AUTÓMATAS BASADOS EN GRAMÁTICAS PARA LA CREACIÓN DE GENERADORES MUSICALES Y SINTETIZADORES DE SONIDO

Proyecto de Tesis para obtener el grado de
Maestro en Música – Tecnología Musical
Presentado por: Mijael Gutiérrez López
Director de Tesis: Dr. Felipe Orduña Bustamante

Escuela Nacional de Música
Universidad Nacional Autónoma de México

Indice

Capítulo 1 Introducción

Las nuevas tecnologías abarcan, hoy en día, casi todos los aspectos de nuestras vidas. Cotidianamente las encontramos interviniendo en nuestras actividades, desde las más básicas a las más especializadas. Incluso, muchas personas lo toman como algo tan natural que no se percatan de que los están utilizando, y muchos otros no concebirían poder vivir su vida sin estas herramientas tecnológicas.

Por supuesto, las artes en general y la música en particular no se han librado de su influencia y apenas estamos tratando de vislumbrar los efectos de esto. Lo que es innegable es que comienzan ya a afectar la forma en que vemos las artes en muchos aspectos.

El uso de la inteligencia artificial y autómatas se ha extendido tanto dentro del ámbito de la tecnología de la información y telecomunicaciones que sería casi imposible el funcionamiento de muchos de los dispositivos que hoy día están presentes en la vida cotidiana.

No debería de extrañarnos entonces que la utilización de estas *máquinas* virtuales llegue también al campo de la música.

Desde hace varios años se han realizado diversos experimentos en los que se hace uso de la inteligencia artificial para construir autómatas que generen material musical o bien, que interactúen junto con los intérpretes en la ejecución de obras musicales. De la misma manera, pero en mucho menor grado, para el control en el procesamiento de señales digitales de audio aplicados a la música.

Tenemos así que han surgido algunos experimentos más o menos célebres como el EMI de David COPE, el BP2 de Bel y Kippen, el ACCSM de Michael Chan y algunos otros aún menos conocidos.

Sin embargo y, curiosamente, a pesar de que el desarrollo de estos dispositivos ha tenido un crecimiento vertiginoso, su aplicación en el área de la música ha sido modesta en comparación con lo que se ha hecho en las telecomunicaciones, informática, entretenimiento y desarrollo de tecnología de uso doméstico.

Hoy día el desarrollo de autómatas aplicados a la música se hace, por lo general, dentro de instituciones educativas, y sólo recientemente unas cuantas empresas comienzan a voltear sus ojos hacia las mismas. Tenemos entonces que, a pesar de los avances, existen muy pocos sistemas de éste tipo que se hayan concebido como herramientas prácticas y accesibles para toda la gente que desee utilizarlas como ayuda para su trabajo musical; no importando su formación musical, ni género y mucho alguna "posición estética". Puntos en los que no entraremos a discusión, siendo éstos no fundamentales para el desarrollo de nuestro estudio.

1.1 El Proceso creativo.

Hablar de composición musical y el proceso creativo involucrado nos lleva por un terreno bastante delicado. Resulta sumamente difícil definir lo que conocemos como *creatividad* y tratar de hacerlo nos llevaría por campos que no son propiamente los que nos ocupan en éste trabajo.

Por otro lado, tratar de definir la *musicalidad* de una obra o fragmento musical, o incluso si algo es musical o no, resulta aún más controversial y escabroso. Sin embargo, dado que la parte medular de ésta investigación no se fundamenta en estos aspectos, es posible continuar en nuestro estudio sin necesidad de detenernos en éstas discusiones.

1.2 - Presentación del tema de tesis.

El tema de ésta tesis es el uso de autómatas para la creación de aplicaciones de computo dirigidas a la música y su implementación práctica.

En ella haremos un breve repaso de diferentes sistemas de inteligencia artificial que se han utilizado para la creación de aplicaciones musicales y nos centraremos particularmente en el uso de autómatas

1.3 Objetivos y resultados generales

Para poder acotar nuestra área de investigación se plantean los siguientes objetivos:

Objetivo General:

Conocer el funcionamiento de autómatas como sistemas de inteligencia artificial y su implementación en la elaboración de aplicaciones de cómputo musical

Los objetivos específicos que buscamos son:

- Análisis de un sistema basado en autómatas y su implementación para la creación de una aplicación que genere pequeñas piezas musicales.
  Las metas no abarcan el crear todo un compositor automático de obras musicales, sino más bien en el análisis de cómo podemos implementar un autómata para la generación de piezas que además pueda extenderse para su uso práctico por músicos de cualquier nivel y género.

- Implementación de un autómata para controlar parámetros de algunos procesadores de audio. Específicamente: Sintetizador FM y Sampler.

1.4 - Descripción del contenido de la tesis. (Organización)

Este trabajo está organizado de la siguiente manera:

En el capítulo 2 se hace una revisión de antecedentes teóricos para un mejor entendimiento del desarrollo del proyecto, se encuentra dividido en dos partes:

1. Un repaso histórico de sistemas de IA empleados para la creación musical así como una revisión general de los sistemas más populares hasta el momento y cómo es que se han implementado.

2. Un repaso sobre la teoría de los autómatas que incluyen gramáticas formales, máquinas del estado finito y máquinas de Turing.

El capítulo 3 se enfoca completamente en la implementación realizada, desde la construcción hasta la forma en que las diferentes partes se encuentran organizadas, los puntos importantes son:

1. Descripción y justificación del material musical utilizado así como su clasificación.
2. Planteamiento del algoritmo
3. Descripción de la gramática y diagrama de Markov para la sección determinista.
4. Programación

En el capítulo 4 se realiza una evaluación del trabajo; cuáles de los objetivos previstos se lograron de acuerdo a la metodología empleada y cuales fueron los factores que intervinieron para realizar algunas modificaciones y replanteamientos.

Se hace una reflexión de los alcances y se propondrán varias líneas de investigación que puedan darle continuidad a este trabajo.

En el capítulo 5 se elaboran conclusiones en base a una relación entre las metas propuestas y los resultados, comparaciones entre los resultados esperados y los resultados obtenidos, y una explicación de por qué las diferencias.

CAPITULO 2

ANTECEDENTES

2.1 Composición automática en la Música.

2.1.1 Antecedentes históricos

La búsqueda de métodos o algoritmos para generar música se remonta a muchos años atrás en la historia: desde las primeras búsquedas de relaciones matemáticas en la música de Pitágoras y sus seguidores (Alperu,1995), pasando por el juego *Musicalisches Wiirfelspiel* de Mozart (Roads, 1996) encontramos muchos procesos ingeniosos con éstos fines, ya incluso con la introducción del algoritmos y el establecimiento de reglas de las gramáticas formales (Chomsky 1957) se especulaba con la posibilidad de crear métodos prácticos para la generación automatizada de la música y el lenguaje. No obstante, debido a que muchos de éstos métodos, aunque simples, conllevan un gran número de operaciones numéricas, no es sino hasta el surgimiento de las computadoras cuando muchos de éstos tratados encontrarán su aplicación práctica.

Uno de los primeros trabajos formalmente realizados en la era de la informática es el de Hiller & Isaacson en 1958. Basado en modelos probabilísticos el sistema funciona generando notas de una manera pseudoaleatoria usando cadenas de Markov, de tal forma que van apareciendo notas que cumplen con cierta tendencia probabilística. (Lejaren; Hiller & Isaacson, 1958). Tenemos asi un sistema basado en *técnicas heurísticas* (Polya, 2004). Sin embargo, varios críticos a estos sistemas arguyen que éstos métodos excluían aquello relacionado con la expresividad y el contenido emocional de la música (Gómez-Zamaolla) aunque es de advertir que los mismos autores afirman que el sistema no se encontraba enfocado a cuestiones subjetivas, sino mas bien a dar los primeros pasos en la composición automatizada. A partir de este momento surgirán una serie de

trabajos que basan su funcionamiento en modelos probabilísticos como funciones de distribución y cadenas de Markov.

Posteriormente irán surgiendo nuevos métodos para la generación de melodías como el de Richard Moore en 1972 (Moore, 1998), éste da un paso adelante en los modelos probabilísticos e intenta humanizar un poco los procesos al introducir algoritmos y basado nuevamente en técnicas heurísticas.

No obstante, históricamente varios autores están de acuerdo en que los primeros trabajos en los que se utilizan técnicas de Inteligencia Artificial propiamente dicha son los trabajos de Rader (1974). Este propone un sistema basado en reglas de aplicabilidad que determinan bajo qué condiciones y en qué momento debe aplicarse cada una de las reglas generativas, teniendo en cuenta para ello los pesos asociados con cada una de ellas Estos son los principios básicos de una máquina de estado finito.

Simon and Sumner (1993), pioneros en inteligencia artificial, intentaron introducir gramáticas formales para describir, o intentar describir el lenguaje musical. Lo más interezante de éste proyecto es que diseñaron un método para identificar patrones más o menos implícitos en obras musicales. Lerdahl and Jackendoff (1983) también intentan plantear un modelo de la estructura musical desde la perspectiva de las ciencias cognitivas. A partir de entonces surgirán toda una familia de modelos basados en gramáticas generativas, que hoy en día siguen teniendo bastante relevancia en el modelado de sistemas relacionados con la música.

Son muchos los sistemas que se han ido introduciendo en el ámbito de la creación musical automatizada, y de hecho, cualquier técnica que aparece es candidata a ser aplicada a la música. No obstante, los problemas de fondo siguen siendo los mismos, y estos son el poder adecuar el algoritmo para que consiga dar resultados que, de entrada, son subjetivos y sujetos a demasiadas convenciones culturales y gustos personales.

Sin embargo hay que destacar, por su relevancia, los trabajos de Bharucha (1993) o Feulner (1993) que utilizan redes neuronales para analizar piezas musicales y reconstruir melodías y armonías de forma similar al modelo. Especial mención nos merece también el trabajo de Marvin Minsky (1981) quien resuelve el problema por medio de *agentes* de manera que la interacción entre varios de ellos produzca el resultado esperado. Cada uno de los agentes estaría especializado en una determinada habilidad o técnica, teniéndose a su vez agentes de orden superior coordinando el funcionamiento global.

2.1.2 Métodos en la composición automática

Los avances en las investigaciones en inteligencia artificial han permitido contar con diversos métodos para automatizar operaciones y tareas. Y cada día aparecen nuevas aportaciones que se suman a la lista.

Sin embargo, y con fines prácticos daremos una explicación breve de los métodos más utilizados y conocidos actualmente:

2.1.2.1 Métodos probabilísticos

Los métodos probabilísticos abarcan de igual manera un gran número de técnicas relacionadas con las teorías de probabilidad y estadística, sin embargo en su mayoría tienen en común en su funcionamiento un generador de números aleatorios y alguna función que delimite los resultados o bien, que genere una tendencia. Para esto son muy comunes las *funciones de distribución* o las *cadenas de Markov*.

De acuerdo con los resultados obtenidos entonces se elabora alguna tabla de relación entre los datos numéricos y el material musical que pueden ser sonidos, notas, frases, contenido rítmico, timbres y prácticamente cualquier otro elemento que se requiera.

Los métodos probabilísticos son quizás los más básicos de todos, sin embargo presentan enormes ventajas como su sencillez al programar, por lo general no requieren de grandes procesos lo que resulta en una gran velocidad de cómputo y bajo consumo de recursos, es por esto que muchos otros algoritmos más complejos emplean en alguno o varios de sus procesos éstos métodos.

En el capítulo 2.2 de éste trabajo se realizará una explicación más detallada de éste punto.

2.1.2.2 Gramáticas

Las gramáticas formales son un conjunto de reglas que son capaces de generar diversas posibilidades combinatorias dentro de un conjunto de elementos dado y cuyos resultados son conocidos como *lenguaje.*

Lenguaje natural: es aquel lenguaje que ha evolucionado con el tiempo para fines de la comunicación humana como el español y aunque se ajustan a reglas, tienen muchas variantes. Esto da como resultado que aumenten su complejidad al tratar de analizarlo desde el punto de vista lógico.

Lenguajes formales: están definidos por reglas gramaticales preestablecidas y se ajustan con rigor a ellas. Como ejemplo podemos citar la lógica y conjuntos, las matemáticas y, por supuesto, los lenguajes de programación.

La teoría de las gramáticas formales apareció en la década de 1950 cuando el lingüista Noam Chomsky publicó su libro titulado *Estructuras Sintácticas* (Chomsky 1957).

2.1.2.2.1 Cadenas

Una vez establecidas las reglas de cómo debe estar estructurado nuestro lenguaje, dado un conjunto de elementos y un estado inicial se pueden ir generando diferentes estados subsecuentes o combinaciones llamados cadenas (*w*).

Ejemplo:

Imaginemos una gramática con estas reglas:

1.- A → bA
2.- bA → c
3.- c → [finaliza]

La idea es sustituir el símbolo inicial de la izquierda por otros símbolos aplicando las reglas. El lenguaje al cual representa esta gramática es el conjunto de cadenas de símbolos que pueden ser generados de esta manera. en este caso, por ejemplo:

A → bA → bbA → bbbA → bbbbA → bbbbbA → bbbbc.

El elemento en mayúsculas es el símbolo *inicial*. Los elementos en minúsculas son símbolos *terminales*. Las cadenas de la lengua son aquellas que solo contienen elementos terminales.

El empleo de éstos métodos nos permiten además generar comportamientos al cambiar las reglas o los estados iniciales. Esto es lo que se conoce normalmente como *autómata*

Dado que la música se estructura en la forma de un *lenguaje*, las gramáticas formales y los autómatas gramaticales presentan grandes ventajas al utilizarlos como generadores automáticos de material musical.

2.1.2.3 Redes Neuronales

Las *redes neuronales* o *neurales* son algoritmos que comenzaron a desarrollarse en la década de los 50's en un intento por emular, de manera muy básica, al funcionamiento de las neuronas del cerebro.

Una red neuronal busca simular las propiedades observadas en los sistemas neuronales biológicos a través de modelos matemáticos recreados mediante mecanismos artificiales, estos pueden ser un circuito integrado, un conjunto de válvulas o un algoritmo informático. El objetivo es conseguir que las máquinas respondan de manera más o menos similar a como responde el cerebro.

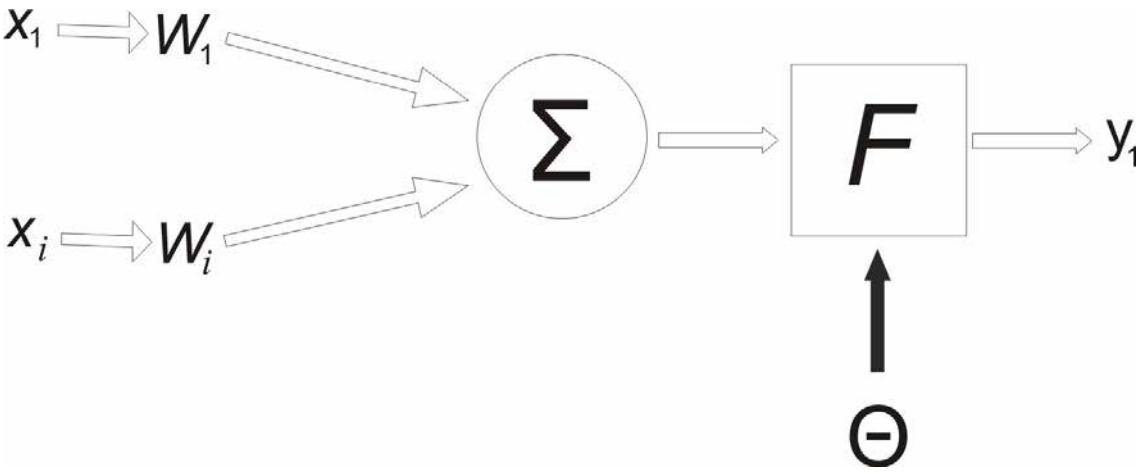
Estas redes se componen de unidades llamadas *neuronas*. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida. Esta salida viene dada por tres funciones:

1. Una *función de propagación* ó función de excitación que, generalmente consiste en la sumatoria ($\boldsymbol{\Sigma}$) de cada entrada ($\boldsymbol{x}$) multiplicada por el peso ($\boldsymbol{W}$)de su interconexión. Si el peso es positivo, la conexión se denomina *excitatoria*; si es negativo, se denomina *inhibitoria*.

2. Una *función de activación (*$\boldsymbol{\Theta}$*)*, que modifica a la anterior. Puede no existir, siendo en este caso la salida la misma función de propagación.

3. Una *función de transferencia (**F**)*, que se aplica al valor devuelto por la función de activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que queramos darle a dichas salidas. Algunas de las más utilizadas son la sigmoide (para

obtener valores en el intervalo [0,1]) y la hiperbólica-tangente (para obtener valores en el intervalo [-1,1]).



Neurona con 2 entradas

Las Redes Neuronales Artificiales (RNA) tienen muchas ventajas, entre las más importantes tenemos:

- Aprendizaje: Tienen la habilidad de aprender mediante una *etapa de aprendizaje*. Esta consiste en proporcionar datos de entrada a la vez que se le indica cuál es la respuesta esperada.

- Autoorganización: Una RNA crea su propia representación de la información en su interior, liberando al usuario de esta tarea.

- Tolerancia a fallos. Debido a que una RNA almacena la información de forma redundante, ésta puede seguir respondiendo aceptablemente aún si se daña parcialmente.

- Autocorrección: Una RNA puede manejar variaciones no críticas en la información de entrada, como señales con ruido u otros cambios en la entrada.

Entre otras ventajas encontramos que las RNA nos permiten la posibilidad de trabajar en tiempo real, no obstante aquí podemos encontrar

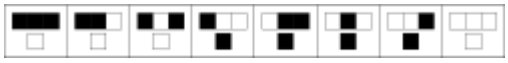
un problema inherente a este tipo de sistemas, y este es que, conforme la red se complica más y más la velocidad de cómputo puede ir disminuyendo o gastando mayores recursos

2.1.2.4 Autómatas celulares

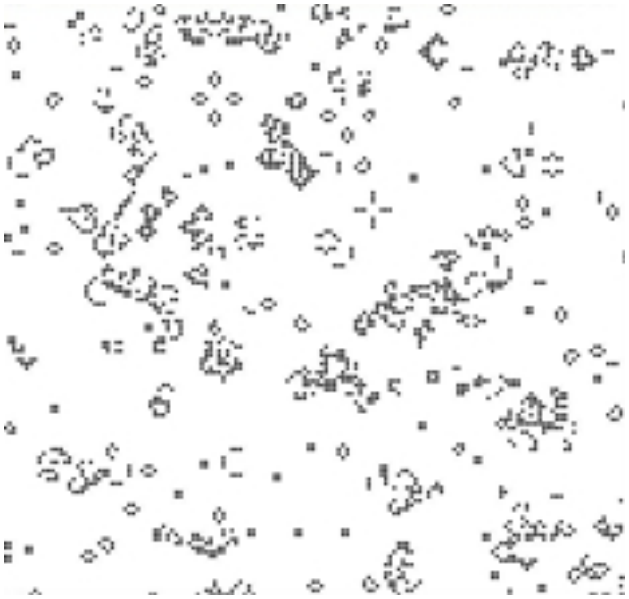
Un Autómata Celular es *un modelo matemático que modela a un sistema dinámico que evoluciona en pasos discretos* (Miranda, 2001). Es adecuado para modelar sistemas naturales que puedan ser descritos como una colección masiva de objetos simples que interactúen localmente unos con otros. Son sistemas descubiertos dentro del campo de la física computacional en la década de los 50's. La teoría de los autómatas celulares inicia en los años 40's con Konrad Zuse y Stanislaw Ulam. Zuse pensó en los “espacios de cómputo” (*computing spaces*), como modelos discretos de sistemas físicos. Sin embargo, es con John Von Neumann y su libro *Theory of Self-reproducing Automata* (1966) cuando se ponen en práctica. Los autómatas celulares, tienen una relación muy cercana con autómatas gramaticales y las máquinas de turing en cuanto a que también contienen una serie de reglas (función de transición) y un estado inicial.

Estos sistemas se volvieron muy populares debido a que, al observar sus comportamientos, arrojan resultados muy similares a la forma en que muchos sistemas complejos actúan y evolucionan, como la propagación de bacterias, la distribución de las nubes, o el flujo de tráfico en una ciudad, por dar unos ejemplos. Pueden ser usados para modelar numerosos sistemas físicos que se caractericen por un gran número de componentes homogéneos y que interactúen localmente entre sí, y de hecho, cualquier sistema real al que se le puedan analogar los conceptos de "vecindad", "estados de los componentes" y "función de transición" es candidato para ser modelado por un autómata celular.

Quizás el ejemplo más famoso de un autómata celular es el *juego de la vida* de John Conway ( ) que consiste en una matriz de celdas donde cada una cambia su estado de acuerdo a una ciertas reglas que actúan en función del estado de las celdas vecinas.

