Samaruga, Lucas Martín

Un modelo de representación y análisis estructural de la música electroacústica

Samaruga, Lucas Martín, Repositorio Institucional Digital de Acceso Abierto,
Marzo del 2015, 230 pp. ,
http://ridaa.unq.edu.ar,
Universidad Nacional de Quilmes, Secretaría de Posgrado,
Doctorado en Ciencias Sociales y Humanas

Un Modelo de Representación y Análisis Estructural de la Música Electroacústica

TESIS DOCTORAL

Lucas Martín Samaruga

samarugalucas@gmail.com

Resumen

Esta Tesis versa sobre la composición con medios electroacústicos digitales en lo que se refiere a sus formas de representar el material sonoro y musical. Su hipótesis más general apunta a que los recursos materiales generalizables dependen del medio de producción y las abstracciones empleadas tanto para representar los elementos técnicos como las concepciones teóricas, ya sean, aplicadas al medio como imposición externa o derivadas de este como noción subyacente.

El principal resultado de esta Tesis es un modelo de representación de materiales/constructos musicales fundamentado teóricamente en los recursos técnicos propuestos por las ciencias informáticas, los conocimientos de la teoría musical y la práctica instrumental, y los principios perceptivos de las configuraciones sonoras. Cada uno de estos campos aporta un punto de referencia a la conceptualización de las estructuras de datos y los procesos informáticos que actúan como la representación material de las entidades musicales. El principio que define el modelo de representación material es la recursión. Las entidades musicales se definen mediante un conjunto finito de primitivas que pueden ser combinadas recursivamente de distintas maneras a diferentes escalas temporales. Según la escala temporal a la que actúe una determinada primitiva o configuración de relaciones compuestas estará dado el significado de sus cualidades musicales.

Independientemente del género de la música estudiada, este modelo de representación puede ser empleado tanto para la composición como para el análisis de la música electroacústica, puesto que el plano de representación desarrollado es estructural/gramatical y, por lo tanto, independiente del significado estético. Si bien su aplicación más directa se dirige a los lenguajes de programación para síntesis de sonido y composición algorítmica, debido a la generalidad propia de las técnicas informáticas, su aplicación es extensible al empleo de *software* especializado por medio del análisis de las concepciones/abstracciones implementadas.

Palabras clave: música electroacústica; programación; representación; análisis musical; composición musical.

Director:

Dr. Oscar Pablo Di Liscia

Co-Director:

Ing. Gustavo Basso

Agradecimientos

Mis agradecimientos inmediatos van dirigidos a las personas e instituciones que hicieron posible la presentación de esta Tesis. Quiero agradecer a mi Director, Pablo, por su infinita ayuda, paciencia y constante dedicación. A mi Co-Director, Gustavo, por todos sus aportes, no menos pacientes y constantes. Al CONICET, por brindarme la posibilidad de desarrollar estas ideas. A la UNQ y el Departamento de Ciencias Sociales por recibirme en su casa y brindarme toda su ayuda y los recursos necesarios. A la UNLP, por la excelente formación que me brindó. Y a Mónica, por iniciar el camino que me trajo hasta aquí.

Índice

Índice de Figuras

Índice de Tablas

CAPÍTULO 1: Presentación

Introducción

La notación tradicional es, según Wishart (1996), un modelo de representación analítico de ciertos componentes que conforman el resultado sonoro del discurso musical. Sin embargo, nota el autor, este modelo se encuentra con problemas estructurales a la hora de representar gestos y procesos continuos con cierta precisión o incluso falla por completo y omite representarlos. Para él, la notación tradicional se basa en la abstracción y la simplificación del sonido concreto. Descompone y discretiza ciertos parámetros, de manera jerarquizada, en relación a los recursos instrumentales para los cuales fue creada. Centrada en la altura como parámetro estructurante, la notación tradicional omite la representación de las sutilezas tímbricas, rítmicas y dinámicas del sonido real, descompone las fluctuaciones continuas en objetos discretos, idealizados, denominados notas musicales.

