla 6 muestra su empleo.

Tabla 4. Lista del uso de los operadores musicales.

Sintaxis	Interpretación
$fragmento'x$	Obtener un par de *semitonos consonante* con *fundamental* $[x \text{ mód } 12]$ que extienda de manera apropiada a *fragmento*. Para ello, se estima el vector $t=sucesor(\{fragmento_U\})$; después se modifica $t_i=0$, si el i-ésimo par del conjunto S^2 no tiene *fundamental* $[x \text{ mód } 12]$. A partir del t filtrado, se selecciona aleatoriamente un par ordenado; asignando la probabilidad de selección para cada i-ésimo par en función de su correspondiente valor t_i. En el ámbito de los *algoritmos evolutivos* el método de selección anterior es denominado *ruleta* [13]. Por tanto, en lo sucesivo, la notación $ruleta(t)$ hará referencia al par seleccionado con el método de *ruleta*.
$x'fragmento$	Obtener un par de *semitonos consonantes* con *fundamental* x que sea un predecesor apropiado para el *fragmento*. De manera análoga a la operación previa, se estima el vector $t=antecesor(\{fragmento_P\})$, luego modifica $t_i=0$, si el i-ésimo par del conjunto S^2 no tiene *fundamental* $[x \text{ mód } 12]$. Finalmente, a partir del t filtrado, se selecciona un par ordenado $ruleta(t)$.
$frag1?frag2$	Obtener un par de *semitonos* que pueda ejecutarse en medio de los *fragmentos* $frag1$ y $frag2$. Con la notación previamente definida, es fácil establecer que la operación $frag1?frag2$ devolverá un *fragmento* que contiene el par ordenado de *semitonos* $ruleta(medio(frag_U, frag_P))$.
$fragmento''x$	Obtener un *fragmento* $res=fragmeto''x$, tal que entre *fragmento* y *res* pueda añadirse un par de *semitonos* adicional; generalmente obtenido a través de $fragmento?res$. Para ello, el operador ('') realiza las misma acciones internas que el operador (′); sin embargo, en lugar de iniciar sus estimaciones a partir del vector $t=sucesor(\{fragmento_U\})$, las inicia a partir del vector $t=sucesor(B)$. El conjunto B está conformado por cada i-ésimo par ordenado de S^2 tal que $b_i \neq 0$, para $b=sucesor(\{fragmento_U\})$.
$x''fragmento$	Obtener un *fragmento* $res=x''fragmeto$, tal que entre *res* y *fragmento* pueda añadirse un par de *semitonos* adicional.

Tabla 5. Lista del uso de los operadores de extensión.

Sintaxis	Interpretación
$frag1 \wedge frag2$	La operación da como resultado un nuevo *fragmento* que contiene los pares ordenados del *fragmento* $frag1$ y posteriormente los del *fragmento* $frag2$.
~*fragmento*	La operación produce un nuevo *fragmento* que contiene los mismo pares ordenados que *fragmento*; pero en sentido inverso. De manera que, el primer par de *semitonos* en ~*fragmento* es el último par en *fragmento*, el segundo par de *semitonos* en ~*fragmento* es el penúltimo en *fragmento* y así sucesivamente. Musicalmente, se dice que ~*fragmento* es la versión *retrógrada* de *fragmento*.
$n:fragmento$	El operador (:) produce un *fragmento* que contiene n veces la sucesión de pares contenida en *fragmento*, es decir, copia n veces *fragmento*. La operación siempre tiene como primer operando un valor numérico n, el cual de manera excepcional no se interpreta como *referencia numérica*.

Tabla 6. Lista del uso de los operadores genéricos.

Sintaxis	Interpretación
$identificador = fragmento$	El operador (=) asocia el primer operando, que rigurosamente es un *identificador*, con un *fragmento*.
$n \mid fragmento$	Basado en *fragmento*, el operador (|) produce una salida en formato MIDI (extensión estándar para almacenar secuencias musicales). La operación siempre tiene como primer operando un valor numérico n, el cual de manera excepcional no se interpreta como *referencia numérica*. El valor numérico n indica cuantos *semitonos* añadir a todos los pares ordenados de *fragmento* para cambiar su *escala* original S_0 a una nueva S_n.
$expresión_1, \cdots, expresión_n$	El operador (,) establece el orden secuencial en el que múltiples expresiones han de ser evaluadas. Por ejemplo, con la secuencia: $a=@, b=a\,4, ab=a \wedge b,$ primero se define a, luego b y finalmente ab. El resultado de la secuencia $expresión_1, \cdots, expresión_n$ siempre es el fragmento resultante de la última expresión, es decir, $expresión_n$. El operador (,) es comparable al delimitador (;) de los lenguajes C/C++/Java [14]. Sin embargo, su definición como operador implica un sutil cambio en la sintaxis. El operador se usa entre las expresiones $expresión_1, expresión_2, expresión_3$ y no al final de cada expresión $expresión_1; expresión_2; expresión_3;$.