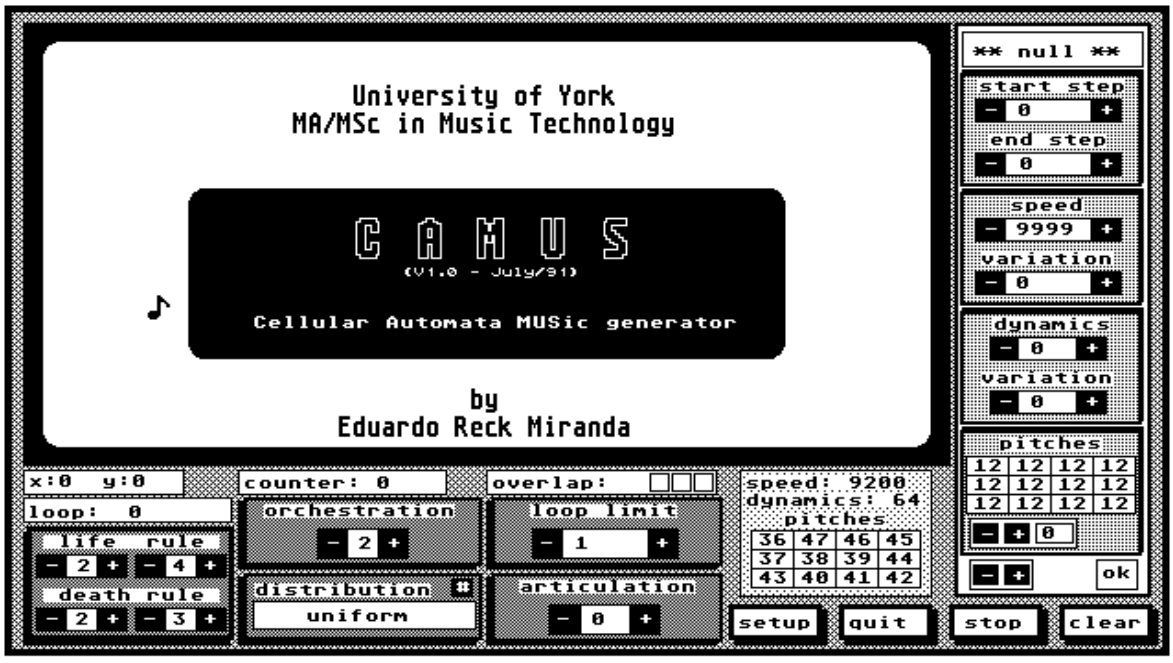


Comportamiento de un Autómata Celular



El Juego de la vida

Entre las aplicaciones musicales más conocidas que utilizan autómatas celulares se encuentran el CAMUS (Miranda, 2001) el cual utiliza autómatas celulares conocidos para ordenar una serie de notas y duraciones o patrones rítmicos.



2.1.2.5 Métodos cognitivos – Agentes

Los sistemas de agentes se pueden considerar sistemas mixtos ya que utilizan varios de los métodos anteriores e incluso incorporan otras disciplinas como la Psicología y la Sociología.

Curiosamente la idea de un sistema basado en agentes surge como una corriente alterna a otros sistemas de Inteligencia Artificial la cual busca resolver un problema, no por medio de la construcción de un gran sistema inteligente complejo, sino mas bien la simplificación de las tareas y su distribución entre sistemas más sencillos pero que cooperen entre sí, al estilo en que las células de un ser vivo trabajan o la manera en como algunos insectos sociales realizan tareas específicas y actúan como un gran organismo.

Los orígenes de la tecnología de agentes (Bradshaw 1996) (Ferber 1999) comienzan con la teoría de la Inteligencia Artificial Distribuida (IAD), ésta se utiliza para la resolución de problemas de forma distribuida. El concepto de agente como entidad computacional aislada surge de la programación orientada a objetos y evoluciona en programación orientada agentes, la cual supera muchas de las limitaciones que la primera presentaba.

Para evitar confusiones, es preciso aclarar la diferencia entre un sistema basado en *agentes* y un *sistema multiagente* (Jennings, 1998).

Un sistema basado en *agentes* es aquel que utiliza el concepto de agente como mecanismo de abstracción, pero aunque sea modelado en términos de agentes podría ser implementado sin ninguna estructura de software correspondiente a éstos. En este caso, hay que hacer un mayor esfuerzo de abstracción, identificar mecanismos de aprendizaje, coordinación, negociación, etc.

Un *agente* es un sistema informático, situado en algún entorno, dentro del cual actúa de forma autónoma y flexible para así cumplir sus objetivos. Además de la interacción con el medio, un agente se caracteriza, utilizando la definición de (Woldridge, 1995), por las siguientes propiedades:

• Autonomía: tiene la capacidad de actuar sin intervención humana directa o de otros agentes.

• Sociabilidad: capacidad de interaccionar con otros agentes, utilizando algún *lenguaje*.

• Reactividad: un agente está inmerso en un determinado entorno llamado hábitat, del que percibe estímulos y ante los que debe reaccionar de cierta forma en un tiempo preestablecido.

• Iniciativa: un agente puede tomar decisiones de cómo actuar para cumplir sus objetivos.

Por otro lado, un sistema *multiagente* es aquel que se diseña e implementa pensando en que estará compuesto por varios agentes que interactuarán entre sí, de forma que juntos permitan alcanzar la funcionalidad deseada (Bussman, 1993). Los sistemas multiagente son adecuados para solucionar problemas para los que hay múltiples métodos de resolución (Chu-Carroll, 1995). Por ello, uno de los aspectos básicos en estos sistemas es la interacción entre los diferentes agentes que los forman, la definición de modelos concretos de cooperación, coordinación o negociación entre los agentes.

Los SM tratan sobre la coordinación inteligente entre una colección de *agentes* autónomos, cómo pueden coordinar sus conocimientos, metas, propiedades y planes para tomar una decisión o resolver un problema (Bond y Gasser, 1988). Los SM se caracterizan por una serie de propiedades:

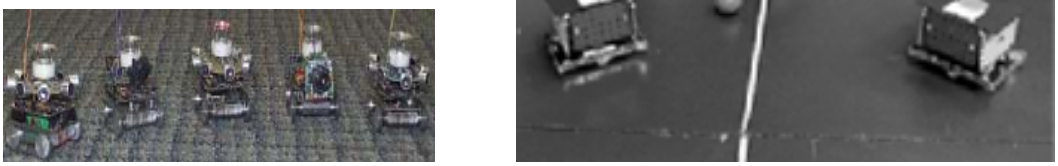
• Descripción de Competencias. A partir de un problema concreto, se define el problema en términos de tareas, subtareas y sus relaciones. De este modo se determina cómo resolver el problema, cómo distribuirlo entre los diferentes agentes y las interacciones entre los mismos. La correcta definición del problema es determinante para realizar su descomposición, su distribución y para fijar el comportamiento de los agentes a la hora de resolver un problema cooperativamente.

• Modelos de Agentes Conocidos. El modelo de agentes conocidos se utiliza para predecir el comportamiento de otros agentes, planear y coordinar actividades locales de acuerdo con una meta global. Dado que los Sistemas Multiagente se componen de agentes que interaccionan con un ambiente externo y que en este entorno existen otros agentes, resulta necesario representar la información relativa a los otros agentes. El modelo de agentes contiene información que se puede utilizar para razonar sobre otros agentes y para satisfacer sus necesidades de comunicación. Esto ofrece la ventaja de que un sistema de esta tipo se vuelva autoconfigurable.

• Comunicación. Este es uno de los puntos clave de los sistemas multiagente. Es de vital importancia establecer un buen sistema de comunicación entre los agentes para establecer la definición del problema, su descomposición y distribución de las tareas. A través de ellos tienen lugar todas las interacciones entre los agentes. Cuando se establecen claramente los métodos de comunicación entre agentes, se dice que se ha establecido un *Lenguaje*. Existen varios lenguajes bastante conocidos, como el Protocolo de Comunicación del Cooper-A (Sommaruga, 1989).

• Organización. Los distintos tipos de organización incluyen: la organización centralizada, donde la toma de decisiones la realiza un solo agente; organización tipo mercado, donde las interacciones se regulan mediante ofertas y contratos; la comunidad plural, donde las soluciones locales de un agente son refinadas globalmente por el resto de los agentes; la comunidad con reglas de comportamiento, donde un conjunto de agentes con capacidades muy diferentes se controla mediante un protocolo de interacción.

• Puntos de Interacción de un Agente. En general, cada agente de un grupo puede tener la necesidad de comunicar a otros agentes información procesada, o bien, pedirles información que no esté a su disposición. Esto lo suele realizar el agente mediante llamadas a funciones de petición, o de envío, de información. A todos los puntos presentes en la aplicación, donde se produce una interacción con otros agentes, se les denomina Puntos de Interacción (IP). Los IP son importantes porque son las localizaciones iniciales para las actividades cooperativas entre los agentes.



Agentes robóticos

2.2 – Antecedentes técnicos

En éste capítulo nos detendremos un poco para explicar de manera más detallada algunos puntos que serán importantes para una mejor comprensión de nuestro trabajo.

2.2.1 – Probabilidades

Los métodos probabilísticos, por ser los más elementales se encuentran presentes en casi todos los sistemas de Inteligencia Artificial.

2.2.1.1 – Teoría de la probabilidad

La teoría de probabilidad es la rama de la teoría matemática que modela los fenómenos aleatorios, también llamados estocásticos.

Un fenómeno aleatorio es aquel que, a pesar de realizar un experimento bajo las mismas condiciones determinadas, tiene como resultados posibles un conjunto de alternativas, como el lanzar una moneda al aire. Estos fenómenos se anteponen a los *fenómenos determinísticos* en los cuales cada vez que se repiten las mismas condiciones se llega al mismo resultado.

2.2.1.2 Probabilidades discretas y continuas

En relación al los resultados que se pueden obtener podemos hacer una división entre probabilidades discretas y continuas.

Una *Probabilidad Discreta* es aquella donde la variable puede tomar valores de un conjunto, ya sea finito o infinito, pero cuyos componentes son contables o *discretos*. Por ejemplo, canicas en una bolsa.

Se define como: X : Ω Є N

Donde:

X - es la función

Ω – es el espacio de todos los resultados posibles

N – es el conjunto de números naturales

Una *Probabilidad continua* es aquella donde la variable es una función *X* cuyos resultados no son contables. Como por ejemplo, los puntos en una línea.

Se define como: X : Ω Є R

Donde:

X - es la función

Ω – es el espacio de todos los resultados posibles

R – es el conjunto de números reales

2.2.1.3 Funciones de distribución

Una Función de distribución es una función *F*(*x*) que representa los resultados que se van obteniendo en un experimento aleatorio.

Entre las varias aplicaciones que nos otorgan las funciones de distribución es que, además de poder registrar y predecir los resultados de eventos aleatorios, nos permiten delimitar las tendencias a partir de un generador de números aleatorios.

2.2.1.4 Probabilidad condicional

Se llama probabilidad condicional a la probabilidad de que un suceso se cumpla habiéndose cumplido ya otro. Se anota "probabilidad de A sabiendo que B se ha cumplido" de la siguiente manera:

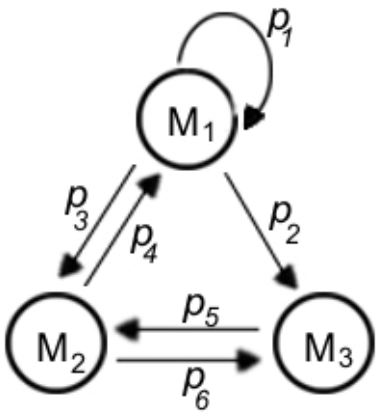
$$p_B(A) \text{ ó } p(A\backslash B)$$

2.2.1.5 Cadenas de Markov

Una cadena de Markov es una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. Las cadenas de este tipo “recuerdan" el último evento y esto condiciona las posibilidades de los eventos futuros. Esta dependencia del evento anterior distingue a las cadenas de Markov de las series de eventos independientes, como tirar una moneda al aire o un dado.

Las propiedades de Markov, nos dicen que s*i se conoce la historia del sistema hasta su instante actual, su estado presente resume toda la información relevante para describir en probabilidad su estado futuro ( ).*

Las cadenas de Markov se pueden representar de varias maneras, una de ellas es mediante un diagrama de estados



Donde tenemos varios estados M_n y las probabilidades p_n de que exista un cambio a otro estado.

Otra forma de representación que resulta sumamente útil es usar una *matriz de transición*

	M1	M2	M3
M1	*p1*	*p2*	*P3*
M2	*p4*		*p6*
M3		*P5*	

Donde la suma de las probabilidades en una fila o una columna es igual a 1

2.2.2 – Autómatas

2.2.2.1 – Gramáticas

En el capítulo 2.1.2.2 hemos explicado ya brevemente que una gramática formal consta de un conjunto finito de reglas de reescritura y un conjunto de elementos *terminales* y *no terminales* junto con un símbolo de inicio. Se basa en la composición de cadenas en términos de *frase* donde cada una de ellas finaliza al aparecer un elemento *Terminal*.

Formalmente hablando, una gramática, se define como una cuádrupla (V, T, R, S) donde:

V: conjunto finito de no terminales.
T: conjunto finito de terminales.
R: conjunto finito de reglas de reescritura.
S: inicio.

Una vez definidos los elementos anteriores se pueden ir generando diferentes combinaciones conocidos como cadenas.

Ejemplo:

Sea:

V {A}

T {a,c}

R { 1.- A → bA

2.- bA → c }

S = A

Podemos construir la cadena:

A → bA → bbA → bbbA → bbbbA → bbbbbA → bbbbc.

Nótese que al construir la cadena se observa que:

a) El inicio debe contener por lo menos un elemento *no terminal*.
b) La cadena se completa hasta la aparición de un elemento *Terminal*.

Veamos un ejemplo con valores para clarificar:

Sean:

V= {ENUNCIADO, SUJETO, PREDICADO, ARTICULO, SUSTANTIVO,
ADJETIVO, VERBO, ADVERBIO, PUNTO }

T = { “el”, “perro”, “toro”, “viejo”, “negro”, “corre”, “come”, “bien”, “mucho”, “.”}

R = { ENUNCIADO → SUJETO + PREDICADO + PUNTO
SUJETO → ARTICULO + SUSTANTIVO + ADJETIVO / ARTICULO + SUSTANTIVO
PREDICADO → VERBO + ADVERBIO / VERBO
ARTICULO → “el”
SUSTANTIVO → “perro” / “toro”
ADJETIVO → “viejo” / “negro”
VERBO → “corre” / “come”
ADVERBIO → “bien” / “mucho”
PUNTO → “.”

S = ENUNCIADO

Si comenzamos a sustituir podemos tener la siguiente combinación:

ENUNCIADO
→ SUJETO PREDICADO PUNTO
→ ARTICULO SUSTANTIVO ADEJTIVO VERBO ADVERBIO “.”
→ el perro negro come mucho.

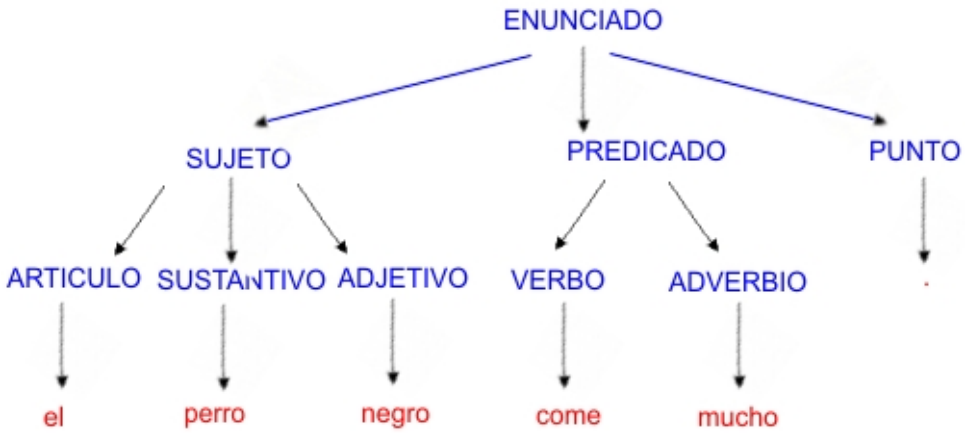
De ésta forma podemos obtener resultados como:

El toro come mucho.
El perro corre bien.
El toro corre.
El toro viejo corre bien.

2.2.2.1.1 - Árbol de derivación

Una manera sencilla y útil de representar una gramática es por medio de un *árbol de derivación* e cual nos permite ver con mayor claridad la forma en que se van dando las diferentes combinaciones.

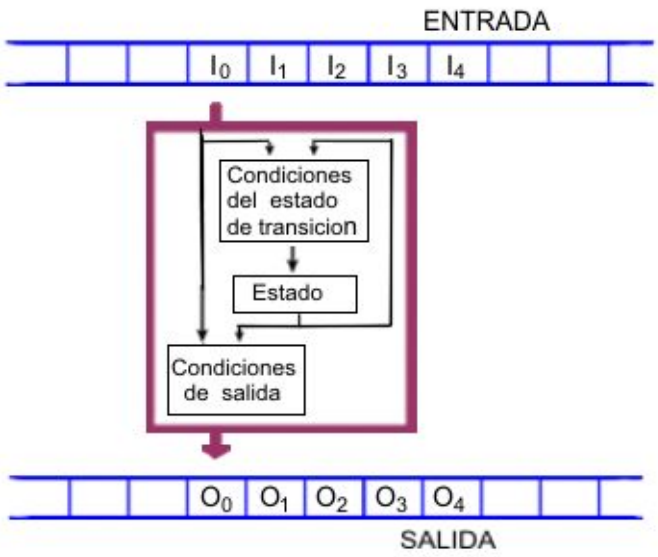
Ej : Usando el ejemplo anterior:



2.2.2.2 – Máquinas del estado finito

Una máquina del estado finito es un sistema que lee una secuencia de símbolos de entrada y escribe, en función de ciertas reglas, otra secuencia de símbolos de salida.

El lugar donde se almacenan los datos tanto de entrada como se salida se les suele llamar *cinta* de entrada y salida respectivamente, y por cada dato de entrada se escribe otro de salida.



Formalmente hablando, una maquina de estado finito puede definirse como un sistema: $N=(I,S,O,\lambda,\delta)$ donde:

I: son los símbolos de entrada.

S: representación de los estados que componen la maquina.

O: son los símbolos de salida.

δ : transformación de SXI (función de estado siguiente)

λ: transformación de SXI (función de salida).

Las máquinas del estado finito pueden representarse de dos maneras para un mejor análisis de cómo trabajan:

- **Diagramas de estado**

El diagrama de estado de una maquina de estado finito esta representado con un grafo rotulado donde los nodos son los estados δ de N.

Ejemplo:

Sea:

I={a,b}

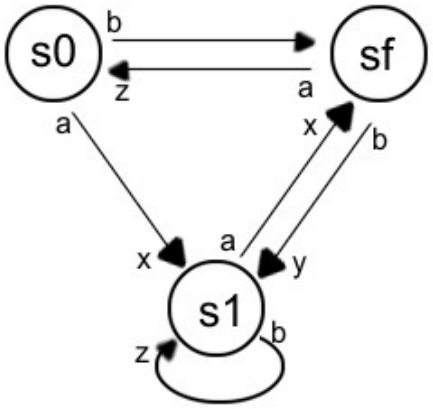
S={s_0,s_1,sf}

O={x,y,z}

δ= { (s0,a)=s1
(s1,a)=sf
(sf,a)=s0
(s0,b)=sf
(s1,b)=s1
(sf,b)=s1 }

λ = { (s0,a)=x
(s1,a)=x
(sf,a)=z
(s0,b)=y
(s1,b)=z
(s0,b)=z }

Tenemos el siguiente diagrama de estado:



- **Tabla de estado**

Alternativamente, la maquina se puede representar por su tabla de estado, que para cada combinación de estado y entrada proporciona el próximo estado y el símbolo de salida.

Siguiendo los datos de nuestro ejemplo anterior tenemos:

	a	b
s0	s1, x	sf
s1	sf, x	s1,z
sf	s0, z	s1,y

2.2.2.3 – Autómatas finitos

Un *autómata finito* es, de hecho, una *máquina de estado finito* que presenta *estados de aceptación* y *rechazo* en lugar de una salida.

Esta maquina puede colocarse en un numero finito de estados, uno de los cuales es el estado inicial y por lo menos uno es el estado de aceptación. A este dispositivo esta unido un flujo de entrada por medio del cual llega una secuencia de los símbolos de un alfabeto determinado.

Formalmente se define un autómata finto como un sistema:
M=(Σ, S, A, s0, F) donde:

Σ: es un conjunto finito de símbolos de entrada, llamado *alfabeto*
S : es un conjunto finito de estados internos.
A : es un subconjunto de S cuyos elementos se llaman estados de aceptación
s0: es un estado inicial de S
F: es una función de estado próximo de S*A en S.

Para su representación introduciremos a nuestro diagrama un doble círculo para representar los estados de aceptación en los nodos correspondientes.

La introducción de los estados de aceptación permite que el sistema contenga ciertas condicionales que regulen su comportamiento de manera más precisa al ir construyendo una cadena.

Ejemplo:

Sea:

Σ ={a , b}, símbolo de entrada

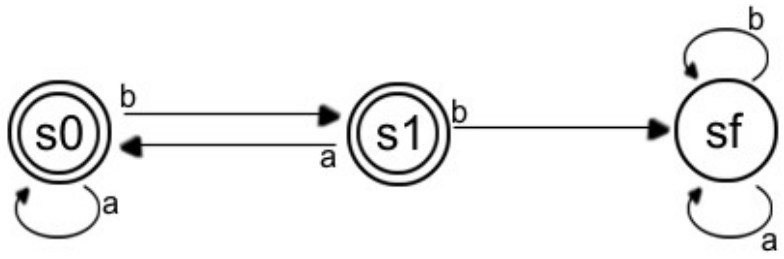
S={s0,s1,s2}, estados

A={s0,s1}, estados de aceptación

s0: estado inicial

F= { (s0,a)=s0
(s0,b)=s1
(s1,a)=s0
(s1,b)=s2
(sf,a)=sf
(sf,b)=sf }

Tenemos:



Junto la siguiente tabla:

	a	b
s0	s0	s1
s1	S0	sf
sf	sf	sf

Podemos notar que por ejemplo que este ejemplo que presentamos evita que en la construcción de una cadena existan dos entradas b sucesivas:

s0, b → s1 y como: s1 Є A entonces es aceptado

s1, b → sf y como: sf no Є A entonces es rechazado

2.2.2.3.1 Autómatas Finitos Deterministas

Esta máquina es un autómata finito cuyo estado de llegada está unívocamente determinado por el estado inicial y el carácter leído por el autómata.