Además de los recursos de la notación musical tradicional y la representación gráfica del fenómeno físico, en la actualidad existen nuevos paradigmas que surgen, en su gran mayoría, de los recursos informáticos empleados para la síntesis y el procesamiento de sonido. Debido a que el empleo de la computadora, como medio instrumental de propósito general, solo define los elementos utilizados para su funcionamiento a bajo nivel, es necesario crear las abstracciones de más alto nivel para representar el sonido y las estructuras musicales. Es así que surgen conceptos como, por ejemplo, los de *unidad generadora*, *canal*, *bus* y *señal de control* que son implementados sistemáticamente por todas las aplicaciones informáticas y los entornos de programación para sonido. Durante el desarrollo de este trabajo se verá que son estas abstracciones informáticas las que pueden ser entendidas analíticamente como recursos instrumentales representativos de estructuras sonoras en el contexto de la música por computadora.

Debido a la naturaleza del medio informático, es posible representar la organización temporal de los sonidos producidos de manera equivalente a la partitura tradicional e incluso dejar registro de nociones estructurales que no forman parte del resultado sonoro, que pueden ser ideadas y representadas equivalentemente de diversas maneras, o que pueden ser interpretadas perceptivamente de manera ambigua.

La división de los programas entre orquesta y partitura es una forma clásica de organizar la lógica de una composición electroacústica desarrollada en entornos para síntesis de sonido y composición algorítmica/asistida. La orquesta se encarga de definir los procesos de síntesis y la partitura especifica los eventos que disponen esos procesos temporalmente. Es un modelo relativamente simple desarrollado junto con las primeras

aplicaciones informáticas de síntesis modular, en particular, los lenguajes de la familia *Music N* (Roads 1995) de los cuales *Csound* (Vercoe 1986) (Roads 1995) es su referente actual. Este modelo se basa en el análisis de los medios de producción musical tradicionales. De estos abstrae la división empírica entre lo que genera sonido y lo que controla la generación de sonido como metáfora de un medio de producción conocido. Sin embargo, para la generación de sonido es necesario controlar los procesos que intervienen dentro de la **abstracción instrumental** (ver **Capítulo 4**). Esto provee mayor grado de control y mayor grado de precisión sobre el control del sonido resultante, a costa de cierto nivel de complejidad debido a la gran cantidad de datos necesarios.

En la actualidad, existen entornos de programación que posibilitan la integración del paradigma orquesta-partitura permitiendo la composición algorítmica tanto de los procesos de síntesis como su organización temporal. Desde el punto de vista de la programación musical, este cambio de paradigma posibilita un desarrollo más flexible de los procesos musicales a distintas escalas y, en cierta forma, unifica los elementos de representación.

Sin embargo, aunque los entornos de programación para síntesis de sonido y composición algorítmica/asistida ofrecen el mayor grado de flexibilidad con respecto a la producción y organización del sonido, la representación de sus **abstracciones instrumentales** predomina por sobre las **abstracciones temporales**. En ellos, la lógica algorítmica de producción de sonido está mucho más desarrollada y estandarizada que su representación estructural-temporal.

Este hecho muchas veces dificulta el entendimiento de la relación entre los recursos instrumentales y las concepciones compositivas. La gran variedad de entornos y herramientas informáticas disponibles generan cierto grado de diversidad y complejidad al tratar de abarcarlos. Es por eso que en esta **Tesis** se considera conveniente desarrollar un modelo de representación que sea generalizable a las técnicas y los entornos de producción existentes. Por medio del análisis del medio informático como medio instrumental y de las posibles concepciones compositivas que este posibilita, se pretende lograr un mayor grado de entendimiento sobre la composición musical con medios electroacústicos.