La gramática libre de contexto para el lenguaje descrito es expresada a través de la notación de Backus-Naur [15] en la Tabla 7.

Tabla 7. Notación de Backus -Naur del lenguaje.

No terminal	Producción
<Dígito>	::= ``0´´|``1´´ |``2´´|...|``8´´|``9´´
<Natural>	::= <Dígito><Natural> | <Dígito>
<ReferenciaN>	::= <Natural>
<Letra>	::= ``A´´|``a´´|``B´´|...|``Z´´|``z´´|
<Identificador>	::= <Letra>
	|<Identificador><Natural>
	|<Identificador><Letra>
<Expresión>	::= <Identificador>|``@´´|``!´´|``!!´´|``~´´
	|<Identificador>``=´´<Expresión>
	|<Identificador>``=´´<ReferenciaN>
	|<Expresión>``,´´<Expresión>
	|<Expresión>``&´´<Expresión>
	|<Expresión>``?´´<Expresión>
	|<ReferenciaN>``:´´<Expresión>
	|<ReferenciaN>``|´´<Expresión>
	|<Expresión>``´´´´<ReferenciaN>
	|<Expresión>``´´´´´´<ReferenciaN>
	|<ReferenciaN>``´´´´<Expresión>
	|<ReferenciaN>``´´´´´´<Expresión>
	|``{´´<Expresión>``}´´
	|``(´´<Expresión>``)´´
	|``[´´<Expresión>``]´´

La precedencia de operadores, que se establece la Tabla 8, puede ser alterada empleando los pares (),{},[] para encerrar cualquier operación; los tres pares pueden usarse indistintamente, siempre y cuando se utilicen los símbolos mutuamente correspondientes para abrir y cerrar.

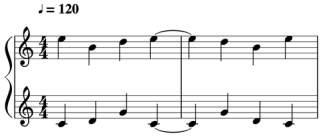
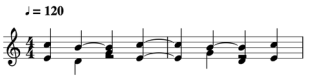
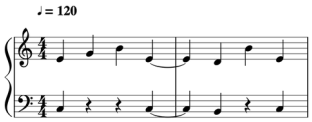
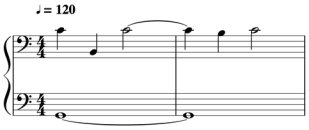
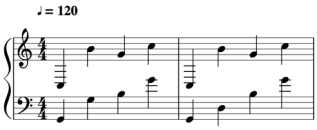
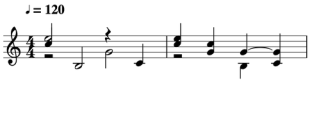
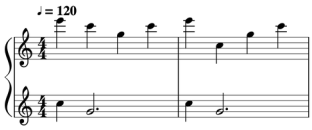
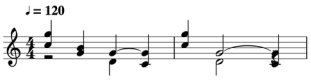
Tabla 8. Precedencia de operadores, los operadores tabulados se resuelven inicialmente de izquierda a derecha.

Mayor precedencia					Menor precedencia
”,’	?,??	~	&	:	=,| ,

5. RESULTADOS

La Tabla 9 presenta resultados obtenidos con la implementación del lenguaje de programación anteriormente descrito. Considérese que, debido a las decisiones de naturaleza aleatoria realizadas por la función *ruleta*, es altamente probable que cada compilación de un mismo código origine una propuesta musical distinta.

Tabla 9. Ejemplos de códigos y sus correspondientes fragmentos musicales resultantes.