La determinación de cual será la transición que ocurra al recibir un símbolo depende del *mecanismo de control* de la maquina, programado para conocer cual debe ser el nuevo estado dependiendo de la combinación del estado actual y el símbolo de entrada. Un AFD se denomina *completo* cuando tiene una transición por cada estado y cada carácter del alfabeto. E *incompleto* cuando no cumple esta condición.

Formalmente un autómata finito determinista es un sistema:
AFD=(Σ, S, δ, s0, A) donde:

- Σ: es el alfabeto de la maquina.
- S: es un conjunto finito de estados.
- δ: es una función de transición de SXS a S
- s0: estado inicial Є S
- A: conjunto de estados de aceptación que es un subconjunto de S.

Se dice que un diagrama de transiciones es determinista si cumple las siguientes condiciones:

- Cada estado de estos diagramas solo tiene un arco que sale para cada símbolo del alfabeto.
- Debe existir por lo menos un arco para cada símbolo del alfabeto.

2.2.2.3.2 Autómatas Finitos No Deterministas

Esta maquina es un autómata finito muy similar a los deterministas salvo que la transición que se ejecuta en una etapa dada puede ser incierta, es posible aplicar ninguna, una, o mas de una, transiciones mediante el mismo símbolo de entrada, como sucede con una maquina que no esta completamente definida.

Un autómata finito no determinista también puede o no tener más de un nodo inicial.

De manera formal: AFN= (Σ, S, p, s, A) donde:

- Σ es el alfabeto de la maquina
- S es un conjunto finito de estados.
- p es una relación de transiciones que es subconjunto de SXS XS
- s es el estado inicial (un elemento de S)
- A: es el conjunto de estados de aceptación (un subconjunto de S).

2.2.2.4 Autómatas de pila

Un Autómata de pila es un autómata finito que, además, cuenta con un sistema de almacenamiento de información que se puede llamar mas tarde y que se conoce como *pila*.

Formalmente: AP= (S, Σ, Γ, T, i, F) donde:

S: es una colección finita de estados

Σ: es el alfabeto de la maquina

Γ: es la colección finita de símbolos de pila

T: es una colección finita de transiciones

i: es el estado inicial (es un elemento de S)

F: es la colección de estados de aceptación (es un subconjunto de S)

2.2.2.5 Máquinas de Turing

En 1900, dentro del Congreso Internacional de Matemáticas de París, David Hilbert (1862-1943) planteó en una conferencia el llamado *entscheidungsproblem*. Esto es, la cuestión de si las matemáticas son decidibles. El problema plantea la interrogante de si hay un método definido que pueda aplicarse a cualquier sentencia matemática y que nos diga si esa sentencia es cierta o no.

En respuesta a esta interrogante, Alan Turing (1912-1954) desarrollo un modelo en 1936 que fue explicado en su artículo llamado *On Computable Numbers* (Copeland, 2004) y que se conoce como la *Máquina de Turing*. Esta responde el problema planeado por Hilbert y, de hecho, se puede probar matemáticamente que para cualquier programa de computadora es posible crear una máquina de Turing equivalente.

A pesar de sus enormes aplicaciones prácticas, las máquinas de turing no era más que una construcción teórica hasta que con el desarrollo de las computadoras pudieron hacerse realmente funcionales volviéndose uno de los pilares más importantes para definir lo que se conoce como *procedimiento mecánico* o más popularmente, *algoritmo.*

La MT es un autómata finito con la particularidad que está conformado por un cabezal de lectura y escritura que puede moverse hacia delante o hacia atrás, y una cinta infinita.

Formalmente: (S, Δ, Γ, δ, s, h) donde:

S: es un conjunto finito de estados (por lo menos uno de inicio y uno de parada).

Δ: es un símbolo denominado blanco Є Γ, el único que se puede repetir infinitamente

Γ: es un conjunto finito de símbolos de cinta (alfabeto)

δ: función de transición de la maquina

si: es el estado inicial Є S

h: es el conjunto de estados finales de aceptación o *parada*. (h nunca=si)

La función de transición esta representada además de la siguiente manera:

δ : (p, x)=(q, y, L/R) : si el estado actual es *p* y el símbolo actual es *x*, pasar al estado *q,* reemplazar *x* por *y* y mover la cabeza a la izquierda (L) o derecha (R).

Aunque en teoría, una máquina de Turing puede realizar cualquier proceso sin encontrar limitaciones respecto al espacio de almacenamiento, en la aplicación real esto no sucede. Es por eso que una máquina de turing se ve limitada por las capacidades de almacenamiento del computador, no obstante, su capacidad es superior a otros tipos de autómatas, como el autómata finito, o el autómata con pila, o igual a otros modelos con la misma potencia computacional.

Otro factor que no debemos olvidar, es que las máquinas de Turing funcionan con tiempos discretos llamados tiempos de *reloj* los cuales, en la mayoría de los casos están determinados por los tiempos del procesador del computador

Ejemplo:

Consideremos una MT definida por:

S={s1,sf}
Γ ={x ,y, Δ}
si = s1
h = sf

Una función de transición δ dada por:

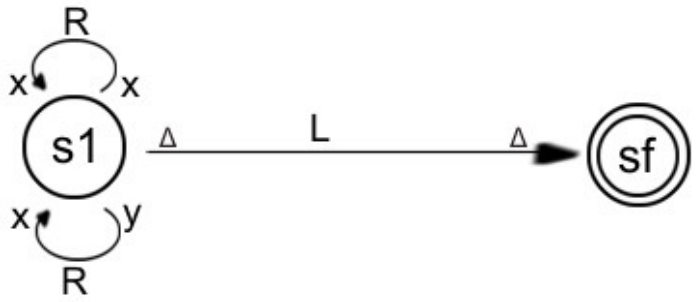
(s1, x) = (s1, x, R)
(s1, y) = (s1, x, R)
(s1, Δ) = (sf, Δ,L)

Y una cadena w = { x y y x Δ }

Las transformaciones serán:

reloj	estado	cadena
1	s1	x y y x Δ
2	s1	x y y x Δ
3	s1	x x y x Δ
4	s1	x x x x Δ
5	s1	x x x x Δ
6	sf	x x x x Δ

El Diagrama de estado estará definido:



2.3 Antecedentes del algoritmo (Punto de partida)

En el año 2003, como parte de un experimento para una revista electrónica de ciencia ficción (www.ciencia-ficcion.com.mx), se desarrollo, en colaboración con el Ing. José Luís Ramírez Gutiérrez, un sistema cuyo objetivo era crear algunas estrofas escritas o “poemas” relacionados a la ciencia ficción.

Funcionamiento

El funcionamiento general de ésta máquina es el siguiente:

1. Almacena información previa en una serie de tablas, y *arreglos.* Tenemos por ejemplo, sustantivos, verbos, adverbios, proposiciones, cada uno de ellos es guardado con un valor numérico que asigna un *peso* (*w*), éste valor jugará un papel importante al final del proceso para establecer tendencias en el estilo.

2. En base al banco de datos almacenados elije la forma en que los elementos iniciales deberán de ir

3. Un Autómata finito comienza la tarea de armar las diferentes oraciones y estrofas de acuerdo a la gramática establecida, realiza las sustituciones hasta obtener una cadena de elementos terminales. Tal y como se explicó en el ejemplo del capítulo 2.2.2.1. La forma en la que ésta va eligiendo los elementos terminales es de manera aleatoria.

4. Una vez construida la oración entra en funcionamiento una máquina de Turing para modificar los datos de acuerdo a ciertas reglas.

Ej.:

- Si el artículo es singular modifica el sustantivo a singular o modifica el artículo a plural.
- Si el sustantivo es femenino cambia el artículo a femenino o cambia el sustantivo a masculino.
- Si es pregunta invierte el orden de sujeto predicado.

Al final del trabajo imprime el resultado en pantalla con opción de que el usuario califique positiva o negativamente, de acuerdo a la respuesta se varía el *peso* (*w*) de cada uno de los componentes para comenzar a establecer una tendencia.

Gracias a que el sistema no requiere operaciones matemáticas complicadas éste permite utilizar un lenguaje de programación sencillo, en esta caso javascript que además facilita su uso vía Internet.

Fragmento del script:

```
var frases=''; var poema='';
random_count=0; nsentences=0;
nwords1=0; nwords2=0; nwords3=0; nwords4=0; nwords5=0; nwords6=0; nwords7=0;
nwords8=0; nwords9=0;

// Ejemplo de Tabla:

list1=''; list2=''; list3=''; list4=''; list5=''; list6=''; list7=''; list8=''; list9='';
frases = "9, 2!"
+"\n2, 2 Y 2."
+"\n1s 4ad!"
+"\nEL 5o 1 3a 6o AL 1."
    +"\nEL 5o 1 3a 6o AL 1 5o."
    +"\nEL 2 ES UN 1 5o."
    +"\nEL 1 4a COMO UN 5o 1."

function random(maxnum) {
 r=Math.floor(Math.random()*maxnum)+1;
 if (r>maxnum) r=maxnum;
 return r;
}

function count_lines(str) {
 len=str.length;
 nword=1;
 for (i=0; i<len; i++) {
  if (str.charAt(i)=="\n") {
   nword++;
  }
 }
 if (str.charAt(len-1)=="\n") nword--;
 return nword;
}

function get_line(str, lnum) {
 len=str.length;
 iline=1; ichar=0; jchar=-1;
 for (i=0; i<len; i++) {
  if (str.charAt(i)=="\n") {
   iline++;
```

```
   if (iline==lnum) {
    ichar=i+1;
   } else if (iline==(lnum + 1)) {
    jchar=i-1;
    if (str.charAt(jchar)=="\r") {
     jchar--;
    }
    break;
   }
  }
 }
 if (jchar<0) jchar=len-1;
 s="";
 for (i=ichar; i<=jchar; i++) {
  s = s + str.charAt(i);
 }
 return s;
}

function make_poem() {
 poema="";
 count_all_lines();
 nlines=random(3)+2;
 for (ilin=1; ilin<=nlines;ilin++) {
  make_poem_line()
 }
 while (poema.indexOf("ÓNs")>0) poema=poema.replace("ÓNs", "ONes");
 while (poema.indexOf("ÁNs")>0) poema=poema.replace("ÁNs", "ANes");
 while (poema.indexOf("Ls")>0) poema=poema.replace("Ls", "Les");
 while (poema.indexOf("Ns")>0) poema=poema.replace("Ns", "Nes");
 while (poema.indexOf("Zs")>0) poema=poema.replace("Zs", "Ces");
 return "<i>"+poema.toLowerCase()+"</i>";
}

function make_poem2() {
 poema="";
 count_all_lines();
 nlines=random(3)+2;
 for (ilin=1; ilin<=nlines;ilin++) {
  make_poem_line2()
 }
 while (poema.indexOf("ÓNs")>0) poema=poema.replace("ÓNs", "ONes");
 while (poema.indexOf("ÁNs")>0) poema=poema.replace("ÁNs", "ANes");
 while (poema.indexOf("Ls")>0) poema=poema.replace("Ls", "Les");
 while (poema.indexOf("Ns")>0) poema=poema.replace("Ns", "Nes");
 while (poema.indexOf("Zs")>0) poema=poema.replace("Zs", "Ces");
 return poema.toLowerCase();
}

function make_poem_line() {
 pattern=get_line(frases, random(nsentences));
 lenpat=pattern.length;
 for (ichr=0; ichr<lenpat; ichr++) {
  chr = pattern.charAt(ichr);
  if ((chr>='1') && (chr<='9')) {
   if (chr=='1') {
    wrd=get_line(list1, random(nwords1));
   } else if (chr=='2') {
    wrd=get_line(list2, random(nwords2));
   } else if (chr == '3') {
```

```
    wrd=get_line(list3, random(nwords3));
   } else if (chr == '4') {
    wrd=get_line(list4, random(nwords4));
   } else if (chr == '5') {
    wrd=get_line(list5, random(nwords5));
   } else if (chr == '6') {
    wrd=get_line(list6, random(nwords6));
   } else if (chr == '7') {
    wrd=get_line(list7, random(nwords7));
   } else if (chr == '8') {
    wrd=get_line(list8, random(nwords8));
   } else if (chr == '9') {
    wrd=get_line(list9, random(nwords9));
   } else {
    wrd='';
   }
   poema=poema+wrd;
  } else {
   poema=poema+chr;
  }
 }
 poema=poema+"<BR>\n";
}

function make_poem_line2() {
 pattern=get_line(frases, random(nsentences));
 lenpat=pattern.length;
 for (ichr=0; ichr<lenpat; ichr++) {
  chr = pattern.charAt(ichr);
  if ((chr>='1') && (chr<='9')) {
   if (chr=='1') {
    wrd=get_line(list1, random(nwords1));
   } else if (chr=='2') {
    wrd=get_line(list2, random(nwords2));
   } else if (chr == '3') {
    wrd=get_line(list3, random(nwords3));
   } else if (chr == '4') {
    wrd=get_line(list4, random(nwords4));
   } else if (chr == '5') {
    wrd=get_line(list5, random(nwords5));
   } else if (chr == '6') {
    wrd=get_line(list6, random(nwords6));
   } else if (chr == '7') {
    wrd=get_line(list7, random(nwords7));
   } else if (chr == '8') {
    wrd=get_line(list8, random(nwords8));
   } else if (chr == '9') {
    wrd=get_line(list9, random(nwords9));
   } else {
    wrd='';
   }
   poema=poema+wrd;
  } else {
   poema=poema+chr;
  }
 }
 poema=poema+"<br/>\n";
}
```

Algunos resultados:

Ej. 1:

la oscura estrella ama desesperada a la luz.
la cálida nova ama desesperadamente a su ávida luna.
la muerta constelación mira calma a la luz.

el pequeño espacio ama tranquilo una luz limpia.
desintegra desesperado como un viejo láser.
todos los piratas desean lunas pequeñas, cálidas.
la muerte es un capitán limpio.
todas las chicas aman capitanes limpios, oscuros.

nunca confrontes un soldado.
explota quieto como una fría constelación.
piratas desintegrad!

Ej 2:

por qué viaja el monstruo?
el valor es un marciano ávido.
todas las lunas desean lásers ávidos, rudos.
los soldados matan como viejos piratas.
todos los soles miran lásers muertos, rudos.

nunca comandes un sol.
por qué explota la constelación?
la nave mata como una ruda nova.
las chicas matan como fríos robots.
amor, amor y coraje.

espaciopuertos explotad!
todos los espacios comandan chicas pequeñas, limpias.
la oscura nave confronta desesperada a la constelación.

Dado que la estructura del lenguaje escrito (español en este caso) presenta similitudes con el lenguaje musical, el sistema aquí descrito presenta un excelente punto de partida para poder desarrollar un autómata que trabaje con música.

CAPITULO 3

Desarrollo

3.1 – Marco Teórico

3.1.1 Sistemas a utilizar

Para este proyecto se ha optado por trabajar con autómatas finitos en lo que respecta al proceso central de nuestro algoritmo, esto es debido a su simplicidad, su facilidad de programación y porque, siendo elementos básicos, es posible ir incrementando posteriormente el poder de la máquina sin tener que rediseñarla desde el principio. Además otorga la posibilidad de poder programar elementos adicionales en forma *modular*, es decir ir agregando *objetos*, esto abre la posibilidad de hacer un sistema compuesto mucho más complejo.

3.1.1.1. Plataformas

Para el desarrollo de este proyecto se han decidido utilizar las siguientes aplicaciones y lenguajes debido a que presentan diversas ventajas:

3.1.1.1.1. MAX/MSP

Esta será plataforma *Host* o anfitriona en la que se trabajará, la ventaja es que, debido a que es un programa diseñado especialmente para aplicaciones de MIDI y Audio en *tiempo rea,l* evita el trabajo de programar objetos o instancias de éste tipo.

Otra ventaja importante es la compatibilidad. En las últimas versiones de MAX se ha incorporado un compilador que permite crear *plugins* que puedan ser ejecutados desde la suite *Pluggo*®; o bien la comunicación con

otros programas vía REWIRE®; esto ofrece la ventaja del trabajo en conjunto con otros programas como PROTOOLS, NUENDO, ó REASON.

3.1.1.1.2. Javascript

Una de las desventajas de utilizar MAX/MSP es que, debido a su interfase gráfica, la programación de ciertas tareas se hace sumamente complicada, y en ocasiones, ni siquiera existe algún objeto prefabricado para lo que se requiere. Sin embargo, en sus últimas versiones, MAX/MSP ha incorporado también la posibilidad de escribir objetos en javascript y Java, lo que nos da la oportunidad de programar en código fuente.

Javascript es un lenguaje de programación concebido para programar aplicaciones para Internet, lo cual ofrece las enormes ventajas de la sencillez y simplicidad en la programación, la compatibilidad con muchos otros sistemas y aplicaciones, además de que no requiere de compiladores.

3.1.1.1.3. Java

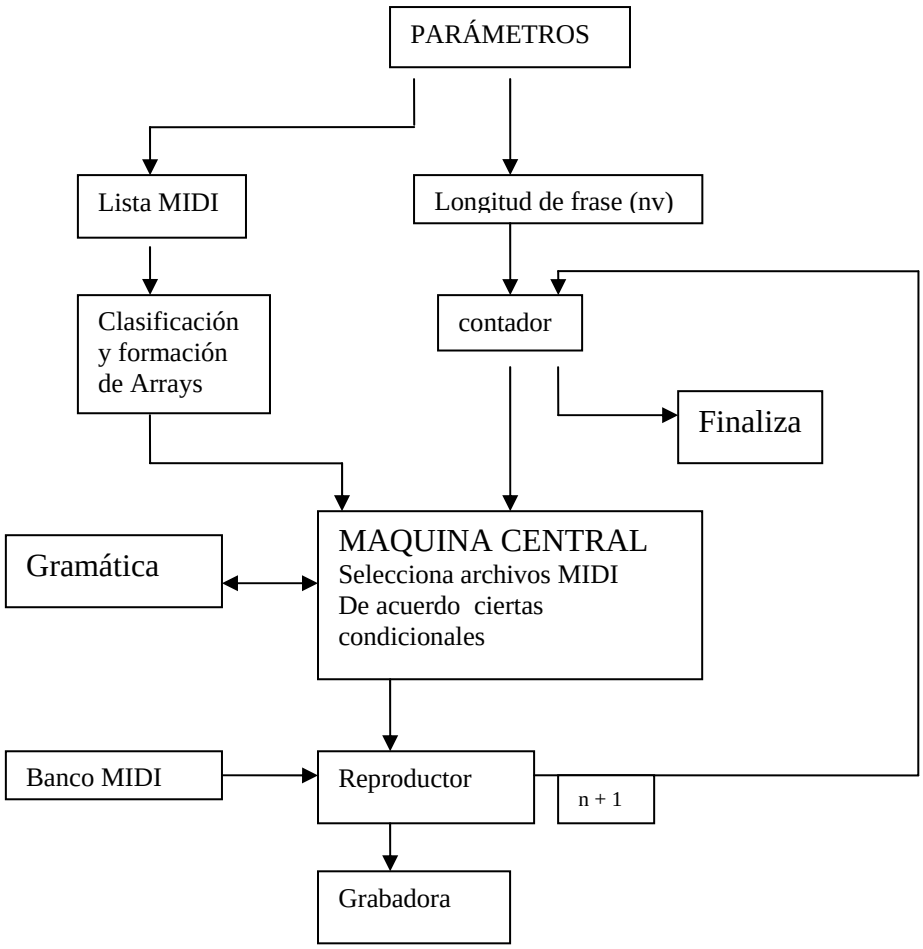
Java representa, por decirlo así, un paso más adelante en lo que a lenguajes de programación se refiere, el uso de Java se ha extendido en los últimos años debido a las enormes ventajas que ofrece. Es un lenguaje bastante poderoso y, debido a que se han creado un gran número de librerías gratuitas para este lenguaje, sus alcances se han multiplicado. Una de las grandes ventajas es que las aplicaciones creadas en Java pueden correr de igual manera en diversos sistemas operativos. La incorporación de soporte Java en MAX/MSP nos permite crear objetos que normalmente no posee y que requerirían demasiado trabajo y tiempo programar nativamente.

3.2 Desarrollo del Sistema

El sistema que se ha desarrollado involucra varias de las técnicas que se describieron en el capítulo 2, el corazón de ésta es un autómata finito (que llamaremos *Maquina Central* o *MC*) sinembargo también involucra aspectos aleatorios y cadenas de Markov. Es por eso que al describirla se considerará como una máquina *compuesta*.

3.2.1 Descripción general

El esquema general de funcionamiento de la máquina esta explicado de la siguiente manera:



Esto es:

- Se recibe una serie de parámetros por parte del usuario entre los que se encuentran:
    - Nombre del banco que se va a utilizar
    - Número de veces que el proceso se repetirá antes de finalizar.

- Recibidos estos parámetros, el sistema crea:
    - Un registro de todos archivos MIDI que se encuentran en el banco, los clasifica y los coloca en una de varias listas con las que creará una serie de arreglos o *arrays;* de éstos se elegirán los datos pertinentes posteriormente.
    - Una variable (*nv*) que representa el número de veces que la rutina deberá trabajar antes de detenerse

- La Máquina Central selecciona los archivos de acuerdo a ciertas condicionales y en relación con la gramática establecida, las decisiones dependen de igual manera de los datos con los que se cuente.