Para poder desarrollar un modelo de representación es necesario el análisis previo de los recursos técnicos disponibles, pero también de las nociones que definen los elementos que están dentro del dominio del problema. De los conceptos musicales que explican la naturaleza del problema, el que resulta de mayor interés para el desarrollo de este trabajo es el de **material musical**, puesto que denomina y abarca los elementos fundamentales de cualquier discurso sonoro. Aunque la noción de **material musical** puede ser dependiente de un estilo o período, serán tenidos en cuenta solo los aspectos que definan comportamientos universales en relación a la naturaleza del sonido,

el medio instrumental de producción y sus formas de representación analítica. Es por esto que, para llegar a una definición de **material musical** adecuada a los entornos informáticos, es necesario analizar las distintas ideas que se desarrollan a través de la práctica musical en torno al concepto de material.

Antes de ingresar en la descripción de los elementos que componen el cuerpo de esta **Tesis**, se considera conveniente introducir tres nociones simples y elementales, que dieron origen a este trabajo y guiaron las intuiciones primitivas. Estas son: la concepción paramétrica del sonido, que significa la capacidad teórica o práctica de construir, reconstituir o transformar elementos sonoros a partir de un conjunto de variables relacionadas; la interpretación y el control que se tiene sobre el sonido, descompuesto y representado mediante ese conjunto de variables, en relación al medio instrumental empleado; y las restricciones que actúan como principio de coherencia, tanto de los recursos instrumentales como de las técnicas compositivas.

Representación paramétrica

El medio de producción sonora, en este caso la computadora, define los recursos manipulables y sus posibles combinaciones. Este es un hecho práctico que puede afectar las características de la producción sonora resultante al facilitar o dificultar, debido a las restricciones pre-impuestas por un sistema sistema, la realización de determinadas ideas musicales (Magnusson 2010a 2010b). Por otra parte, los condicionamientos impuestos por el medio definen también nuevos recursos de representación y manipulación de sonido, los cuales afectan las noción de **material musical** en relación a su concepción histórica. Una de las nociones que se desarrollan a lo largo de esta **Tesis** es la de **comportamiento abstracto**[1] como característica de la representación paramétrica en relación a los procesos musicales e informáticos.

Desde mediados del siglo XX, en parte de la música acústica y gran parte de la música electroacústica, comienza a tener mayor importancia la **representación paramétrica** del sonido.

Esta consiste en la representación del sonido por medio de la representación disociada de sus parámetros constitutivos. Ya no predomina la concepción de estructuras simbólicas relacionadas a los medios instrumentales tradicionales y los recursos interpretativos, sino que se vuelve posible la abstracción total del sonido como entidad independiente. Es importante notar, que aquí no se está haciendo referencia a la noción de música acusmática definida por Schaeffer (1988) en relación a la pérdida de referencialidad de sonidos grabados, sino a su aplicación a cualquier entidad sonora

generada, sin importar la metodología empleada.

Las técnicas de **síntesis abstracta** de sonido, en oposición a las técnicas por modelado físico (Bilbao 2009), son el paradigma de este cambio. Las técnicas de **síntesis abstracta** refieren al empleo de modelos matemáticos para generan funciones o sistemas que representan la señal resultante. Esta clasificación está en oposición a la síntesis por modelado físico que emplea modelos de representación de los objetos físicos que generan determinadas señales acústicas en base al comportamiento de los materiales al ser excitados. Ejemplos de diversos tipos de **síntesis abstracta** son: la síntesis aditiva, la síntesis sustractiva, la síntesis por frecuencia o amplitud modulada y la síntesis granular. Mientras que la síntesis por modelado físico representa la señal resultante mediante la cualidades de un elemento material y la interacción con este, la **síntesis abstracta** representa la señal resultante mediante la combinación de recursos abstractos. Se disocia, en cierta forma, el medio de producción, entendido convencionalmente como instrumento musical, del resultado sonoro[2].