Ejemplo	Código	Compilación 1	Compilación 2
1	[1] primerPar=@, [2] segundoPar=primerPar'7, [3] tercerPar=segundoPar'11, [4] [5] fragmento = primerPar & segundoPar & [6] tercerPar & primerPar, [7] [8] 0| ~fragmento & fragmento		
2	[1] primerPar=@, [2] segundoPar=primerPar'7, [3] tercerPar=segundoPar'11, [4] [5] fragmento = primerPar & segundoPar & [6] tercerPar & primerPar, [7] [8] segundoPar = primerPar ? tercerPar, [9] [10] fragmento2 = primerPar & segundoPar & [11] tercerPar & primerPar, [12] [13] 0|fragmento & fragmento2		
3	[1] primerPar=@, [2] tercerPar=primerPar"7, [3] cuartoPar=tercerPar'0, [4] [5] segundoPar = primerPar ? tercerPar, [6] fragmento1 =primerPar & segundoPar & [7] tercerPar & cuartoPar, [8] [9] segundoPar = primerPar ? tercerPar, [10] fragmento2 =primerPar & segundoPar & [11] tercerPar & cuartoPar, [12] [13] fragmentos = fragmento1 & fragmento2, [14] [15] 0 | fragmentos		
4	[1] primerPar=@, [2] tercerPar=primerPar"7, [3] cuartoPar=tercerPar'0, [4] [5] fragmentos = 2: [6] { primerPar & [7] [primerPar ? tercerPar] & [8] tercerPar & [9] cuartoPar [10] }, [11] [12] 0 | fragmentos		

Descripción del Ejemplo 1: La primera línea crea el *fragmento* designado primerPar, el cual incluye un par de *semitonos consonantes* con *fundamental* 0. En la segunda línea, se crea un *fragmento* designado segundoPar, el cual incluye un par de *semitonos consonantes* con *fundamental* 7 que además es apropiado para extender al *fragmento* primerPar. De manera semejante, tercerPar designa a un *fragmento* que es apropiado para extender segundoPar y que contiene un par de *semitonos* con *fundamental* 11. En las líneas [5] y [6], se unen los segmentos primerPar, segundoPar, tercerPar, y de nuevo, primerPar en un solo *fragmento* designado fragmento. Finalmente en la línea [8], se genera una salida en formato MIDI conformada por las versiones *retrógrada* y original de fragmento.

Descripción del Ejemplo 2: Hasta la línea [7], el código es idéntico al usado en el Ejemplo 1. En la línea [8], el *fragmento* segundoPar, es remplazado por algún *fragmento* que contenga par de *semitonos* que, sin importar su *fundamental*, sea apropiado para colocarse entre los *fragmentos* primerPar y tercerPar. La nueva versión de segundoPar es empleada en las líneas [10] y [11] para crear fragmento2, que puede ser considerado como una

variación de fragmento. Finalmente en la línea [13], se genera un MIDI basado en fragmento y fragmento2.

Descripción del Ejemplo 3: En la primera línea se crea el *fragmento* primerPar que contiene un par *consonante* con *fundamental*0; en la segunda línea se crea el *fragmento* tercerPar que contiene un par *consonante* con *fundamental*7. En la tercera línea, se crea el fragmento cuartoPar que contiene un par de *semitonos* con *fundamental*0 y que es apropiado para extender el tercerPar. Nótese que en la línea [2], el operador (") asegura que siempre existe un *fragmento* apropiado para colocarse entre primerPar y tercerPar; tal *fragmento*, segundoPar, es encontrado a través del operador (?) en las líneas [5] y [9]. Por cada ejecución del operador (?) se obtienen distintas versiones de segundoPar que son usadas respectivamente en fragmento1 y fragmento2. La línea [13] crea fragmentos uniendo fragmento1 y fragmento2. En la línea [15] fragmentos, es usado para genera la salida MIDI.

Descripción del Ejemplo 4: Este código realiza exactamente las misma funciones que el código del Ejemplo 3. Considérese que, en el Ejemplo 3, las líneas [5][6][7] y [9][10][11] son idénticas. Para evitar esta repetición y facilitar la lectura del código, se emplea la instrucción (:) que obtiene la unión de los *fragmentos* resultantes de la ejecución iterada de las operaciones contenidas entre ({...}). Adicionalmente, se ha remplazado segundoPar por la operación que lo define.

Respecto al *corpus* empleado para producir las matrices M y T, se consideraron 62 obras. Los compositores pertenecientes al *corpus* son: Bach, Beethoven, Bizet, Borodin, Brahms, Chopin, Clementi, Field, Grieg, Handel, Liadov, MacDowell, Massenet, Mendelssohn, Mozart, Offenbach, Pachelbel, Prokofiev, Puccini, Rebikov, Saint-Saens, Satie, Schubert, Schumann, Strauss, Tchaikovsky y Verdi.