- Reproduce el sonido correspondiente que busca y carga desde la carpeta correspondiente; una vez finalizada la reproducción envía un mensaje para repetir la rutina, conforme se va acercando al *nv*. Cambia su comportamiento y eventualmente finaliza. Al resultado sonoro final, desde que comienza el proceso hasta que termina, se le designará el nombre de *Frase Resultante*.

- Opcionalmente, existe la posibilidad de grabar el resultado en un archivo MIDI que llamaremos *archivo MIDI Resultante*.

3.2.2. Funcionamiento de la Máquina Central

3.2.2.1. Contador

A diferencia de la forma en que un constructor autómata de texto, como el constructor de poemas descrito en el Capítulo 2.3, en la creación de frases musicales tenemos mayor libertad de combinación, a menos, por supuesto, que quisiéramos imitar una forma musical estricta lo cual no es nuestro caso.

No obstante esta ventaja, surge un problema inherente, y este es que se debe asignar de entrada una longitud a la *Frase Resultante* para poder calcular en qué parte del total se encuentra operando el sistema. Para esto se asigna una variable *nv* que está directamente relacionada con el número de veces que la rutina se ejecutará.

3.2.2.2. Definición de la gramática

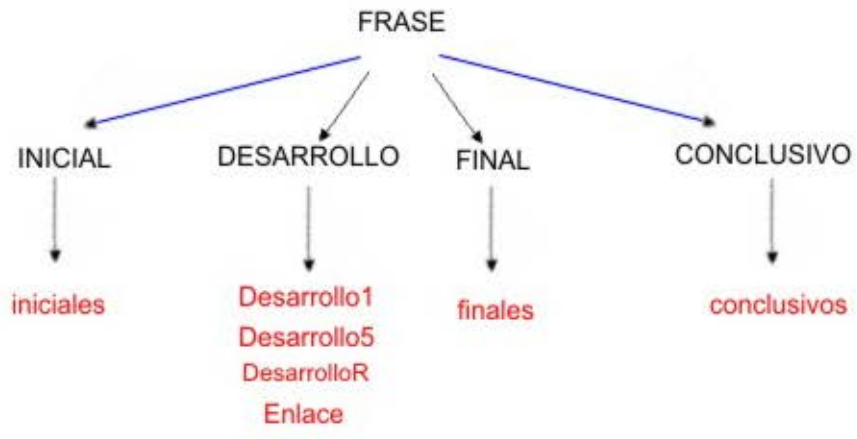
Este autómata posee una gramática (V, T, R, S) definida por:

V= {FRASER, INICIAL, DESARROLLO, FINAL, CONCLUSIVO}

T = { *iniciales*, *desarrollo11, desarrollo15, desarrollo55, desarrollo51, desarrolloR, enlaces, finales11,finales51*, *conclusivos* } <Banco MIDI variable >

R = { FRASER → INICIAL + DESARROLLO + FINAL + CONCLUSIVO
INICIAL → *iniciales*
DESARROLLO → (*desarrollo11, desarrollo15, desarrollo55, desarrollo51, desarrolloR, enlace*)
FINAL → (*finales51, finales 11)*
CONCLUSIVO → *conclusivos*
}

S = FRASER



Como se mencionó, la construcción de la frase estará directamente relacionada con el tamaño total preasignado de la frase, dado por la variable *nv* y una variable llamado contador *c* que va incrementando su valor en 1 después de completar un ciclo. De ésta manera tenemos que:

Si C = 1 → INICIAL

C = NV-xf → FINAL (*xf* es variable)

C == NV → CONCLUSIVO

C > NV → *Finaliza*

Else → DESARROLLO

En la sección DESARROLLO se definirá un comportamiento más complejo, ya que existe una subdivisión de elementos de desarrollo y una interrelación directa con los eventos FINAL y CONCLUSIVO, éstos son:

DESARROLLO11 (D11)

DESARROLLO15 (D15)

DESARROLLO51 (D51)

DESARROLLO55 (D55)

DESARROLLOR (DR)

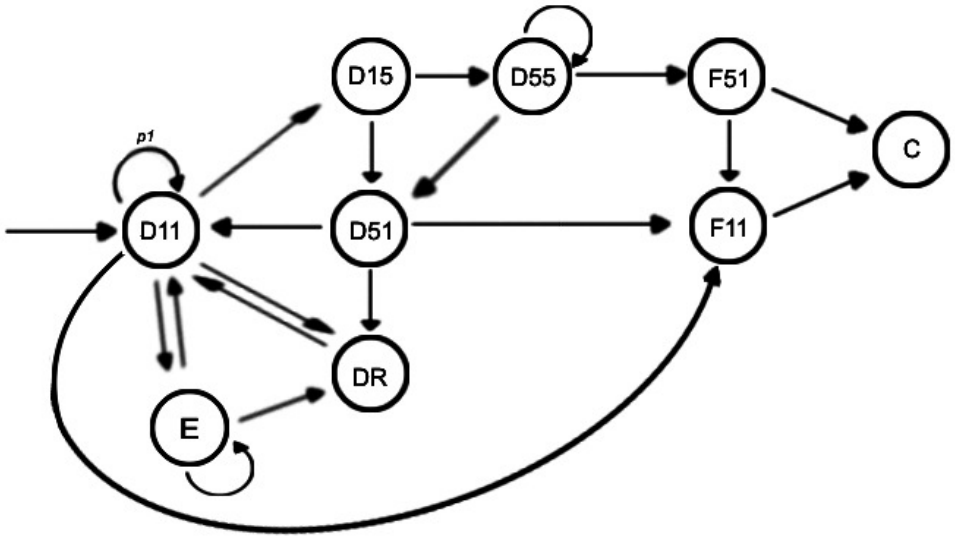
ENLACE (E)

FINAL11 (F11)

FINAL51 (F51)

CONCLUSIVO (C)

La forma en que estos elementos se encuentran relacionados se encuentra definida por el siguiente diagrama de Markov:



Donde: Diagrama Markov

Pn = es la probabilidad de que exista un cambio de estado (cada una de las flechas posee un elemento p, solo se ha dibujado la primera)

Las tablas de probabilidades se encuentran incluidas en la programación y son cargadas junto a los parámetros iniciales, son totalmente modificables.

3.2.2.4 Polifonía

Para crear un resultado más completo en cuanto a las voces que pueden reproducirse simultáneamente, todo el sistema anterior se ha triplicado para que cada uno de ellos haga uso de una base de datos independiente, correspondiente a una instrumentación o una voz diferente. Teóricamente éste se puede multiplicar hasta donde las capacidades de cómputo de nuestros ordenadores nos lo permitan, sin embargo en el caso de éste sistema se ha asignado una polifonía de 3 voces.

3.2.3. Bases de datos

3.2.3.1. Selección de banco de datos MIDI

La selección de bancos MIDI es una tarea que requiere un cuidado muy especial, ya que en este punto intervienen muchos elementos subjetivos que, no obstante su difícil control, influyen completamente en el resultado sonoro.

Dado que el resultado musical, desde al punto de vista estético no es la parte central de éste proyecto, como ya se comentó al principio, no nos detendremos mucho en discutir los pormenores de la razón musical por la cual se seleccionaron, sin embargo, dada su importancia, explicaremos las razones prácticas por las que se consideró que los bancos utilizados son convenientes por su facilidad para ejemplificar la forma en que el autómata trabaja.

Entendiendo que una de las ventajas del sistema es que puede ser alimentado con cualquier clase de banco y operar de manera satisfactoria

3.2.3.2. Banco utilizados

En esencia, el sistema está pensado para poder operar de manera satisfactoria si se le alimenta con cualquier tipo de banco. Esto da la

oportunidad de alimentarlo con una serie de bancos que pueden irse agregando y funcionar de manera adecuada, esto ofrece la posibilidad de expandir el uso del sistema a diferentes estilos musicales.

No obstante, para ejemplificar el funcionamiento se han elegido 2 tipos de bancos que son jazz y barroco.

3.2.3.2.1. Jazz

El banco de jazz nos ofrece amplias ventajas, debido a la naturaleza de éste género facilita la improvisación y resulta idóneo desde el punto de vista operacional del sistema.

3.2.3.2.1. Barroco

Aunque éste género posee una estructura más rígida y reglas muy estrictas, en algunos casos ofrece posibilidades de improvisación que resultan, al igual que el jazz, adecuadas para nuestro sistema.

Se tiene conciencia que el estilo barroco sigue reglas de armonía y contrapunto muy estrictas, no es el objetivo de éste sistema realizar obras barrocas perfectamente acordes con estas reglas, hacerlo implicaría modificarlo a tal grado que quedaría demasiado especializado en éste género, lo cual es algo no deseable si se quiere construir un sistema práctico para todo tipo de música. Tenemos entonces que este banco se trata más bien de una recombinación de fragmentos de obras barrocas tomadas de algunos trabajos de J.S. Bach.

3.2.3.3. Polifonía

Para que el sistema pueda reproducir hasta el total de 3 voces independientes, es necesario crear de antemano una base de que tenga asignados de antemano las voces que se utilizarán. En el caso de los bancos de jazz, estas voces corresponden a piano, percusión y bajo; para el caso de los bancos de barroco se ha utilizado, hasta el momento una sola voz.

3.2.3.4. División

En lugar de alimentar al sistema con una serie de obras y dejar que éste se encargue de dividir los fragmentos se ha optado por hacer una división “manual” de las mismas, esto es, que en éste punto ha hecho su intervención el factor humano de quien ha realizado la división. Esto ofrece la desventaja de la intervención de factores subjetivos que no podemos controlar pero ofrece también las ventajas de que el resultado presentará mayor coherencia, evitamos truncar fragmentos importantes, y nos permite un mayor control en cuanto a la clasificación de los bancos.

3.2.3.5. Clasificación y nomenclatura

La clasificación de los elementos, especialmente aquellos presentes en la sección de DESARROLLO, obedece a la necesidad de reducir incongruencias, sobre todo en los ámbitos de tempo y tonalidad. Para que nuestro sistema sepa como clasificarlos se ha seguido la siguiente nomenclatura:

XYabcddd.mid

Donde:

X – Es el tipo de voz que puede ser:

- P – voz principal
- B – voz secundaria
- D – voz tercera

Y – Indica el tipo de elemento Terminal al que pertenece:

- I – Iniciales
- D – Desarrollo
- R – Reposo
- F – Finales
- C – Conclusivos

ab – Son indicadores que indican un cambio de estado del archivo y nos servirán para reducir las incongruencias, la relación de éstas están explicadas

en el diagrama de Markov. Aunque es posible agregar más elementos en los bancos utilizados, sólo usamos el 1 y el 5.

A manera de ejemplo podemos mencionar nuestro banco Barroco, en el cual se utiliza esta nomenclatura para indicar en qué tonalidad comienza y termina el elemento, p. ej:

11 indica que empieza y termina en el mismo tono (Tónica)

15 indica que empieza en un tono y termina en la Dominante (modulación)

Aunque en el caso del barroco utilizamos estos indicadores para señalar un cambio en la tonalidad, éstos pueden utilizarse para marcar cambios de dinámica, cambios de instrumento, cambios de carácter, cambios de efecto, etc.

c – es otro número indicador de estado, en el caso de los bancos aquí utilizados nos indica el tempo, se utilizarán los dígitos del 1 al 9, por lo que consideraremos el 5 como un estado intermedio.

ddd – es un número de 000 a 999 que nos indica el número de archivo.

Ejemplo:

PD155110.mid

P – Es la voz principal

D – es un elemento de desarrollo

15 – comienza en estado 1 y va al estado 5

5 – posee un estado x con valor de 5

110 - es el archivo número 110

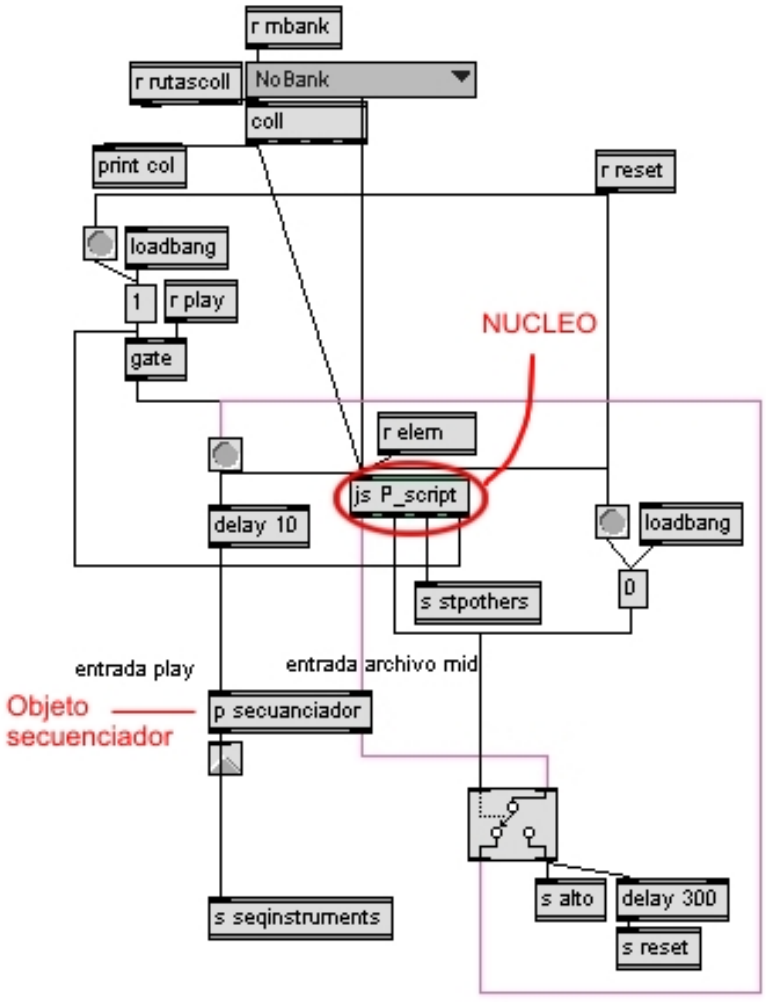
Lectura de los archivos

Se ha programado un algoritmo que es capaz de leer y clasificar de forma autónoma los archivos MIDI, de ésta manera todos los archivos pueden colocarse en una carpeta y no importa el número de ellos el sistema puede crear sus propias bases de datos, éstas son creadas en un archivo de texto.

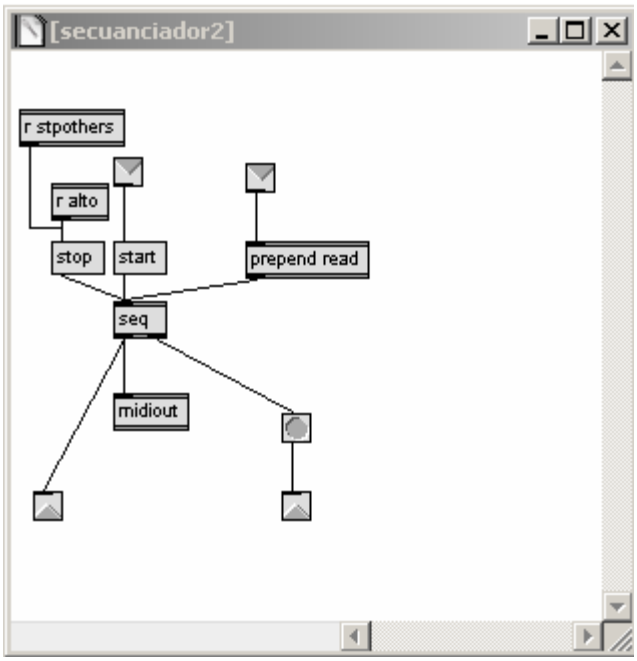
3.3 Programación

3.3.1 Programación en MAX/MSP

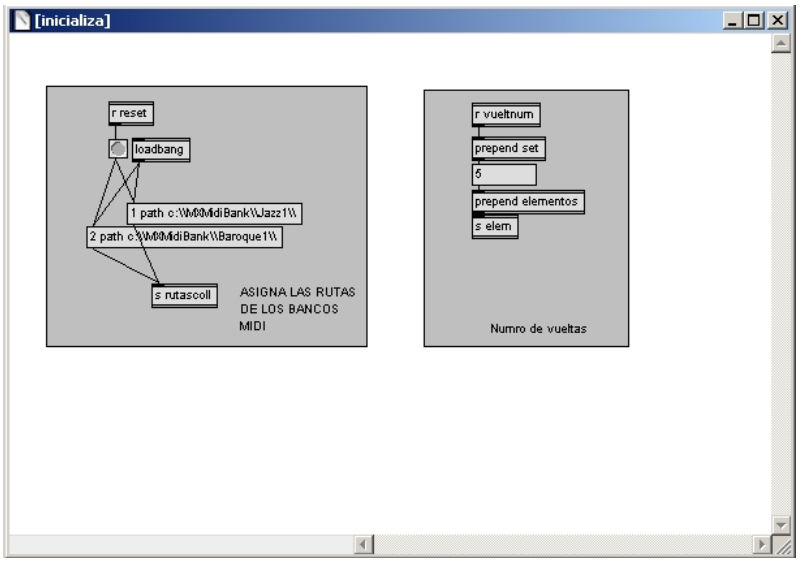
Como se mencionó, se realiza una programación en MAX/MSP para aprovechar las facilidades que éste software ofrece. Esta se encuentra diseñada de la siguiente manera:



El objeto secuenciador se encuentra estructurado de la forma:



De la misma manera tenemos un patch creado para establecer los los valores predeterminados, así como asignar el lugar donde se encuentran los archivos que, para evitar conflictos, se establecen con rutas absolutas.



Algunos de estos valores, como el número de cicloss se envían al script, no obstante el script ya contiene algunas variables predeterminadas para evitar posibles errores.

```
// Aqui se definen los valores defalut
    function elementos (elem) {
        nvueltas = elem;
    }

// Aqui se define la funcion de reset de los valores
  function reset () {
        nvueltas = 30;
        contador = 1;
        outlet (1,0);
  }

// Aqui se define el folder de los archivos midi
   function path (p){
          pth = p;
    }
```

3.3.2 Núcleo

3.3.2.1 Lector

Como se mencionó, se incluye un algoritmo sencillo que realiza la lectura de los bancos y los clasifica automáticamente, éste proceso se lleva a cabo en el núcleo, dentro del script.

```
// Aqui se leen lor Arrays del archivo de texto
  function readlines(s) {
        var f = new File(s);
        var i,a,c;
        if (f.isopen) {
                ci=0; cd1=0; cd5=0; ce=0; cf=0; cc=0;
                while (a=f.readline()) {
                        if (a.substr(0,1)=='[') {
                                b=a;
                        } else {
```

```
                                if (b=='[iniciales]') {
                                        pifilenames[ci] = a;
                                        ci++;
                                } else if (b=='[desarrollo1]') {
                                        pd1filenames[cd1] = a;
                                        cd1++;
                                } else if (b=='[desarrollo5]') {
                                        pd5filenames[cd5] = a;
                                        cd5++;
                                } else if (b=='[enlace]') {
                                        pefilenames[ce] = a;
                                        ce++;
                                } else if (b=='[final]') {
                                        pffilenames[cf] = a;
                                        cf++;
                                } else {
                                        pcfilenames[cc] = a;
                                        cc++;
                                }
                        }
                }
                f.close();
        } else {
                post("No se pudo abrir el archivo: " + s + "\n");
                outlet(3,0);
        }
```

3.3.2.2 Máquina central

La programación del núcleo del sistema, se realiza en javascript, con el siguiente código fuente:

```
// Aqui se aciva el porceso (núcleo del sistema)

  function bang () {
        post ("contador p: ", contador);
        if (contador==1) {
                outlet (0,pth+pifilenames[random(pifilenames.length - 1)]);
        } else if (contador==nvueltas) {
                outlet (0,pth+pcfilenames[random(pcfilenames.length - 1)]);
                outlet (2,"bang");
        } else if (contador==nvueltas-1) {
                outlet (0,pth+pffilenames[random(pffilenames.length - 1)]);
        } else {
                if (random(PE)==1)  // PE es la probabilidad de que ocurra un enlace (si
es 5 es un 20%)
                        outlet (0,pth+pefilenames[random(pefilenames.length - 1)]);
                else {
                        if (last_tone==5) {

                                if (random(PD5)==1)   {
                        // PD5 es la probabilidad de que elija un elemento de transición
                                        xxx=random(pd5filenames.length - 1);
                                        last_tone=pd5filenames[xxx].substr(3,1);
                                        outlet (0,pth+pd5filenames[xxx]);
                                } else {
                                        xxx=random(pd1filenames.length - 1);
                                        last_tone=pd1filenames[xxx].substr(3,1);
                                        outlet (0,pth+pd1filenames[xxx]);
                                }
                        } else {
                                xxx=random(pd1filenames.length - 1);
                                last_tone=pd1filenames[xxx].substr(3,1);
                                outlet (0,pth+pd1filenames[xxx]);
                        }
                }
        }

    if (contador>=nvueltas){
        outlet (1,1);
        contador=1;
     } else {
        contador = contador + 1;
       }

  }
```

3.3.3 Polifonía

Para que el sistema sea capaz de reproducir las tres voces el proceso anterior se triplica y se agrupa en un solo objeto de MAX. Sin embargo, las salidas de los reproductores MIDI se agrupan para poder escuchar el resultado conjunto y para dar la posibilidad de grabar una secuencia MIDI resultante que

sea la suma de las tres. Además, se ofrece la posibilidad de exportar el archivo para su utilización posterior e incluso ser utilizado por otra aplicación.