La teoría de la síntesis por modelado físico se conoce incluso antes que las técnicas de **síntesis abstracta** (Bilbao op. cit.). Estas últimas surgen a raíz de la capacidad de computo reducida provista por los primeros sistemas informáticos y la necesidad de diseñar recursos computacionalmente eficientes. Es como consecuencia de esta necesidad que se desarrollan nuevas técnicas de producción y representación del sonido basadas en los recursos provistos por los medios informáticos.

El sonido generado por **síntesis abstracta**, al estar desprovisto de su referencialidad con respecto a las fuentes sonoras, pierde gran parte de su simbolismo tradicional, y esta pérdida hace que sea necesario crear nuevas abstracciones que reemplacen, de manera analíticamente diferente, los elementos con los cuales se elaboran los materiales musicales. Es por esto que desde los inicios de las investigaciones que llevaron al desarrollo de esta **Tesis** se adoptó operativamente el concepto de **información sonora parametrizada**, como un flujo de información sonora, independiente del medio de producción y del medio y formato de transmisión, sobre el cual se tiene control y al cual se le pueden asignar significados diversos puesto que son recursos inherentemente abstractos. Durante el desarrollo del presente trabajo se podrá observar como estos recursos abstractos evolucionan en entidades musicalmente significativas que se relacionan intrínsecamente con los recursos informáticos.

Para el análisis de las abstracciones musicales se considerará la **dimensión temporal** como un rasgo fundamental que, en muchos casos, es lo que define las cualidades objetivables de los recursos abstractos como materiales musicales. La significación de los recursos se define principalmente por la escala temporal en la cual

actúan. Para el análisis de estos principios se empleará la clasificación temporal desarrollada por Curtis Roads en su libro *Microsound* (Roads 2002). Esta clasificación diferencia distintas escalas de duraciones que van de lo microscópico a lo macroscópico y las asocia a elementos musicales representativos. Es adoptada como medida de referencia puesto que simplifica el espacio temporal en referencia a estructuras significativas tanto para la percepción como para la composición sonora. Las escalas temporales empleadas, desde las más pequeñas a las más grandes, son las siguientes: 1) *muestra*, el tiempo definido por la frecuencia de muestreo que representa el nivel atómico del sistema computacional; 2) *micro*, comienza en el límite de la percepción temporal y se mide en milisegundos; 3) *objeto sonoro*, va desde las fracciones de segundo hasta los varios segundos, es la escala en la cual se define la percepción de eventos sonoros tradicionalmente asociados con elementos musicales básicos; 4) *meso*, se mide en segundos o minutos y es la escala a la cual se definen los procesos formales en la música y, 5) *macro*, se mide generalmente en minutos u horas y es representativa de estructuras formales de largo aliento o piezas musicales como un todo. No se entrará en detalle sobre las definiciones de cada escala salvo de manera práctica o en caso de ser necesario aclarar situaciones que puedan resultar ambiguas.

En los entornos de programación para síntesis de sonido aparecen abstracciones que representan distintos aspectos constitutivos del material sonoro, e.g. las envolventes, los algoritmos que generan o procesan sonido, los eventos de control, las líneas temporales. Pero ninguno de esto elementos representa la cualidad de un parámetro determinado sino el comportamiento que define esa cualidad al estar aplicado a un determinado parámetro. Esto es lo que se denomina como **comportamiento abstracto**. Por ejemplo, una envolvente puede ser espectral y definir la cualidad tímbrica de un sonido en un momento determinado o puede ser dinámica y definir la evolución temporal de la amplitud de un sonido. Las envolventes espectrales definen estadísticamente el comportamiento de un parámetro dentro de un único instante, mientras que las envolventes dinámicas definen el comportamiento a lo largo del tiempo. Generalmente, estas últimas surgen y se catalogan por su simbolismo en relación al comportamiento dinámico de los instrumentos acústicos, e.g. envolvente percusiva o de ataque-decaimiento-sostenimiento-liberación (conocida como ADSR por sus siglas en inglés), pero su uso no está restringido al control de la amplitud, son instancias específicas del concepto de envolvente, con una determinada forma, que afectan a la evolución temporal de la amplitud y se aplican a una determinada escala temporal. La misma abstracción se puede emplear para controlar distintos elementos que pueden actuar semánticamente de distintas maneras. Estas abstracciones, al estar definidas de manera general desde el dominio informático, son referidas por Bresson (2007) y Winsor (1991) como *nivel sub-*