6. CONCLUSIONES

A partir de los ejemplos contenidos en la Tabla 9 de la sección anterior, se pueden realizar las siguientes observaciones:

- El lenguaje consigue crear composiciones basadas en los *fragmentos* del *corpus* que respeten las directivas del usuario. Sin embargo, existen algunos errores técnicos que los compositores humanos evitarían con facilidad; el más evidente de ellos es la inclusión de *intervalos* excesivamente largos.
- Es común que se produzcan obras que, a pesar de seguir las normas armónicas, no resulten atractivas. Para tales situaciones, resulta una gran ventaja que cada compilación produzca propuestas musicales distintas; además, esta característica permite explorar alternativas musicales de manera más concisa.
- El uso lenguaje de programación requiere cierto dominio, pero sólo sobre aspectos elementales de la música. De manera que, puede ser empleado sin problemas por principiantes.

En general, se concluye que el lenguaje propuesto genera resultados aceptables y que es un medio que permite la creación espontánea de múltiples obras alternativas.

RECONOCIMIENTOS

Los autores quisieran agradecer a la Mtra. Evelyn Groesch, directora de la Escuela de Música del Estado de Tlaxcala, por todas las facilidades brindadas.

REFERENCIAS

[1] Rameau JP. *Treatise on Harmony*. New York: Dover Publications; 2012.
[2] García Gago J. *Tratado de contrapunto: tonal y atonal*. Barcelona: Publicacions Clivis; 1977.
[3] Agon A. OpenMusic: *Un langage visuel pour la composition musicale assistée par ordinateur* [tesis doctoral]. Paris: Université de Paris 6; 1998.

[4] Sorensen A. Impromptu: an interactive programming environment for composition and performance. *Proc. of the Australasian Computer Music Conference*. Brisbane. Australia. 2009;2(4):1-6.

[5] Boulanger R. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. Cambridge (MA): MIT Press; 2000.

[6] Blackwell A, Collins N. The programming language as a musical instrument. *Proc. of the Psychology of Programming Interest Group*. Brighton, UK. 2005;5(3):284-289.

[7] Tchaikovsky P. *Guide to the Practical Study of Harmony*. New York: Dover Publications; 2005.

[8] Tymoczko D. *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. New York: Oxford University Press; 2011.

[9] Rimsky-Korsakov N, Achron J, Hopkins N. *Practical Manual of Harmony*. New York: C. Fischer; 2005. doi: 10.1177/002743213101800145.

[10] Van Der Merwe A, Schulze W. Music generation with Markov models. *IEEE MultiMedia*. 2011;18(3):78-85. doi: 10.1109/MMUL.2010.44.

[11] Edwards M. Algorithmic composition: computational thinking in music. *Communications of the ACM*. 2011;54(7):58-67. doi: 10.1145/1965724.1965742.

[12] Bolch G, Greiner S, de Meer H, Trivedi K. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. 2nd ed. Hoboken (NJ): John Wiley & Sons; 2006.

[13] Lipowski A, Lipowska D. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*. 2012;391(6):2193-2196. doi: 10.1016/j.physa.2011.12.004.

[14] Deitel HM, Deitel PJ. *Cómo programar en C/C++ y Java*. 5ª ed. México: Pearson Educación; 2004.

[15] Cooper K, Torczon L. *Engineering a Compiler*. 2nd ed. Amsterdam: Elsevier Science; 2011.

ACERCA DE LOS AUTORES

Erick G.G. de Paz obtuvo el grado de ingeniero en computación por la Universidad Autónoma de Tlaxcala en 2016. Es miembro de la MACVNR; y entre sus intereses de investigación se incluyen la composición algorítmica, la minería de patrones y las heurísticas de optimización. Actualmente, estudia en un programa de posgrado en el Tecnológico Nacional de México.

Perfecto Malaquías Quintero Flores obtuvo el grado de Doctor en 2013 por la Universidad de Montpellier, Francia. Sus áreas de investigación son los sistemas difusos, la minería de datos y la computación de alto desempeño. Actualmente es miembro del núcleo académico de posgrado del Tecnológico de Apizaco perteneciente al Tecnológico Nacional de México.

Xavier Quiñones Ladrón de Guevara y Solís, graduado como concertista clarinetista a nivel técnico por la Escuela de Música de la UNAM. Realizó estudios de Armonía, Contrapunto y Composición con el maestro Humberto Hernández Medrano en el “Taller Polifónico de México” y estudios de piano profesional bajo la dirección del maestro Leopoldo González. Ha publicado obras y arreglos originales para los conjuntos de cámara del Clarinet Institute (Los Angeles). Desde 1999, imparte cursos de armonía, análisis y contrapunto en la Escuela de Música del Estado de Tlaxcala. Durante el período 2009-2010, fue miembro del comité de planeación del fondo Estatal para la cultura y las artes de Tlaxcala. Desde 2010, es coordinador académico de la EMET.