3.3.4. Interfaz de usuario

Para facilitar el funcionamiento es imprescindible realizar una interfaz de usuario lo mas estructurada posible, de otra manera, el manejo de la aplicación puede ser tan complicada que resultaría poco útil desde el punto de vista práctico.

La interfaz entonces debe permitir controlar todos los parámetros asignables por el usuario, sin que éste tenga que introducirse en el funcionamiento interno del programa para hacerlo funcionar, este es el aspecto más importante cuando se realiza la planificación.

En el desarrollo de software comercial una interfaz accesible constituye uno de los elementos que definirán el éxito o fracaso del software desarrollado, aún cuando internamente su desempeño sea excepcional.

Un aspecto tal vez menos importante, pero que no debe ser descuidado es cuidar que el aspecto visual de la interfaz permita observar los controles de manera ordenada y coherente con las diferentes funciones.

Y es que una interfaz amigable facilita al usuario el poder operar el software de manera sencilla y eficiente, sin necesidad de una capacitación especial o instrucciones complicadas.

La interfaz de esta aplicación se encuentra dividida en 3 secciones:

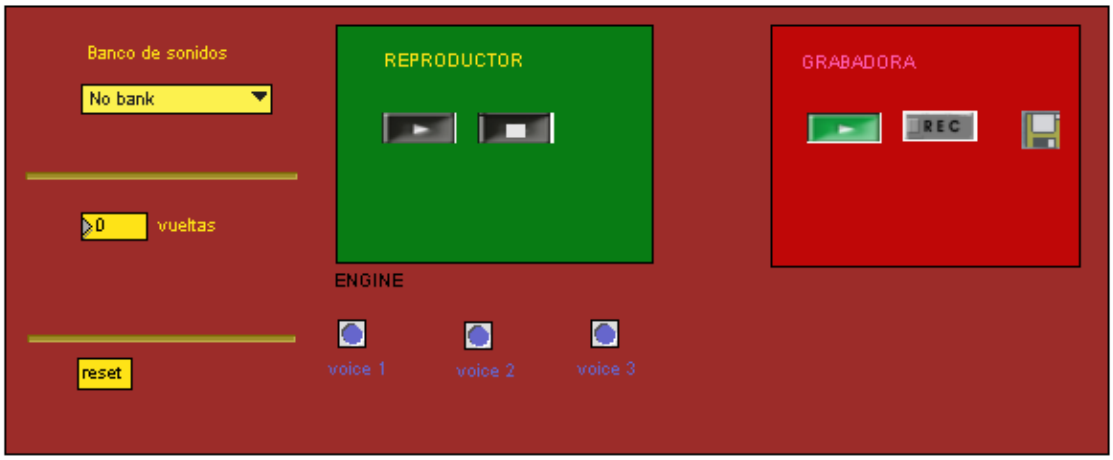
- Parámetros (Settings) – Donde se elije el banco a utilizar, el número de ciclos (longitud de la *Frase*) o bien si se permite que el sistema calcule la longitud; y un botón de reinicialización (reset).

- Reproducción – Donde se incluyen botones de reproducción (play) y parado (stop). Se incluyen también indicadores de que voces se encuentran activas.

- Grabación – Se incluyen botones de grabación (record), parado (stop) y reproducción (play) del archivo resultante, además de una opción para salvar (save) el archivo MIDI resultante.

En el proyecto se han introducido elementos extras en la programación en MAX para que esto sea posible. Tenemos así un menú que permite elegir el banco MIDI, un menú donde podemos asignar la duración de la melodía con opción de asignar duración aleatoriamente. Sección de reproducción y grabación con controles y un botón de reset para cargar los valores predeterminados.

Se han incluido también indicadores que nos permiten ver qué voces se encuentran operando.



3.3.5. Compatibilidad

Como se mencionó, MAX/MSP ofrece además la ventaja de poder exportar los archivos como instrumentos VST®. Tenemos entonces que nuestra programación puede llamarse desde algún programa anfitrión (o *host)* como NUENDO o PROTOOLS e insertarse como un instrumento, o bien desde el mismo MAX/MSP vía REWIRE® .

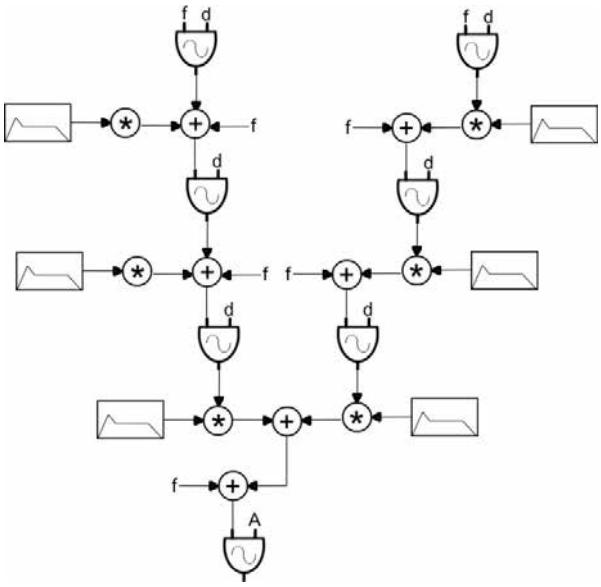
3.4 Implementación del autómata en un sintetizador FM

La razón por la cual se ha decidido aplicar el sistema a un sintetizador FM es para conseguir que los parámetros que comúnmente se encuentran estáticos en sintetizadores de éste tipo, como la frecuencia modulante o el índice de modulación, tengan un comportamiento dinámico en el tiempo, generando espectros mas interesantes desde el punto de vista sonoro.

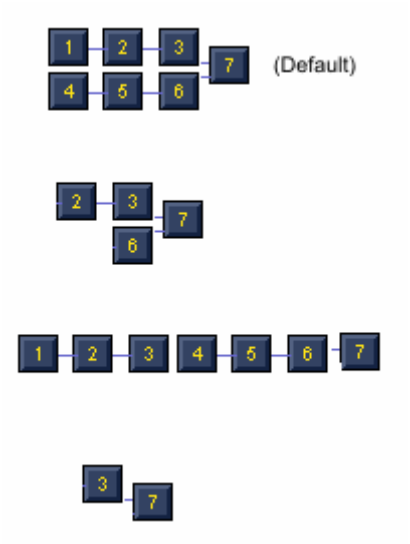
La implementación del sistema en el control de parámetros de un sintetizador FM requiere de realizar ajustes tratando de no alterar el funcionamiento básico del sistema mismo.

Lo primero es contar con un sintetizador FM al que se pueda insertar el autómata. Para esto se ha hecho uso de un sintetizador programado en MAX/MSP que consta de 1 oscilador principal y 6 osciladores modulantes que pueden interconectarse de manera variable. Cada unote estos 6 osciladores ofrece la opción de asignar una frecuencia, un índice de modulación (*d*) y una amplitud controlada por una envolvente.

De manera predeterminada el sistema de osciladores se encuentra interconectado de la siguiente manera:



Se observa que se tiene un oscilador portador ó *carrier* y seis osciladores modulantes. Hay que mencionar que, debido a la manera en la que el sintetizador se encuentra diseñado, éste permite cambiar la disposición de los osciladores 2-6 en muchas otras combinaciones por ejemplo:



3.4.1. Incorporación del autómata

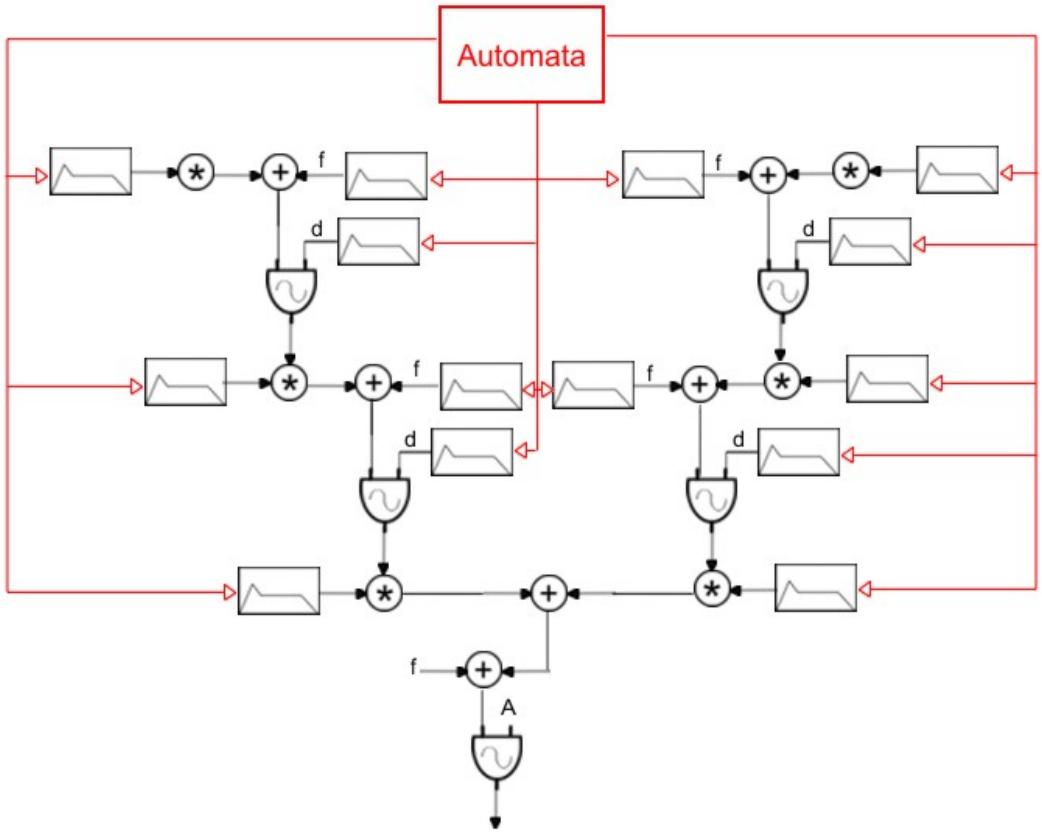
El autómata se incorpora al sistema para que éste realice la elección de los parámetros iniciales como la interconexión del sistema y además controle de manera dinámica y en tiempo real la frecuencia modulante y el índice de modulación.

Uno de los puntos que deben de tratarse con cuidado al hacer que estos parámetros varíen en el tiempo es que éstos deben permanecer con el mismo comportamiento en caso de que la frecuencia portadora varíe; esto es, conforme nos movamos por la escala musical. De no cuidarse esto resultaría en la pérdida total de la “afinación” musical de nuestro sintetizador.

Por supuesto que aún así habrá, en ocasiones, alguna pérdida significativa de temperamento, sin embargo dado que el propósito fundamental de nuestro sistema es crear espectros dinámicos esto no será considerado como un efecto indeseable.

La manera en la que se optó por resolver la cuestión antes mencionada es almacenar el movimiento de los parámetros en forma de una envolvente. Tenemos entonces que el autómata graficará una envolvente en el tiempo, de acuerdo a alguna función matemática y una vez creadas podemos variar nuestra frecuencia portadora en la escala musical.

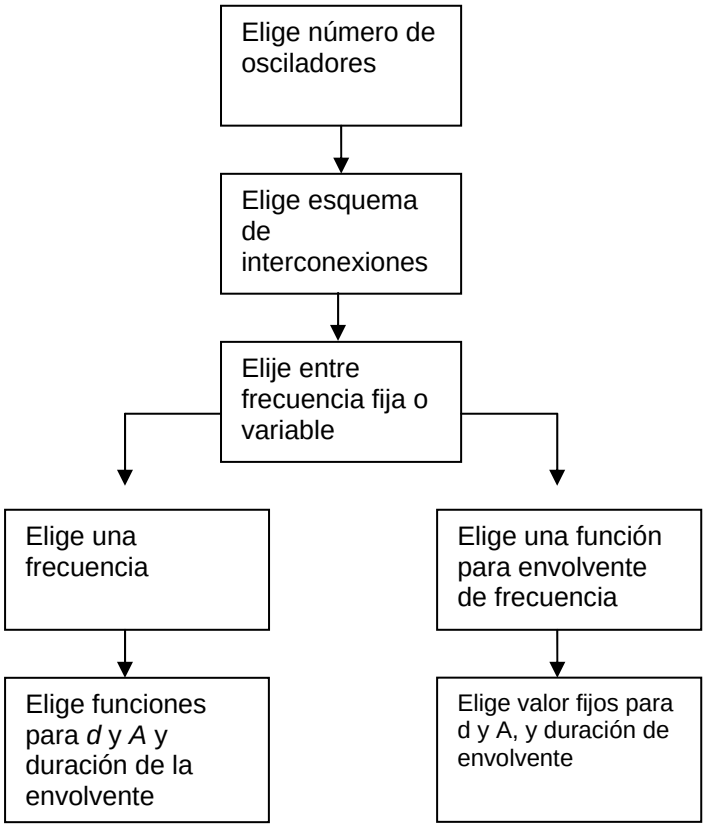
Tendremos entonces que nuestro diagrama para el sintetizador quedara de la siguiente manera:



Observamos entonces que los valores que eran constantes como d y f, ahora varían de acuerdo a una envolvente. En el caso de la frecuencia el rango en *y* será de 0 a 10000 mientras que en d será de 0 a 128; la envolvente de amplitud que ya contaba con una envolvente será de 0 a 1.

3.4.2. Esquema general

El esquema general del funcionamiento del sintetizador FM es el siguiente:



3.4.3. Gramática

En un principio la intención era la incorporación del mismo autómata que en el generador de frases musicales, la diferencia radicaría que en vez de elegir archivos MIDI, elegiría funciones para graficar las envolventes.

Sin embargo, las diversas modificaciones que sufrió en ésta adecuación han llevado al planteamiento de una gramática diferente y un proceso probabilístico distinto, menos complejo, pero conteniendo exactamente los mismos principios de un autómata finito.

La gramática ahora no hace referencia a la estructura musical sino a la interconexión entre osciladores y qué parámetros serán estáticos y cuales variables en el tiempo.

Tenemos entonces que la gramática GFm (V, T, R, S) está definida por:

V= {GENERAESPECTRO, FRECUENCIAF, INDICE, AMPLITUD}

T = {frecuenciaV, dfija, dvariable, Afija, Avariable}

R = { GENERAESPECTRO → FRECUENCIAF,/ frecuenciaV
FRECUENCIAF → INDICE + AMPITUD
INDICE → dfija / dvariable
AMPLITUD → Afija / Avariable
}

S = GENERAESPECTRO

Como se puede observar se ha decidido que cuando la frecuencia modulante sea variable, los demás parámetros queden estáticos. Eso es debido a que cuando variamos la frecuencia modulante el barrido en el espectro es tan radical que no es conveniente agregar más componentes al mismo, modificando los otros dos parámetros restantes evitando así posibles distorsiones.

De esta manera:

(F, v) → (d,f),(A,f)
(F, f) → (d,v),(A,v)

Donde:

F es frecuencia
d es el índice de modulación
A es la amplitud
v es variable
f es fijo

3.4.4. Automatización aleatoria de la interconexión entre osciladores

Hay que recordar que el sistema mencionado anteriormente se encuentra repetido 6 veces. Para darle mayor variedad y posibilidades combinatorias, se ha implementado un sistema adicional que decidirá cuantos de los 6 osciladores modulantes estarán activados así como la manera en la que éstos deberán de interconectarse. Sin embargo, para favorecer la variedad, se ha decidido que la forma más conveniente para implementar esto es a través de un generador aleatorio.

3.4.5. Funciones de graficación de envolventes

Las funciones que se han seleccionado para la graficación de envolventes son las siguientes:

Y = kx
Y = sen x
Y = random
Y = abs (arcsen x)
Y = log (x)
Y = arc tangente x
Y = 1/x

Se hace notar que estas funciones, con excepción de la aleatoria grafican curvas que son bastante suaves en sus pendientes lo cual facilita evitar distorsiones que puedan ocasionar problemas.

3.4.6. Interfaz gráfica

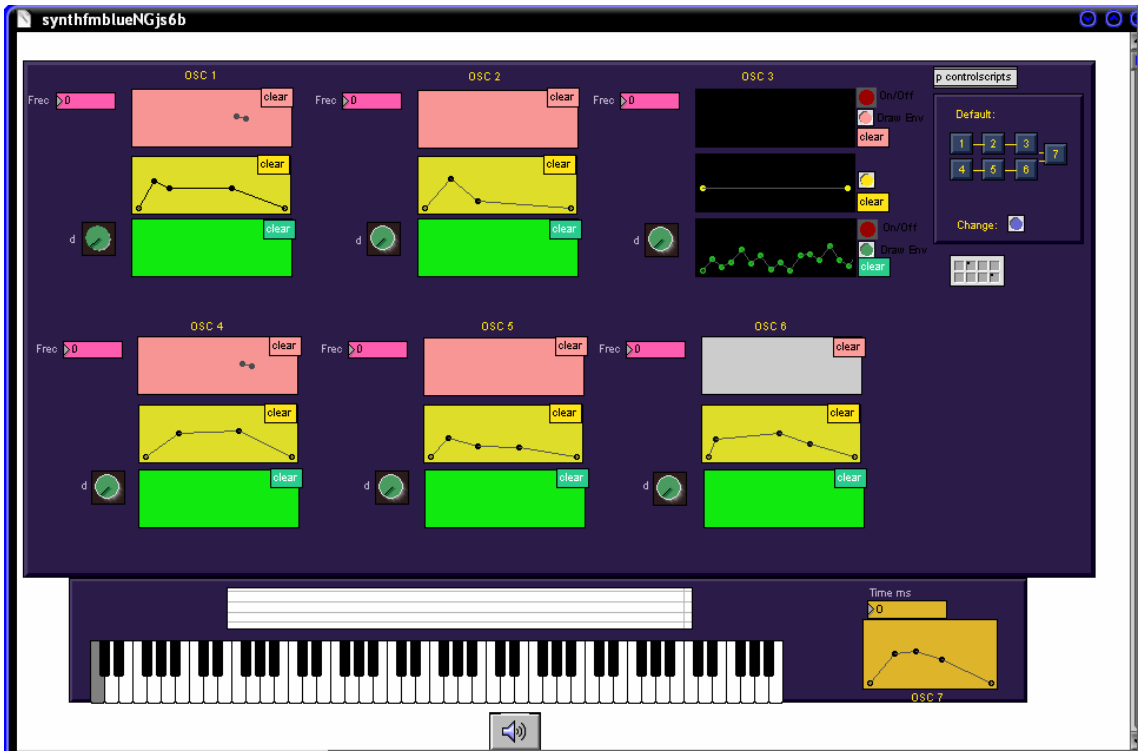
Nuevamente se ha intentado organizar la interfaz de usuario lo mas amigable posible para que éste pueda visualizar los parámetros y dar menor o mayor control al instrumento. Debido a la cantidad más o menos amplia de

componentes la vista aún así resulta un poco complicada, sin embargo se ha tratado al máximo de simplificar los controles para evitar confusiones.

Tenemos así una ventana general que simula un instrumento virtual, que posee la vista de los 6 osciladores modulantes con sus ventanas de envolventes. Tenemos un interruptor que permite encender y apagar cada envolvente y un botón que nos permite que se generen aleatoriamente éstas en caso de que así lo requiramos.

El interruptor mas relevante para nuestro caso es aquel en donde se acciona el mecanismo de generación de envolventes en los diferentes parámetros según se explicó en el esquema general.

Tenemos entonces una vista del siguiente tipo:



Capitulo 4 Resultados

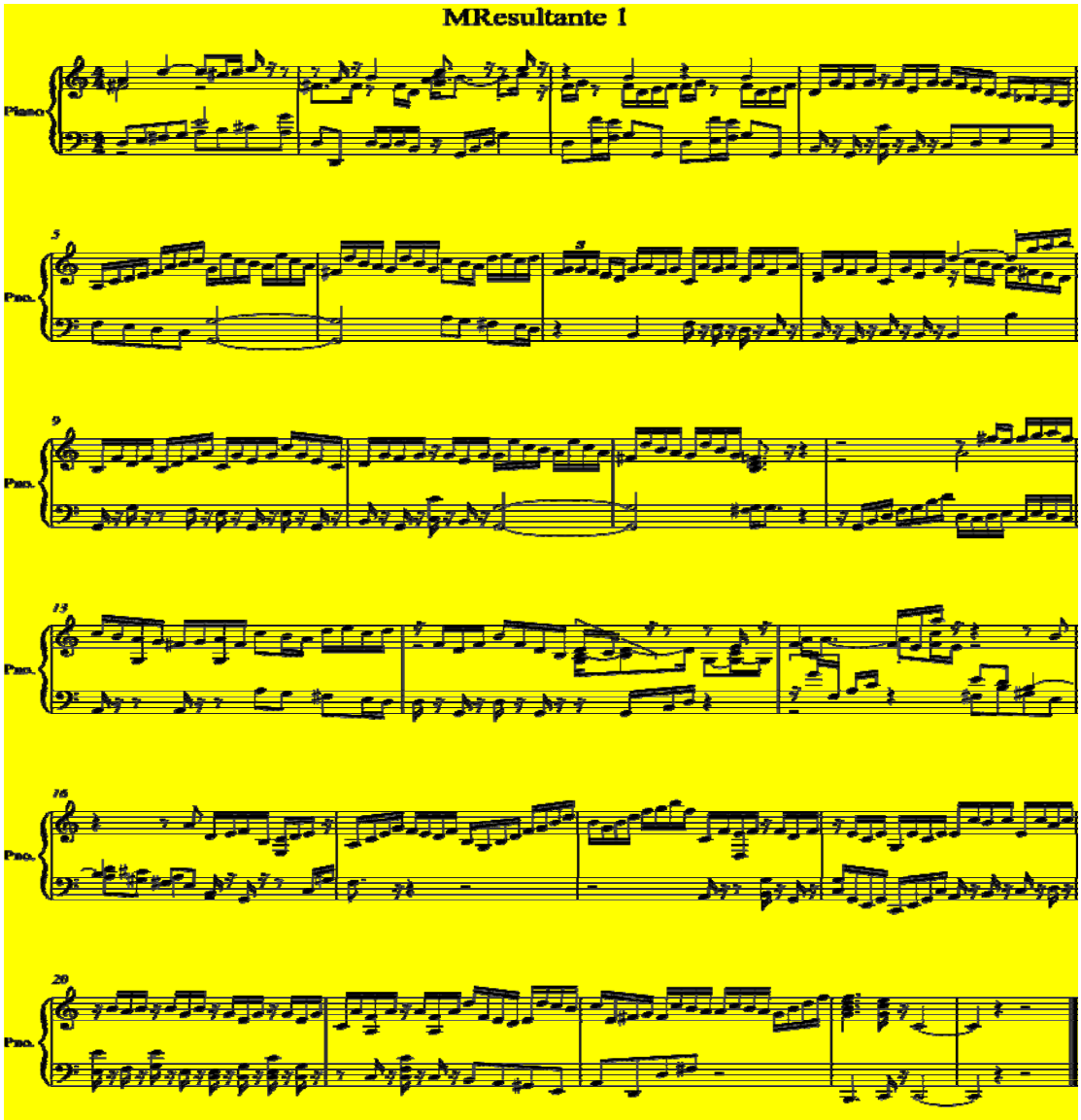
4.1 Evaluación del desempeño

El método de recombinación introducido por COPE aplicado a este algoritmo arroja resultados musicales muy elementales pero efectivos hasta los lineamientos que se plantearon.