simbólico o *gramatical* respectivamente, puesto que el concepto de envolvente, en los ejemplos anteriores, no adquiere significado musical hasta que se aplica específicamente a un parámetro en una determinada escala temporal[3]. Sin embargo, el sonido se vuelve multidimensional y no hay, *a priori*, parámetros que sean considerados más relevantes que otros, la abstracción manipulable está a nivel sub-simbólico y por esto puede adquirir relevancia como entidad musical general[4].

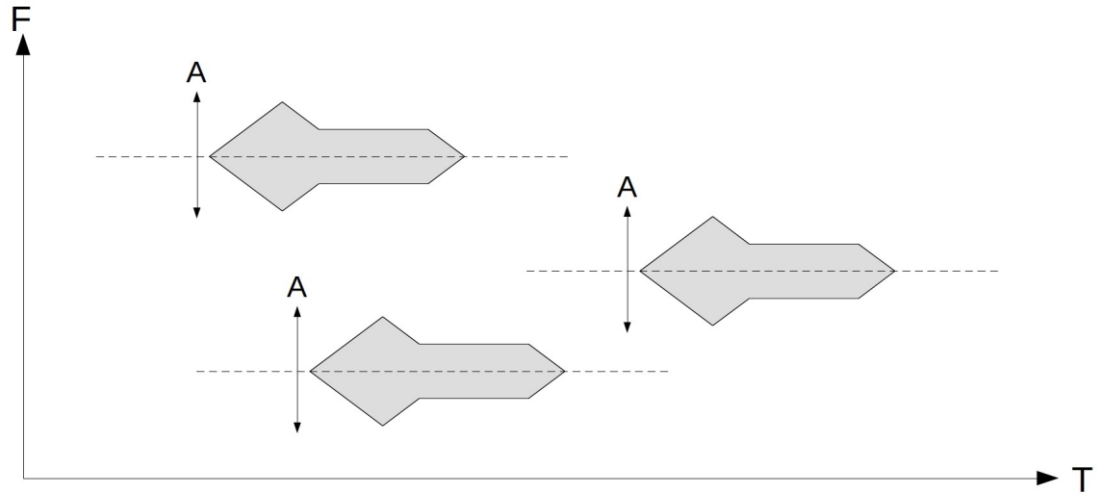


Figura 1 Representación esquemática en dos dimensiones de la frecuencia y la duración de tres eventos sonoros en relación a los ejes vertical y horizontal externos, con la adición de la envolvente dinámica simétrica como tercer dimensión incrustado como sub-gráfico que representa la amplitud en relación al tiempo. Este tipo de representación es común en entornos como HighC (Thiebaut et al. 2008), derivado del UPIC de Xenakis (Squibbs 1996).

Con respecto a la expresión gráfica del sonido esto implica un cambio importante del eje de representación. En la notación musical tradicional, los dos parámetros más importantes son la altura y la duración y ambos se representan elementos específicos que se combinan en el símbolo “nota musical” representado en dos dimensiones de forma relativa. La duración de las notas es métricamente proporcional y relativa a la posición en el orden sucesivo, mientras que la altura es relativa al espacio pentagramado y la clave de referencia. En los entornos de **síntesis abstracta** lo que usualmente se denomina nota o evento no necesariamente implica la jerarquización de ninguno de los parámetros que representa. Al estar descompuesto en componentes de igual jerarquía, la entidad sonora pasa a ser un conjunto de comportamientos y se vuelve difícil implementar una representación gráfica sintética como la notación tradicional. Sin embargo, al igual que en la notación tradicional, se puede simplificar la complejidad visual seleccionando, de manera variable, él o los parámetros que sean más representativos de un material sonoro particular como se muestra en la Figura 1.