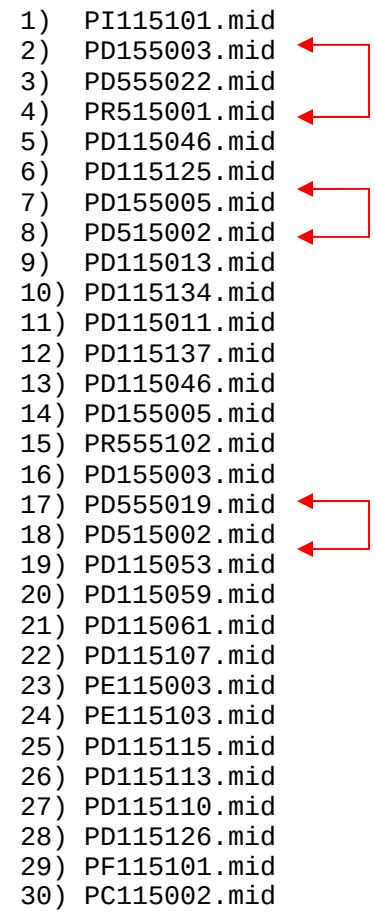
Para comenzar, se tomaran algunas melodías generadas por la máquina a manera de muestra y se procederá a analizar si los resultados arrojados fueron los esperados:

4.1.1. Resultados MIDI

Se generaron 10 melodías resultantes, con diferentes longitudes utilizando los bancos MIDI preestablecidos el resultado fue el siguiente:



Se observa la siguiente estructura combinatoria:



Observamos entonces la siguiente estructura:

En el en el primer ciclo se eligió un elemento inicial:

1) P**I**115101.mid

De acuerdo a la nomenclatura explicada en el capítulo **3.2.3.5.,** se observa entonces que el elemento pertenece e la voz principal (P), además es un elemento inicial (I) y su estado de transición es 11, esto es, sin alteraciones. El indicador (5) únicamente nos marca un tempo *andante*, y los tres últimos dígitos nos indican que se trata del banco número 101.

Los indicadores que serán más relevantes para el análisis que se realiza en éste momento será el indicador del tipo de elemento (Y) junto con los indicadores de transición (*ab)* de la nomenclatura: *XYabcddd.mid.*

Continuando con el análisis se observa entonces que:

2) P**D15**5003.mid
3) P**D55**5022.mid
4) P**R51**5001.mid

Del ciclo 2 al ciclo 4 observamos 2 elementos de desarrollo D y un elemento de cadencia R, tenemos además indicados los estados de transición 15>55>51 de acuerdo con las probabilidades de Markov establecidas, en el caso de ésta banco indican una modulación pasajera al quinto grado y un regreso rapido hacia el primero.

En los ciclos 5 y 6:

5) P**D11**5046.mid
6) P**D11**5125.mid

Se continúa con elementos de desarrollo sin alteraciones, esto es, en el primer grado, mientras que en los ciclos 7 y 8:

7) P**D15**5005.mid
8) P**D51**5002.mid

Vuelve a presentarse un cambio de estado muy breve.

De los ciclos 9 al 13 se continúa en estados de desarrollo sin alteraciones (*11*):

9) P**D11**5013.mid
10) P**D11**5134.mid
11) P**D11**5011.mid
12) P**D11**5137.mid
13) P**D11**5046.mid

En los ciclos 14 al 18 se observa nuevamente una serie de cambios de estado, primeramente de *1-5* en el ciclo 14 y luego del 15 al 16 hay un regreso al estado 1 sin modulación, esto obedeciendo la propiedad en que el estado R dado que es un estado de reposo puede regresar al estado 5 o al estado 1 sin restricción. Sin embargo del ciclo 16 al ciclo 18 volvemos observar un cambio de 1 a 5 y de regreso a 1 (*1>5>1*):

14) P**D15**5005.mid
15) P**R55**5102.mid
16) P**D15**5003.mid
17) P**D55**5019.mid
18) P**D51**5002.mid

Del ciclo 19 al 28 tenemos estabilidad en el estado *1-1*:

19) P**D11**5053.mid
20) P**D11**5059.mid
21) P**D11**5061.mid
22) P**D11**5107.mid
23) P**E11**5003.mid
24) P**E11**5103.mid
25) P**D11**5115.mid
26) P**D11**5113.mid

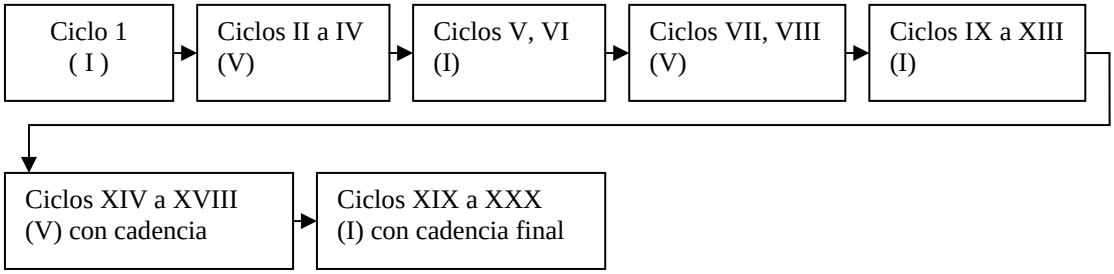
27) P**D11**5110.mid
28) P**D11**5126.mid

En el ciclo 29, que es el penúltimo, se tiene un elemento final que prepara la secuencia de elementos para su elemento conclusivo en el ciclo 30, que siempre deberá concluir en estado *1-1.*

29) P**F**115101.mid
30) P**C**115002.mid

Se ha podido observar entonces algunos cambios de estado de *1-5* que regresan a 1 obedeciendo las probabilidades establecidas por defecto que son de un 20% de permanecer en el estado de transición y 80 de regresar al estado inicial.

Haciendo un esquema formal incluyendo los cambios de estado tenemos una pieza con la siguiente estructura (medida por ciclos):



Para el resto de las *Melodías Resultantes* tnemos las siguientes transiciones (solo se han marcado los cambios al V, los demás permanecen en I):

MR2	MR3	MR4	MR5
PI115101	PI115105	PI115105	PI115103
PD115048	PD115011	PD115123	PD115014
PD115104	PD115008	PD115121	PD115042
PD115002	PD115039	P**D15**5110	PD115121
PD115103	PD115060	P**D51**5006	PF115104
PD115137	PD115031	PD115125	PC115001
PD115104	PE115006	PD115111	
PD**15**5006	PD**15**5110	PF115004	
PD**55**5112	PD**51**5128	PC115003	
PD**55**5006	PD115037		
PD**51**5002	PD115134		
PD115139	PD115045		
PD115032	PD115106		
PF115003	PD115125		
PC115002	PE115004		
	PD**15**5109		
	PD**55**5113		
	PD**55**5005		
	P**F51**5001		
	PC115003		

MR6	MR7	MR8	MR9
PI115007	PI115006	PI115105	PI115103
PD115039	PD115015	PE115004	PD115125
PD115066	PD115115	PD115122	PE115005
PD**15**5108	PD115126	PD**15**5108	PD115107
PD**55**5007	PR115002	PD**55**5009	PD115048
PD**55**5110	PD**15**5006	PD**51**5006	PD115044
PR**55**5116	PD**51**5049	PD115113	PD115063
PR**55**5103	PD115126	PE115003	PD115060
PD**55**5108	PD115024	PD115128	PD115126
PR**55**5116	PE115103	PD115063	PD115061
PD**15**5114	PD115010	PD115112	PD**15**5108
PD115126	PD115017	PD115023	PD**55**5110
PD115015	PD115128	PD115102	PD**51**5002
PD115038	PD115126	PD115045	PE115003
PE115103	PD115101	PD**15**5006	PD115101
PD115060	PD115101	PD**55**5110	PD115045
PD115011	PD115017	PD**15**5114	PD115005
PD115040	PD**15**5006	PD115128	PE115003
PD115104	PD**51**5001	PD115048	PD115018
PD115004	PD115031	PD115116	PD115049
PD115007	PD115011	PD115123	PD115128
PD115050	PE115007	PD115033	PD115029
PD115004	PD115046	PR115002	PD115015
PE115004	PD115105	PD115040	PD115054
PD115017	PD115130	PD115005	PD115042
PD**15**5110	PD115057	PD115139	PD115012
PF**51**5001	PD115123	PD115126	PR115002
PC115002	PD115004	PD**15**5108	PD**15**5006
	PF115104	PF**51**5001	PD**51**5128
	PC115002	PC115002	PF115002
			PC115003

Se observa entonces las tendencias probabilísticas en los estados de transición obedecen a los parámetros que establecimos, el sistema entonces presenta un funcionamiento correcto de acuerdo a lo esperado

4.1.2 Limitaciones

Si bien un sistema basado en la recombinación de elementos musicales ya existentes ofrece varias facilidades al momento de su implementación posee siempre las mismas debilidades, la más notoria por supuesto es que su efectividad y su aceptación por parte del usuario descansan sobre la selección de un banco de sonidos MIDI cuidadosamente elaborado.

Tenemos entonces que el resultado sonoro depende de un trabajo de selección o elaboración previo que descansa enteramente en el individuo que lo realiza y por supuesto, esto abre la puerta a discusiones filosóficas sobre en

qué momento realmente se realiza la composición de una pieza, si en el momento de poner en funcionamiento el sistema o a la hora de elaborar los bancos.

Sinembargo, ésta limitante no es exclusiva de el sistema aquí elaborado, de hecho, éste punto constituye en realidad la coyuntura más frágil de cualquier sistema de inteligencia artificial aplicado a la música, no importando lo complejo y eficiente que éste sea en su programación algorítmica.

4.2. Resultados Sintetizador FM

En cuanto al sistema aplicado al sintetizador FM se generaron también diferentes combinaciones de envolventes en frecuencia, índice de modulación y amplitud (*f,d,A*) que generaron diferentes espectros cambiantes en el tiempo.

Para las siguientes combinaciones de prueba se dejaron fijos los parámetros de número e interconexión de osciladores con fin de clarificar la comparación de los resultados:

Prueba 1:
Osciladores: 3 variables 1 fijo controlado por el teclado
Numeración: 1, 2,3 y 7 donde 7 es el oscilador principal
Interconexión: 1>2>3>7

Combinación Resultante 1:

Oscilador:	Combinación *f, d, A*
1	f: (fija, 300hz), d: (aleatoria), A: (curva senoidal)
2	f: (fija 150hz), d: (logarítmica), A: (lineal)
3	f: (aleatoria), d: (fija 40), A: (fija 0.3)

Combinación Resultante 2:

1	f: (aleatoria), d: (fija26), A: (fija 0.3)
2	f: (fija 843hz), d: (fija 76), A: (lineal)
3	f: (exponencial), d: (fija 32), A: (fija: 0.8)

Combinación Resultante 3:

1	f: (fija 0hz), d: (inversa), A: (lineal)
2	f: (aleatoria) d: (fija 600), A: (fija 0.7)
3	f: (senoidal), d: (fija 34), A: (0.3)

Para medir los resultados en relación a los espectros sonoros que se generan al utilizar las combinaciones obtenidas se utilizó una medición gráfica con tres variables, tiempo, frecuencia y amplitud conocida como sonograma.

4.2.1 Características del dispositivo de graficación del sonograma

Para la graficación del sonograma se realizó una pequeña programación en MAX/MSP y se conectó con la salida de señal general del sintetizador:

Las características del dispositivo graficador de sonograma es la siguiente:

Eje X: *tiempo*
Rango de tiempo de la ventana: 15 segundos

Eje Y: *frecuencia*
Rango de frecuencias: de 0 a 22050 hz. Tipo de vista: logarítmica

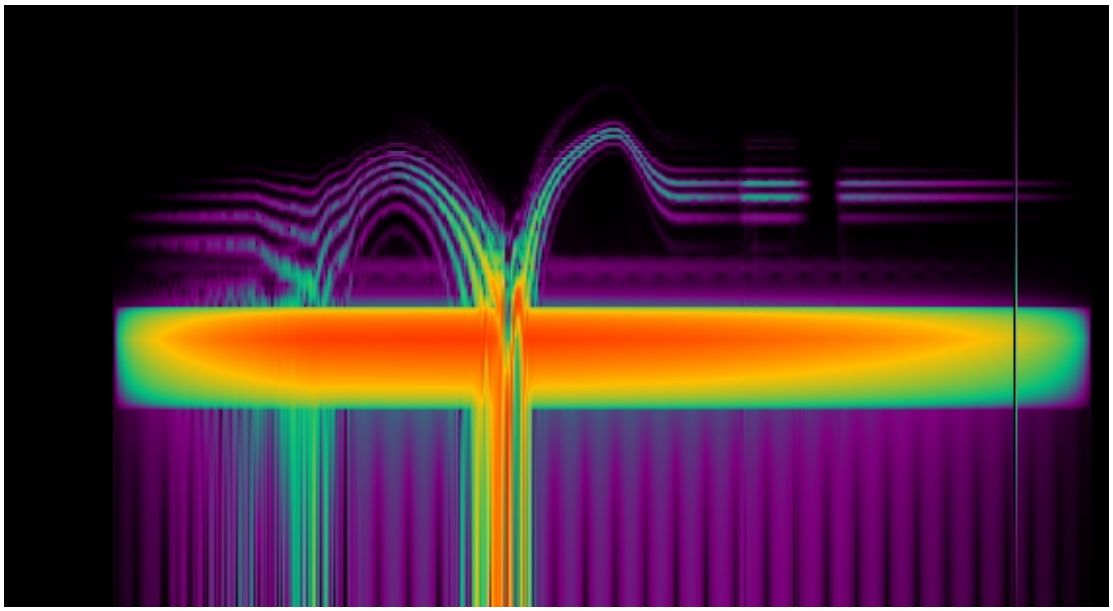
Colores e intensidad de color: *Amplitud*

Escala comparativa (de 1 a 6):
Rojo 6, naranja 5, amarillo 4, verde 3, azul 2, violeta 1

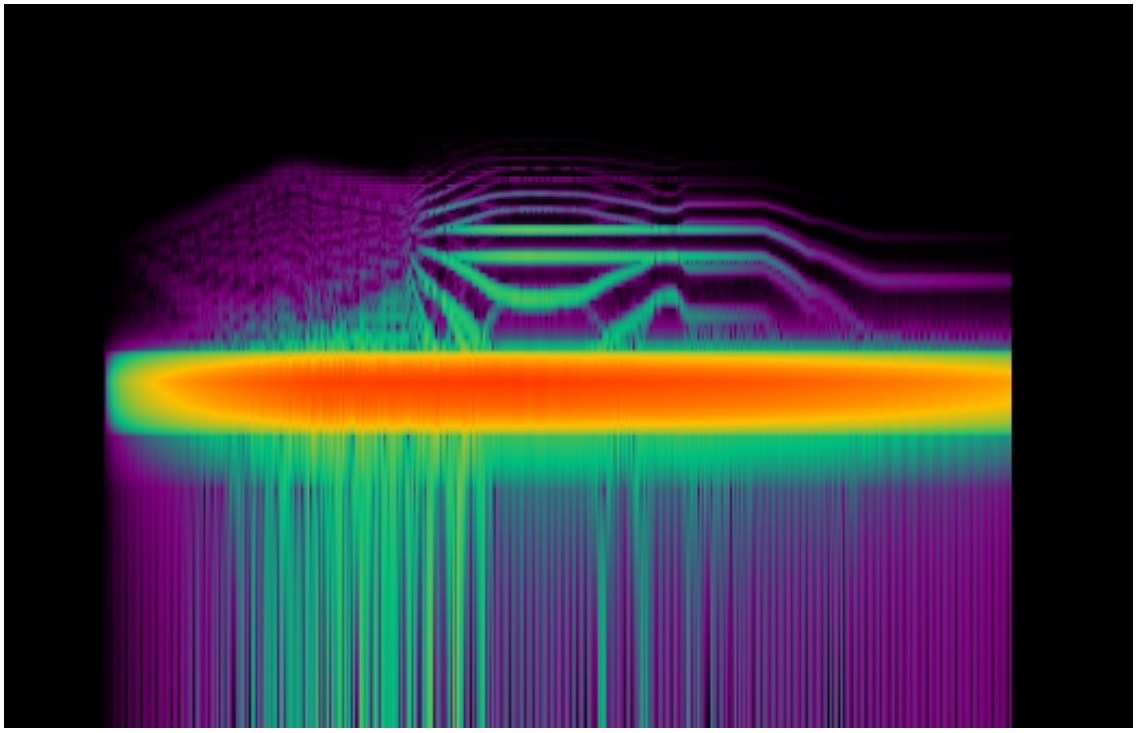
Frecuencia de muestreo: 44.1Khz
Rango de amplitud –INF a 1.578207 dB
Tipo de ventana de muestreo: Blackman-Harris

4.2.2. Sonogramas

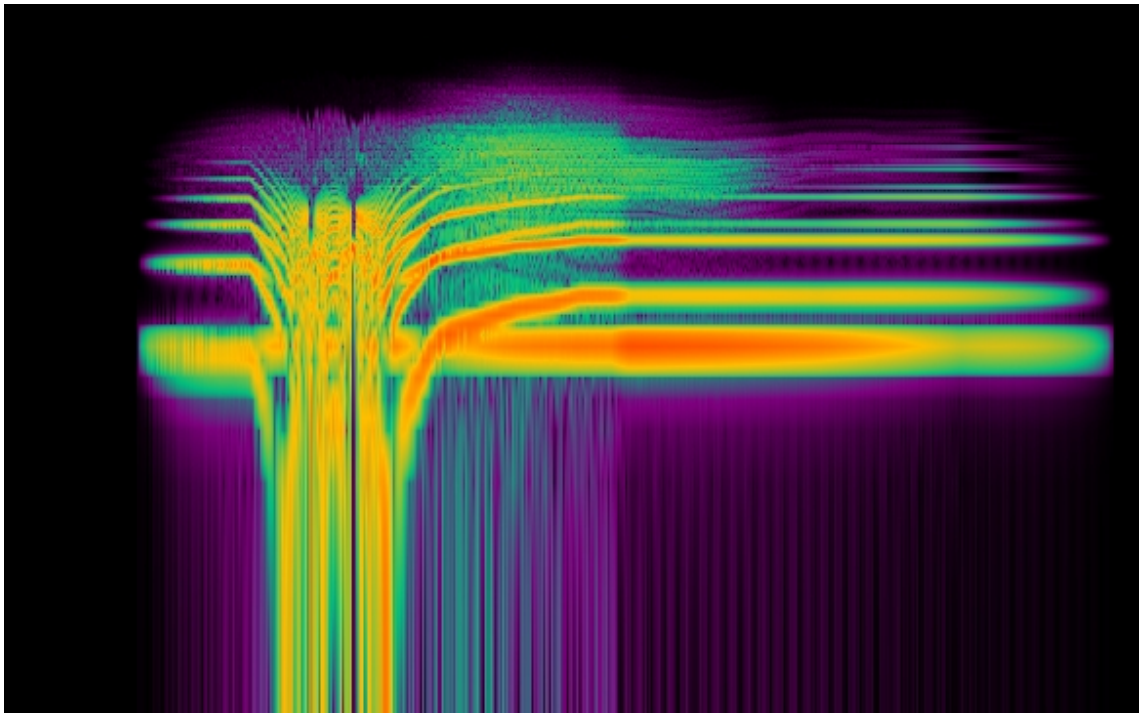
Los sonogramas respectivos de las combinaciones obtenidas son:
Combinación Resultante 1:



Combinación Resultante 2:

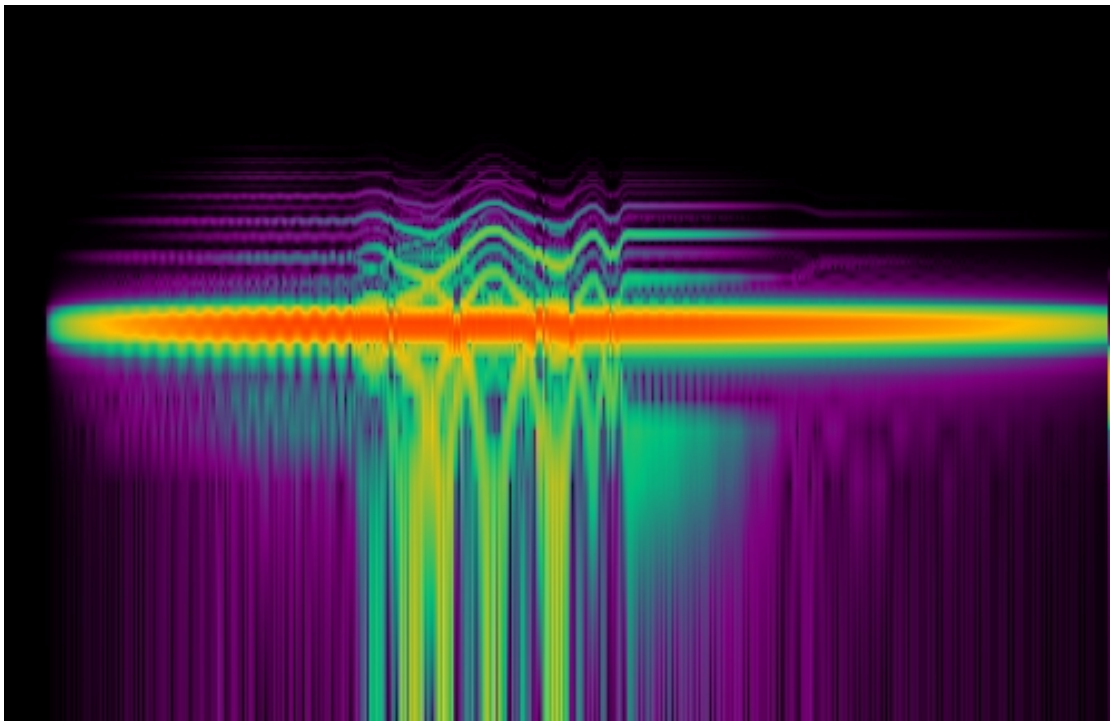


Combinación Resultante 3:

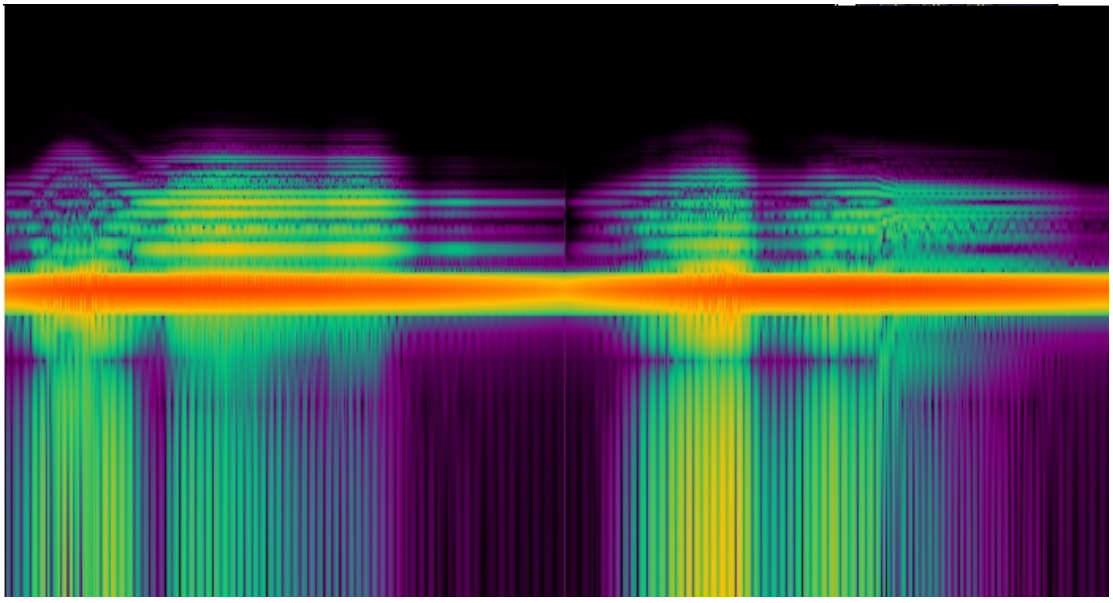


Para algunas otras combinaciones generadas con las mismas condiciones se generaron los siguientes espectros:

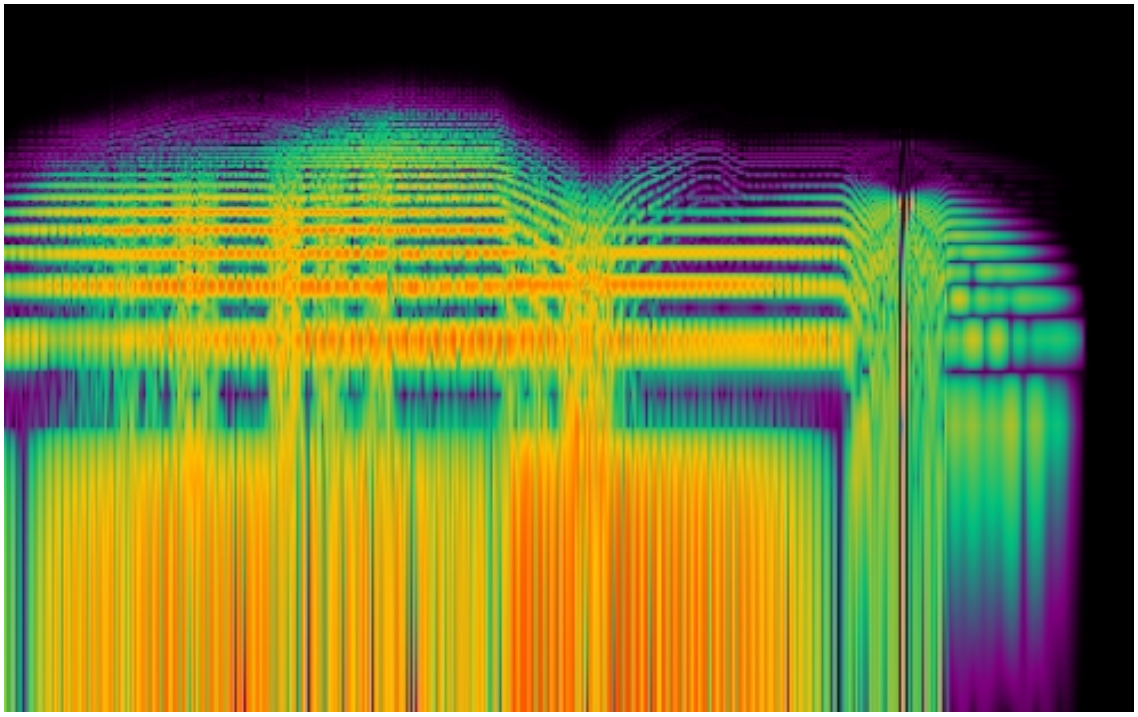
Combinación Resultante 4:



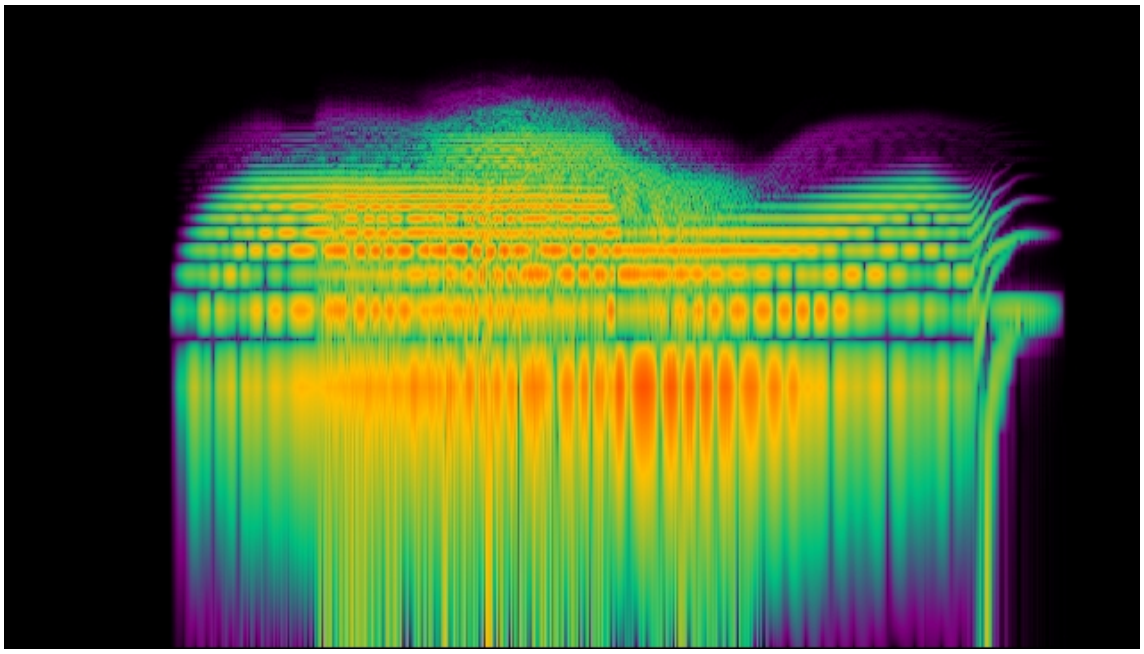
Combinación Resultante 5:



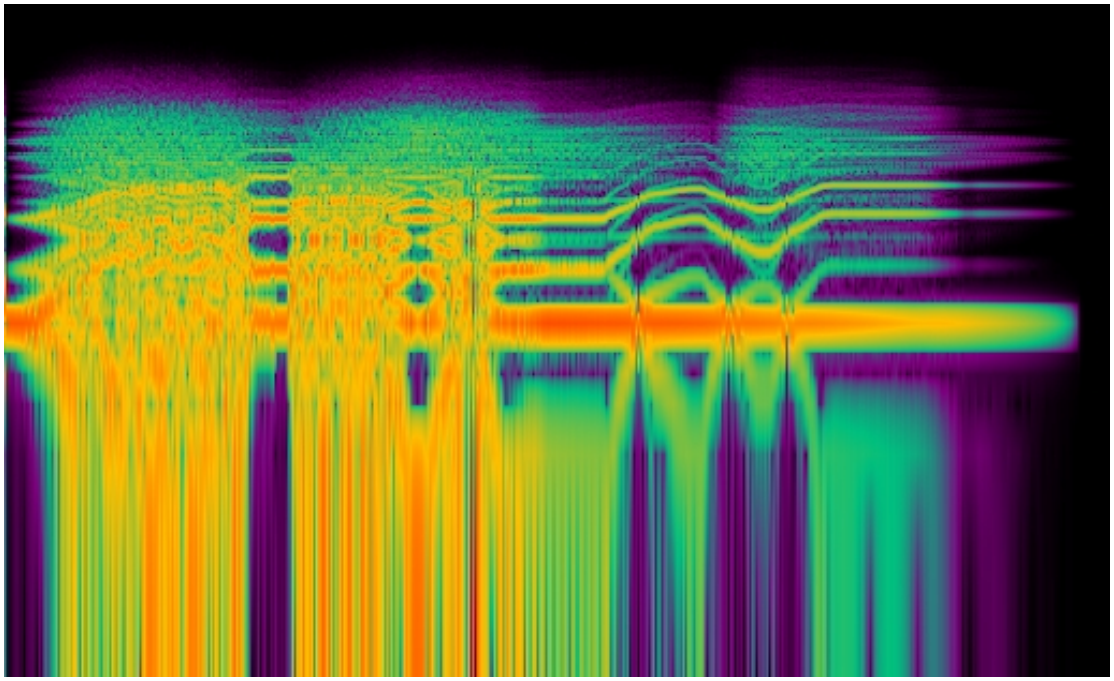
Combinación Resultante 6:



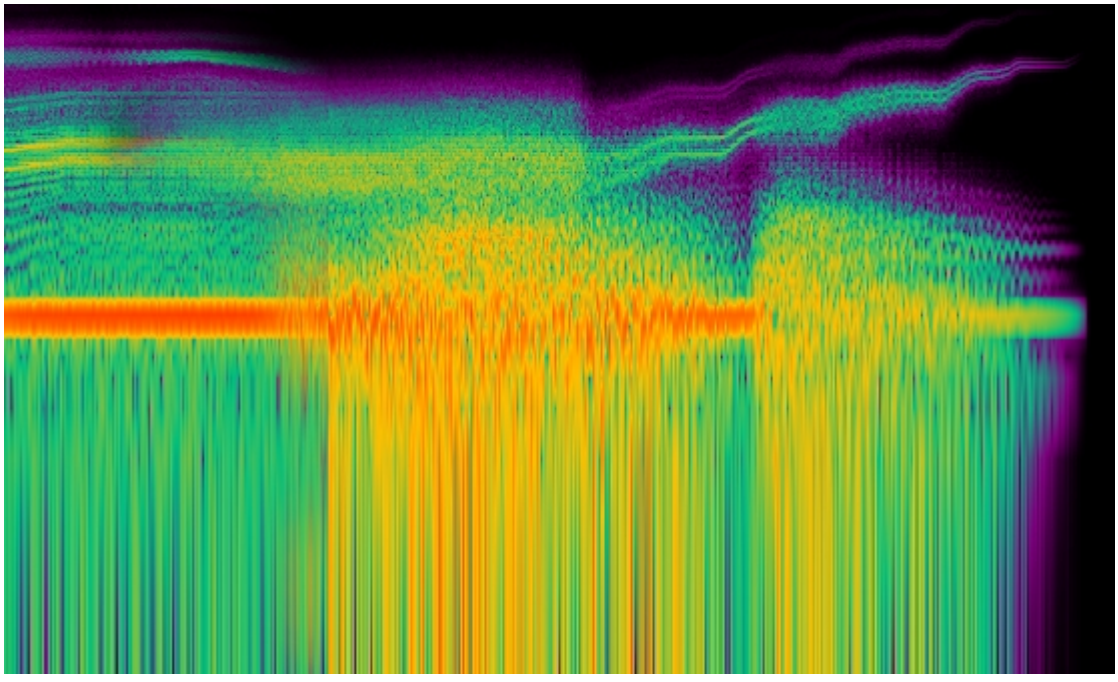
Combinación Resultante 7:



Combinación Resultante 8:



Combinación Resultante 9:



Se puede observar un espectro de frecuencias más dinámico en el tiempo que además no requiere de una programación muy elaborada por parte del usuario, de esta manera se obtienen espectros interesantes con la facilidad de oprimir un par de botones y seleccionar un par de parámetros, esto resulta altamente atractivo para usuarios que no desean invertir gran cantidad de tiempo para producir sonidos útiles.

Capítulo 5 Conclusiones

5.1 Conclusiones varias

Los autómatas gramaticales y las máquinas de estado finito constituyen un sistema de inteligencia artificial básico pero eficiente. Resulta relativamente sencillo generar un comportamiento mediante reglas simples; además no requieren de grandes recursos informáticos para su programación.

Los resultados que se obtuvieron coinciden con lo esperado inicialmente, por lo que, en cuanto al desempeño del sistema, podemos decir que ha sido satisfactorio. Se han generado pequeñas piezas musicales utilizando la técnica de recombinación establecida por COPE (1996) que mantienen su coherencia tanto sintáctica como formal.

La implementación de este sistema nos muestra entonces las ventajas que tienen los autómatas gramaticales y las máquinas de estado finito en la creación de sistemas de inteligencia artificial aplicados a la música. Aunque su naturaleza es sencilla, esto es, las tareas que desarrollan son básicas, su capacidad para generar comportamientos específicos se adecua muy bien al proceso natural de la creación musical.

Por supuesto que los resultados pueden parecerle a algunos como sonoramente simples, sinembargo, resultan sorprendentes cuando advertimos que éstos se obtuvieron de algoritmos muy compactos. Esto contrasta con otros sistemas como las redes neuronales que si bien son un sistema de inteligencia artificial mucho mas complejo, requiere de enormes recursos informáticos y una programación complicada extensa.

Esta es una de las razones por las que quizás los autómatas gramaticales y las máquinas del estado finito se mantienen vigentes como sistemas de inteligencia artificial muy utilizados.

No solo eso, sino que apoyados en las nuevas teorías planteadas por expertos en Inteligencia Artificial como Marvin Minsky (1981), Jennings, (1998) y Bussman & Müller H. J. (1993), en relación con los sistemas basados en agentes, los autómatas vuelven a ser tomados en cuenta en la creación de Inteligencia Artificial más compleja que puede competir en efectividad con otros sistemas más elaborados.

En relación al sintetizador FM, su tratamiento fue aún más sencillo, y sinembargo los resultados sonoros son mucho más notorios al oído. Esto, debido a que los espectros generados tienen comportamientos más complejos que los que llegan a presentarse en otros sintetizadores FM.

Tenemos entonces que, aunque en este punto no se trataron comportamientos más complejos, las posibilidades para su utilización por parte de compositores o intérpretes son más inmediatas. De esta manera, éste ha sido un paso importante en la búsqueda por incorporar sistemas de inteligencia

artificial al procesamiento de señales o la síntesis de sonido y abre la puerta para desarrollar investigaciones posteriores.

5.2 El paso siguiente (Investigaciones futuras)

El sistema implementado cumple con los objetivos fijados desde el inicio, sin embargo, existen algunas cuestiones que, aunque se dejaron a un lado inicialmente, será muy interesante incluir como complemento o continuación a esta investigación.

El primer punto por supuesto, deberá tratar la tarea de seleccionar de manera automática los bancos MIDI a partir de un archivo completo. Es de hacer notar que éste puede llegar a ser un proceso tan complejo como el generador mismo y deberá plantearse cuidadosamente para que pueda ser útil.

Un paso un poco más ambicioso será el crear un sistema capaz de analizar un archivo MIDI existente y poder, de alguna manera, generar datos estadísticos de las regiones armónicas, armonías utilizadas y patrones rítmicos para su posterior incorporación a otro sistema que intente reconstruir una pieza completa con los mismos patrones estadísticos dando como resultado una imitación musical.

Es de hacer notar que en los puntos anteriores surge ya una cuestión interesante, y ésta se refiere a que si se crean sistemas que realicen tareas específicas, deberá entonces de crearse un método para poder coordinar todo los sistemas en uno solo. Y aquí la teoría de agentes encaja casi a la perfección.

Podemos decir entonces que un paso casi obligado en la búsqueda por un sistema más complejo e independiente a partir del trabajo ya realizado, será el crear un sistema basado en agentes informáticos completo con todas sus características ya descritas.

5.3 La cuestión de la traducción de algoritmos en música
El problema de todos los algoritmos musicales

El problema mas importante que presentan todos los sistemas enfocados a la creación musical automática, no importando lo complejo o sencillos que estos sean es el mismo, y constituye el obstáculo a salvar más importante de todos.

Y es que, aun cuando el sistema funcione con toda eficiencia, el resultado sonoro buscado siempre cae en el campo de lo subjetivo.

Tenemos entonces que aún cuando el sistema de Inteligencia artificial sea muy complejo o completo y trabaje con toda eficiencia, la decisión de qué

parámetros musicales serán los trabajados, recae completamente en el individuo que lo crea, y éste toma las decisiones basado en su propia concepción de lo que es la música y sus criterios personales, dando entrada a variables subjetivas que pueden ser polémicas.

Este será entonces el punto más delicado al momento de planear cualquier sistema de IA aplicado a la música y debe tomarse en cuenta desde el principio.

Los resultados pueden o no ser aceptables para nuestra percepción musical (que además varia enormemente dependiendo de la persona, y su concepto de lo que para ella) nos enfrentamos entonces a paradigmas subjetivos altamente cuestionables sobre lo que es la musicalidad o creatividad, asuntos que de inicio presentan muchos problemas en su definición, pero que no deben de interferir, aunque parezca contradictorio, en la elaboración de los sistemas.

No obstante no nos cabe duda que conforme vayan habiendo avances en los métodos de IA y en áreas como la cognición musical, muy pronto nos estaremos encontrado con sistemas que hayan logrado salvar este obstáculo y arrojen resultados mas aceptables para ese concepto abstracto y personal de lo que conocemos nosotros como música.