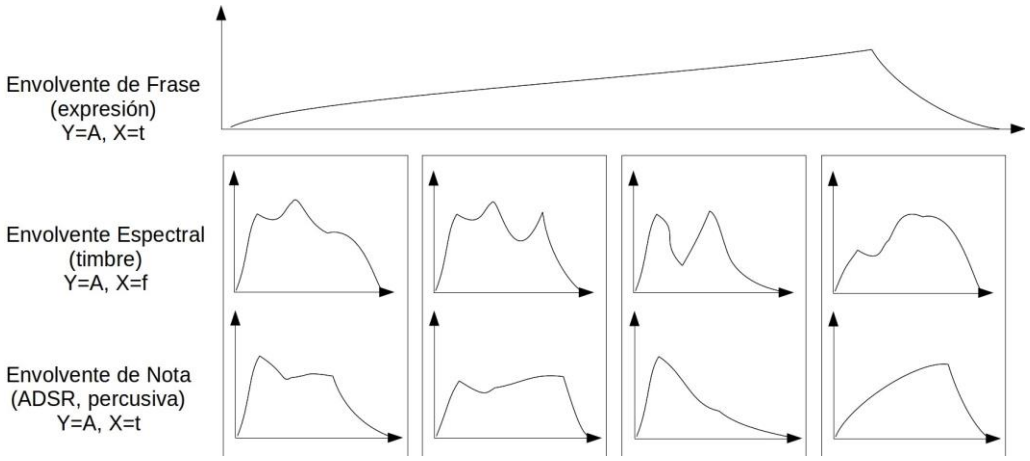


Figura 2 Representación estructural de tres parámetros significativos, de abajo hacia arriba: envolvente dinámica por nota, envolvente espectral (invariable) por nota y envolvente de expresión dinámica por frase que afecta una sucesión de cuatro notas.

Se puede decir que las funciones envolventes actúan como el comportamiento abstracto de un determinado parámetro y, por lo tanto, pueden ser consideradas como procesos musicales. Al estar claramente delimitadas dentro de una escala temporal se pueden relacionar con otras entidades que actúan de manera similar, ya sea sobre el mismo parámetro, en el caso más simple, u otro. Por ejemplo, si consideramos una envolvente dinámica a nivel de nota musical, esta puede volverse la abstracción representativa del objeto nota y relacionarse con las envolventes dinámicas de otras notas en una sucesión melódica. Si sobre esta sucesión melódica se aplica una envolvente de frase musical, la abstracción que controla el parámetro amplitud es la misma pero actúa a distinta escala temporal aunque sobre los mismos elementos (Figura 2).

Las estructuras paramétricas son multidimensionales por naturaleza, cada dimensión puede ser controlada de manera simultánea desde distintas escalas temporales. El resultado objetivable de una estructura paramétrica puede volverse el parámetro de entrada de otra entidad a igual o distinta escala temporal. Por ejemplo, la señal resultante de procesos de síntesis sobre los cuales se manipulan parámetros básicos, puede ser tomada como parámetro de otro proceso del cual se controle el mismo u otro parámetro. En la práctica actual, un ejemplo claro de esto es la síntesis por frecuencia modulada donde se varía la frecuencia y la amplitud de un oscilador cuya señal resultante se toma como parámetro de otro oscilador. Este ejemplo se vuelve más interesante si se considera además que el parámetro del sonido resultante que se está controlando no es la frecuencia sino el timbre.

Según a qué escala temporal estén actuando las envolventes y las señales resultantes de procesos de **síntesis abstracta**, estas definen distintas cualidades del resultado sonoro. Las mismas abstracciones pueden estar actuando de manera simultánea a distintas escalas temporales sobre los mismos elementos (por ejemplo, la envolvente dinámica de frase, las envolventes espectrales y las envolventes dinámicas por nota en la Figura 2). Esta continuidad de las escalas temporales que se produce entre los parámetros controlables y la resultante sonora es un factor importante a tener en cuenta en el desarrollo de la representación de la música electroacústica. Con las técnicas de producción sonora actuales, el compositor puede tener la necesidad de trabajar a niveles de *microsonido* (Roads 2002) y al mismo tiempo necesitar de estructuras que posibiliten el control de los objetos sonoros y las escalas *meso* y *macro* temporales.

Los conceptos que definen la concepción paramétrica de la música electroacústica en relación a la tradición musical se pueden resumir en los siguientes puntos:

- Se pierde el simbolismo “nota musical”.
- El sonido es multidimensional y está representado por la composición paramétrica.
- Las abstracciones manipulables no representan parámetros específicos.
- Las estructuras son *multidimensionales* y las dimensiones son *multitemporales*.

La interpretación y el control paramétrico

En el modelo partitura-intérprete el ejecutante tiene cierta libertad para interpretar las ideas sonoras del compositor. Diferentes estilos musicales o concepciones compositivas permiten distintos grados de libertad que se aplican a uno o varios parámetros. Por ejemplo, en algunos estilos interpretativos de la música renacentista y barroca se deja librada la ornamentación a criterio del intérprete, aun siendo un recurso afecta un parámetro estructuralmente tan importante como es la altura y su organización melódica. Aunque sean considerados ornamentos, elementos accesorios a la estructura musical, estos pueden adquirir gran relevancia a nivel de superficie pudiéndosedistinguir diferencias sustanciales entre distintas ejecuciones. Más comúnmente, el *tempo* y la intensidad del sonido son los parámetros que quedan a criterio del intérprete en cuanto a la ultimación de sus detalles. Incluso en la interpretación de la música del período romántico, las duraciones, como relaciones proporcionales, pueden ser alteradas por el intérprete mediante el empleo del *rubato*. El *rubato* es una fluctuación en el *tempo* de la pieza, se suele aplicar a escala *meso temporal* afectando la percepción de las relaciones proporcionales entre duraciones y, por lo tanto, afectando al ritmo del **material musical**.

La cultura interpretativa en distintos estilos demuestra que incluso los parámetros fundamentales, como la altura, la duración y la intensidad, pueden variar dentro de ciertos límites y esas variaciones son consideradas de manera positiva. En la música del siglo XX se le presta mayor atención a estos factores y se elaboran distintas ideas. Por un lado, están las corrientes que demandan del intérprete la mayor fidelidad posible con respecto a las intenciones del compositor plasmadas en la escritura musical, como se da en las corrientes relacionadas con el serialismo. Por otro lado, se explota el concepto de indeterminación en la notación musical dando mayor libertad al interprete para definir parámetros básicos del sonido e incluso casi su total organización a nivel *meso temporal*. En todos estos casos se puede apreciar que existe la indeterminación en mayor o menor grado. También se puede observar que varía la escala temporal y estructural dentro de la cual se le permite al intérprete realizar modificaciones.

El empleo de técnicas extendidas y la composición dedicada a solistas en particular, también afecta al control sobre la realización sonora de una pieza musical aunque, en este caso, es una retroalimentación que sucede durante el proceso compositivo (Sigal 2014).