Anexo 1 Códigos fuente

1.- Script Principal:

```
// Aqui se declaran las variables y valores
outlets=4;
setoutletassist (0,"send sequence name"); setoutletassist (1,"bang
before last sequence");
setoutletassist (2,"stop others");

var xxx;
var last_tone=1;
var nvueltas = 30;
var contador = 1;
var pth = String("");

var pifilenames = new Array ();
var pd1filenames = new Array ();
var pd5filenames = new Array ();
var pefilenames = new Array ();
var pffilenames = new Array ();
var pcfilenames = new Array();

// Aqui se define la funcion random
   function random(maxnum) {
        r=Math.floor(Math.random()*maxnum)+1;
        if (r>maxnum) r=maxnum;
        return r;
   }

// Aqui se definen los valores defalut
     function elementos (elem) {
         nvueltas = elem;
     }

// Aqui se define la funcion de reset de los valores
   function reset () {
         nvueltas = 30;
         contador = 1;
         outlet (1,0);
   }

// Aqui se define el folder de los archivos midi

    function path (p){
      pth = p;
    }

// Aqui se leen lor Arrays del archivo de texto
    function readlines(s) {
      var f = new File(s);
      var i,a,c;
      if (f.isopen) {
            ci=0; cd1=0; cd5=0; ce=0; cf=0; cc=0;
            while (a=f.readline()) {
                  if (a.substr(0,1)=='[') {
                        b=a;
                  } else {
                        if (b=='[iniciales]') {
                              pifilenames[ci] = a;
```

```
                        ci++;
                    } else if (b=='[desarrollo1]') {
                        pd1filenames[cd1] = a;
                        cd1++;
                    } else if (b=='[desarrollo5]') {
                        pd5filenames[cd5] = a;
                        cd5++;
                    } else if (b=='[enlace]') {
                        pefilenames[ce] = a;
                        ce++;
                    } else if (b=='[final]') {
                        pffilenames[cf] = a;
                        cf++;
                    } else {
                        pcfilenames[cc] = a;
                        cc++;
                    }
                }
            }
            f.close();
      } else {
            post("No se pudo abrir el archivo: " + s + "\n");
            outlet(3,0);
      }

    }

// Aqui se aciva el porceso
   function bang () {
      post ("contador p: ", contador);
      if (contador==1) {
            outlet (0,pth+pifilenames[random(pifilenames.length - 1)]);
      } else if (contador==nvueltas) {
            outlet (0,pth+pcfilenames[random(pcfilenames.length - 1)]);
            outlet (2,"bang");
      } else if (contador==nvueltas-1) {
            outlet (0,pth+pffilenames[random(pffilenames.length - 1)]);
      } else {
            if (random(5)==1)  // el diez es para darle 20%de
probabilidad a que ocurra un enlace
                  outlet (0,pth+pefilenames[random(pefilenames.length -
1)]);
            else {
                  if (last_tone==5) {
                        // 20% de que regrese a uno.
                        if (random(5)==1)   {
                              xxx=random(pd5filenames.length - 1);
                              last_tone=pd5filenames[xxx].substr(3,1);
                              outlet (0,pth+pd5filenames[xxx]);
                        } else {
                              xxx=random(pd1filenames.length - 1);
                              last_tone=pd1filenames[xxx].substr(3,1);
                              outlet (0,pth+pd1filenames[xxx]);
                        }
                  } else {
                        xxx=random(pd1filenames.length - 1);
                        last_tone=pd1filenames[xxx].substr(3,1);
                        outlet (0,pth+pd1filenames[xxx]);
                  }
            }
      }
```

```
      if (contador>=nvueltas){
           outlet (1,1);
           contador=1;
       } else {
           contador = contador + 1;
          }

   }
```

2.- Script Generador de poema

```
inlets = 2;
outlets = 2;

var frases=''; var poema='';
random_count=0; nsentences=0;
nwords1=0; nwords2=0; nwords3=0; nwords4=0; nwords5=0; nwords6=0;
nwords7=0; nwords8=0; nwords9=0;
list1=''; list2=''; list3=''; list4=''; list5=''; list6=''; list7='';
list8=''; list9='';
frases = "9, 2!"
+"\n2, 2 Y 2."
+"\n1s 4ad!"
+"\nEL 5o 1 3a 6o AL 1."
    +"\nEL 5o 1 3a 6o AL 1 5o."
    +"\nEL 2 ES UN 1 5o."
    +"\nEL 1 4a COMO UN 5o 1."
    +"\nLOS 1s 4an COMO 5os 1s."
    +"\nPOR QUÉ 4a EL 1?"
    +"\n4a 6O COMO UN 5o 1."
    +"\nDONDE ESTÁ EL 5o 1?"
    +"\nTODOS LOS 1s 3an 1s 5os, 5os."
    +"\nNUNCA 3es UN 1."
    +"\nLA 5a 7 3a 6a A LA 7."
    +"\nLA 5a 7 3a 6adamente A SU 5a 7."
    +"\nLA 8 ES UNA 7 5a."
    +"\nLA 7 4a COMO UNA 5a 7."
    +"\nLAS 7s 4an COMO 5as 7s."
    +"\nPOR QUÉ 4a LA 7?"
    +"\n4a 6o COMO UNA 5a 7."
    +"\nDONDE ESTÁ LA 7 5a?"
    +"\nTODAS LAS 7s 3an 7s 5as, 5as."
    +"\nNUNCA 3es UNA 7."
    +"\nEL 5o 1 3a 6o A LA 7."
    +"\nEL 5o 1 3a 6o UNA 7 5a."
    +"\nEL 2 ES UNA 7 5a."
    +"\nEL 1 4a COMO UNA 5a 7."
    +"\nLOS 1s 4an COMO 5as 7s."
    +"\n4a 6o COMO UNA 5a 7."
    +"\nTODOS LOS 1s 3an 7s 5as, 5as."
    +"\nLA 5a 7 3a 6a AL 1."
    +"\nLA 5a 7 3a AL 1 5o."
    +"\nLA 8 ES UN 1 5o."
    +"\nLA 7 4a COMO UN 5o 1."
    +"\nLAS 7s 4an COMO 5os 1s."
    +"\n4a 6o COMO UN 5o 1."
    +"\nTODAS LAS 7s 3an 1s 5os, 5os."
    ;
```

```
    list1 =
"ESPACIO\nASTRONAUTA\nCAPITÁN\nSOLDADO\nEXTRATERRESTRE\nMARCIANO\nMONS
TRUO\nLÁSER\nPIRATA\nROBOT\nESPACIOPUERTO\nSOL";
    list2 = "CORAJE\nVALOR\nRETO\nAMOR";
    list3 = "COMAND\nMIR\nAM\nDESE\nCONFRONT";
    list4 = "VIAJ\nEXPLOT\nDESINTEGR\nMAT";
    list5 =
"PEQUEÑ\nVIEJ\nFRÍ\nCÁLID\nOSCUR\nLIMPI\nRUD\nÁVID\nMUERT";
    list6 = "RÁPID\nCALM\nQUIET\nTRANQUIL\nDESESPERAD";
    list7 =
"NAVE\nLUZ\nNEBULOSA\nFLOTA\nNOVA\nCHICA\nESTRELLA\nCONSTELACIÓN\nLUNA
";
    list8 = "AVENTURA\nDESOLACIÓN\nMUERTE\nVIDA\nFE";
    list9 = "OH\nOOH\nAH\nSEÑOR\nDIOS\nWOW\nDIABLOS";

function count_all_lines() {
 nwords1 = count_lines(list1);
 nwords2 = count_lines(list2);
 nwords3 = count_lines(list3);
 nwords4 = count_lines(list4);
 nwords5 = count_lines(list5);
 nwords6 = count_lines(list6);
 nwords7 = count_lines(list7);
 nwords8 = count_lines(list8);
 nwords9 = count_lines(list9);
 nsentences = count_lines(frases);
}

function random(maxnum) {
 r=Math.floor(Math.random()*maxnum)+1;
 if (r>maxnum) r=maxnum;
 return r;
}

function count_lines(str) {
 len=str.length;
 nword=1;
 for (i=0; i<len; i++) {
  if (str.charAt(i)=="\n") {
   nword++;
  }
 }
 if (str.charAt(len-1)=="\n") nword--;
 return nword;
}

function get_line(str, lnum) {
 len=str.length;
 iline=1; ichar=0; jchar=-1;
 for (i=0; i<len; i++) {
  if (str.charAt(i)=="\n") {
   iline++;
   if (iline==lnum) {
    ichar=i+1;
   } else if (iline==(lnum + 1)) {
    jchar=i-1;
    if (str.charAt(jchar)=="\r") {
     jchar--;
    }
    break;
   }
```

```
  }
 }
 if (jchar<0) jchar=len-1;
 s="";
 for (i=ichar; i<=jchar; i++) {
  s = s + str.charAt(i);
 }
 return s;
}

function make_poem() {
 poema="";
 count_all_lines();
 nlines=random(3)+2;
 for (ilin=1; ilin<=nlines;ilin++) {
  make_poem_line()
 }
 while (poema.indexOf("ÓNs")>0) poema=poema.replace("ÓNs", "ONes");
 while (poema.indexOf("ÁNs")>0) poema=poema.replace("ÁNs", "ANes");
 while (poema.indexOf("Ls")>0) poema=poema.replace("Ls", "Les");
 while (poema.indexOf("Ns")>0) poema=poema.replace("Ns", "Nes");
 while (poema.indexOf("Zs")>0) poema=poema.replace("Zs", "Ces");
 return "<i>"+poema.toLowerCase()+"</i>";
}

function make_poem2() {
 poema="";
 count_all_lines();
 nlines=random(3)+2;
 for (ilin=1; ilin<=nlines;ilin++) {
  make_poem_line2()
 }
 while (poema.indexOf("ÓNs")>0) poema=poema.replace("ÓNs", "ONes");
 while (poema.indexOf("ÁNs")>0) poema=poema.replace("ÁNs", "ANes");
 while (poema.indexOf("Ls")>0) poema=poema.replace("Ls", "Les");
 while (poema.indexOf("Ns")>0) poema=poema.replace("Ns", "Nes");
 while (poema.indexOf("Zs")>0) poema=poema.replace("Zs", "Ces");
 return poema.toLowerCase();
}

function make_poem_line() {
 pattern=get_line(frases, random(nsentences));
 lenpat=pattern.length;
 for (ichr=0; ichr<lenpat; ichr++) {
  chr = pattern.charAt(ichr);
  if ((chr>='1') && (chr<='9')) {
   if (chr=='1') {
    wrd=get_line(list1, random(nwords1));
   } else if (chr=='2') {
    wrd=get_line(list2, random(nwords2));
   } else if (chr == '3') {
    wrd=get_line(list3, random(nwords3));
   } else if (chr == '4') {
    wrd=get_line(list4, random(nwords4));
   } else if (chr == '5') {
    wrd=get_line(list5, random(nwords5));
   } else if (chr == '6') {
    wrd=get_line(list6, random(nwords6));
   } else if (chr == '7') {
    wrd=get_line(list7, random(nwords7));
   } else if (chr == '8') {
```

```
    wrd=get_line(list8, random(nwords8));
   } else if (chr == '9') {
    wrd=get_line(list9, random(nwords9));
   } else {
    wrd='';
   }
   poema=poema+wrd;
  } else {
   poema=poema+chr;
  }
 }
 poema=poema+"<BR>\n";
}

function make_poem_line2() {
 pattern=get_line(frases, random(nsentences));
 lenpat=pattern.length;
 for (ichr=0; ichr<lenpat; ichr++) {
  chr = pattern.charAt(ichr);
  if ((chr>='1') && (chr<='9')) {
   if (chr=='1') {
    wrd=get_line(list1, random(nwords1));
   } else if (chr=='2') {
    wrd=get_line(list2, random(nwords2));
   } else if (chr == '3') {
    wrd=get_line(list3, random(nwords3));
   } else if (chr == '4') {
    wrd=get_line(list4, random(nwords4));
   } else if (chr == '5') {
    wrd=get_line(list5, random(nwords5));
   } else if (chr == '6') {
    wrd=get_line(list6, random(nwords6));
   } else if (chr == '7') {
    wrd=get_line(list7, random(nwords7));
   } else if (chr == '8') {
    wrd=get_line(list8, random(nwords8));
   } else if (chr == '9') {
    wrd=get_line(list9, random(nwords9));
   } else {
    wrd='';
   }
   poema=poema+wrd;
  } else {
   poema=poema+chr;
  }
 }
 poema=poema+"<br/>\n";
}
```

Anexo 2 Glosario

Aiff,Wav	Formatos de archivo de audio digital comúnmente utilizados. Son los estándares para las plataformas Mac y PC
Algoritmo	Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema. De un modo más formal, un algoritmo es una secuencia finita de operaciones realizables, no ambiguas, cuya ejecución da una solución de un problema en un tiempo finito.
Arreglo o array	Arreglo. Es una variable que contiene una serie de datos numericos o alfanumericos
AU (Audio Units)	Formato creado por Apple para la ejecución de plugins.
Determinismo	El determinismo afirma que todo acontecimiento están causalmente determinadas por la irrompible cadena causa-consecuencia
DX (DirectX)	Formato estándar de windows para la ejecución de aplicaciones multimedia y plugins.
Efecto de inserción	Efecto que se introduce en la ruta de la señal de un canal de audio
Estocastico	Se denomina estocástico a aquel sistema que funciona, sobre todo, por el azar. Las leyes de causa-efecto no explican cómo actúa el sistema (y de modo reducido el fenómeno) de manera determinista, sino en función de probabilidades.
Host	Anfitrion. Es una aplicacion desde la cual pueden ejecutarse otras aplicaciones
Lenguajes de bajo nivel	Lenguajes de programación gran complejidad que se utiliza para crear aplicaciones básicas o muy avanzadas
Loop	Archivo de audio que se repite constantemente, muy utilizado para secuenciar patrones rítmicos
Max/Msp	Software de síntesis y procesamiento de audio en tiempo real desarrollado por el IRCAM y comercializado por Cycling'74. Que se ha vuelto el estándar utilizado por las la mayoría de las universidades y escuelas de música del mundo.
MIDI	Musical Instruments Digital Interface. Estándar de comunicación para instrumentos musicales u ordenadores introducido en los 80
modular	En un sistema modular es posible agregar por separado diferentes instancias o modulos para agrandarlo o aumentar su complejidad
objeto	En la programacion orientada a objetos es un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. Contiene siempre 3 partes importantes: RELACIONES, PROPIEDADES, METODOS.
Pluggo®.	Suite de plugins disenados y distribuidos por la compania de software Cycling'74
Plugin	Aplicación que corre dentro de otra y que, en el caso de audio, puede ser un efecto o un instrumento virtual
Programacion orientada a objetos	La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.
Reaktor	Software de síntesis y procesamiento en tiempo real desarrollado por Native Instruments.
Regiones armonicas	Regiones de una pieza musical que poseen una tonalidad en particular y que pueden o no pertenecer al tono principal de la obra
Rewire	Tecnología desarrollada por Steinberg que permite correr un software como esclavo dentro de un anfitrión
Runtime Enviroment	Programa creado para correr una aplicación que no puede ejecutarse de manera independiente.
suite	Una suite de plugins en un conjunto de los mismos que se encuentran agrupados en relacion a su funcion. Por ejemplo, audio.
Vst (Virtual Studio Technology)	Formato creado por Steinberg y que ha sido aceptado por muchos fabricantes como el standard actual para plugins sean efectos o instrumentos virtuales
Wrapper	Plugin anfitrión compatible que corre una aplicación u otro plugin no compatible dentro de sí.

BIBLIOGRAFIA

BALABAN, K. and LASKE O. *Understanding Music with AI: Perspectives on Music Cognition.*AAAI Press (July 16, 1992)

Bond A.H., Gasser L. *Readings in Distributed Artificial Intelligence*. Morgan Kaufman Pub. (1988)

BOULANGER, Richard. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing,and Programming*. The MIT Press; Bk&CD-Rom edition (March 6, 2000).

BRADSHAW, J., Software Agents, AAAI Press/The MIT Press. (1996)

CHOMSKY, N. *Syntactic Stuctures.* The Hauge: Mouton and Co. (1957).

CHU-CARROLL C. y CARBERRY S. (1995) 'Conflict Detection and Resolution in Collaborative Planning' INTELLIGENT AGENTS II: Agent Theories, Architectures and Languages, pp 111-127, Wooldridge, Michael J, Mueller, P and Tambe, Milind, (Eds), Springer Verlag, Berlin 1995.

COOK, Perry R. *Real Sound Synthesis for Interactive Applications*. AK Peters, Ltd.; Book & CD edition (June, 2002).

COPE, David. *Experiments in Musical Intelligence*. A-R Editions; Bk&CD-Rom edition (July, 1996).

COPE, David. *The Algorithmic Composer*. A-R Editions; Bk&CD-Rom edition (June, 2000).

COPE, David. *Virtual Music : Computer Synthesis of Musical Style*. The MIT Press; Book & CD edition (March 1, 2004).

COPELAND, B. Jack . *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*, Oxford University Press, Oxford UK (2004)

DODGE, Charles. *Computer Music : Synthesis, Composition, and Performance*. Wadsworth Publishing; 2 edition (July 2, 1997).

FERBER, J. Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence, Addison-Wesley.1st ed. (1999)

HOPCROFT JOHN E.; ULLMAN JEFFREY D. *Introducción a la teoría de autómatas, lenguajes y computación*. CECSA. (2002).

J.E. HOPCROFT, J.D. ULLMAN, *Introducción a la teoría de Autómatas, Lenguajes y Computación*, 2 Edición. Ed. Addison - Wesley, USA 2002

JAKENDOFF, R. Semantics and cognition. Cambridge, MA: MIT Press. (1983).

JOHN VON NEUMAN. *Theory of Self-Reproducting Automata*. University of Illinois Press. Ubana and London (1966).

KELLY, DEAN *Teoría de Autómatas y Lenguajes Formales*, Ed. Prentice Hall. (1995)

LAURENE V. Fausett. *Fundamentals of Neural Network*. Prentice Hall; US Ed edition (January 15, 1994)

LEJAREN A., Jr. HILLER & ISAACSON, Leonard M. *Experimental Music. (Composition with an Electronic Computer)*. Mc Graw Hill (1959)

LERDAHL, Fred; JACKENDOFF, R. *Teoría generativa de la música tonal.* Akal. (1996).

MANNING, Peter. *Electronic and Computer Music*. Oxford University Press; Revised edition (January, 2004).

MINSKY, Marvin Lee. *Computation: Finite and Infinite Machines (Automatic Computation)*. Prentice Hall (June 1967)

MIRANDA, Eduardo. *Composing Music with Computers*. Focal Press; Bk&CD-Rom edition (May 30, 2001).

MIRANDA, Eduardo. *Computer Sound Design: Synthesis Techniques and Programming.* Focal Press; 2 edition (October, 2002).

MOORE, F. Richard*. Elements of Computer Music*. Prentice Hall PTR; (February 17, 1998).

NEUMAN, John. *Theory of Self-reproducing Automata* . University of Illinois Press. (London, 1966)

POHLMANN, Ken C. *Principles of Digital Audio*. McGraw-Hill/TAB Electronics; 5 edition (April 20, 2005).

POLYA ,George. *How to solve it.*Princeton University Press. (April, 2004)

ROADS Curtis, Composers and the Computer (Computer Music & Digital Audio Series). William Kaufmann Inc. (June, 1985).

ROADS Curtis, *Foundations of Computer Music*. The MIT Press; Reprint edition (February 6, 1987).

ROADS Curtis, *Musical Signal Processing*. Aa Balkema (June, 1997).

ROADS Curtis, *The computer music tutorial*. The MIT Press; (1996).

ROWE, Robert. *Machine Musicianship*. The MIT Press; Bk&CD-Rom edition (March 1, 2004).

SCHWANAUER Stephan;LEVITT David. *Machine Models of Music.* The MIT Press (January 8, 1993)

SIMON, H. A., and SUMNER, R. K. 1993. *"Patterns in Music"* en: SCHWANAUER, S.M. & LEVITT, D.A. *Machine Models in music.* Pags: 83-110. MIT Press. Cambridge (1993)

STEIGLITZ, Ken. *A Digital Signal Processing Primer*. Prentice Hall; 1st edition (January 5, 1996).

WEISBERG, R. W. *Creativity: Beyond tha Myth of Genius.* W. H. Freeman and Company, New York, USA. (1993).

WINKLER,Todd. *Composing Interactive Music: Techniques and Ideas Using Max*. The MIT Press; Bk&CD-Rom edition (January 26, 2001).

ZOLTZER, Udo. *DAFX:Digital Audio Effects*. John Wiley & Sons; 1st edition (May 15, 2002).

ZOLTZER, Udo. *Digital Audio Signal Processing*. John Wiley & Sons (August, 1997).

Artículos y Revistas:

BOTTONI, P.; CAMMILLI, M., FARALLI, S. Generating multimedia content with cellular automata. Multimedia, IEEE Volumen 11, Número 4, Oct.-Dic. 2004 Pags:78 – 83.

J. Sabater, J. L. Arcos, R. López de Mántaras. Using Rules to support Case-Based Reasoning for harmonizing melodies. Conferencia. IIIA, España. Falta el año

LOPEZ de MANTARAS, Ramón; ARCOS Josep Lluis. AI and Music. From composition to expresive performance. AI Magazine archive

Volume 23 , Pages: 43 – 57. 2002.

Jennings R., Sycara K., Wooldridge M., (1998) "A Roadmap of Agent Reasearch and Development", Autonomous Agents and Multiagent Systems, vol 1, nº 1 pp 275-306. Springer

Wooldridge M. y Jennings N. R. (1995) Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, vol. 10(2) pp. 115-152, 1995. Springer

Rader Gary M. "*A method for composing simple traditional music by computer".* Communications of the ACM Volume 17 , Issue 11
Pages: 631 - 638. ACM Press New York, NY, USA. (November 1974)

Rader Gary M. *"A method for composing simple traditional music by computer"*. Machine models of music. Pages: 243 - 260.MIT Press Cambridge, MA, USA (1992).

Minsky Marvin. *Music, Mind, and Meaning*. Computer Music Journal, Fall 1981, Vol. 5, Number 3

Revistas on-line:

Miguel Gómez-Zamalloa Gil *Sistema de composición musical automática Aproximaciones Preliminares.* www.guru-games.org/people/pedro/aad/miguel_zamalloa.pdf

Bussman S. Y Müller H. J. (1993) A communication achitecture for cooperating agentes. Compt. Artif. Intell. 12(1), 37-54.

Medios electronicos:

VARIOS. Wikipedia, la enciclopedia libre. Ver. Español. http://es.wikipedia.org. Wikimedia Foundation, Inc.

conferencias

CHAN Michael. Automated composer of style sensitive music. 2004. Faltan demás datos