Si en lugar de analizar el problema desde las nociones compositivas/interpretativas se analiza desde los recursos instrumentales se puede apreciar que existen ciertas limitaciones en cuanto a la escala temporal sobre la cual el intérprete puede actuar para modificar los materiales. En la música instrumental el intérprete suele tener control expresivo dentro de las escalas: objeto sonoro, meso y macro temporales. Las escalas de objeto sonoro y meso temporal abarcan las notas musicales, la melodía, las frases musicales, la textura, la forma, etc. La escala macro temporal actúa a nivel de la pieza en su totalidad o un conjunto de piezas si, por ejemplo, se tiene en cuenta la selección del instrumento, el recinto, las técnicas de grabación, etc.

La precisión en el control de los parámetros en una pieza musical se puede variar de muchas maneras. Si la obra es para piano, tocado de manera tradicional y sin modificaciones en el mecanismo, no se puede cambiar el comportamiento tímbrico salvo que se cambie el piano o la sala. Si la obra es para guitarra, se puede cambiar el timbre a nivel de nota musical, aunque el sonido sigue siendo del tipo “guitarra”, este puede variar de manera notable. Si se quiere variar el timbre a unidades temporales más pequeñas no es posible cambiar de instrumento en el primer caso, pero si es posible cambiar el toque en el segundo. Sin embargo, un cambio de toque arbitrario a nivel de eventos individuales puede ser considerado técnicamente erróneo para la interpretación de una determinada pieza.

Estos ejemplos reflejan el tipo de control que el medio de producción sonora tiene sobre los posibles resultados. Ya sea que se emplee el modelo partitura-intérprete o que estos roles se aúnen en el modelo de producción de la música electroacústica. Es el

medio el que define forzosamente estas posibilidades a distintas escalas temporales las cuales pueden variar según la flexibilidad de este.

Delimitación de los recursos compositivos

Para abordar un problema de creación musical es necesario definir los recursos con los cuales se va a trabajar. En la composición acústica, la elección del conjunto instrumental (e. g. cantidad y tipo de instrumentos) y las técnicas instrumentales (e. g. convencionales o extendidas) a emplear para una determinada pieza es uno de los primeros factores que se definen, y es, tal vez, el que más condiciona técnicamente el resultado sonoro. Esto se debe a que el sonido está definido tímbricamente por el instrumento que lo genera pero también porque los recursos particulares de cada instrumento afectan al comportamiento de los parámetros del sonido. No todos los instrumentos pueden generar sonido en todo el rango audible o cambiar la alura gradualmente y de manera continua, algunos pueden hacerlo dentro de un determinado rango de alturas, otros pueden hacerlo mediante técnicas extendidas que afectan otras cualidades como la emisión (en relación a la sonoridad) y el timbre del sonido en general. Este tipo de pensamiento musical se diferencia del pensamiento de estructuras musicales abstractas o estructuras semánticas (e. g. nota, acorde, motivo, frase o desarrollo temático en la música tonal), predominante en la teoría sobre los estilos musicales anteriores a la segunda mitad siglo XX. Se podría decir que en esta forma de pensamiento abstracto de estructuras semánticas musicales, la instrumentación está disociada y es simplemente el medio necesario para la realización de las ideas musicales (e.g. una pieza para piano orquestada o una misma pieza tocada por distintos instrumentos no pierde su identidad como obra musical). En la práctica es poco probable que se emplee solamente una de estas dos concepciones, incluso en la música previa a la segunda mitad de siglo XX, puesto que de requerirse, imaginativamente, recursos sonoros distintos, es posible agregar instrumentos a un orgánico preexistente o pensar la orquestación como una etapa posterior de la elaboración musical en la cual la composición está centrada en otros aspectos (e.g. tímbricos) que pueden hacer que las características de las estructuras semánticas pre-elaboradas se vean drásticamente variadas de manera intencional.

Todos estos factores pueden definir una filosofía, estética o un estilo musical particular, pero lo que importa para este trabajo es que están relacionados con la técnica y constituyen el camino necesario para la elaboración musical. El uso de la orquestación en relación a la concepción de estructuras semánticas musicales es solo un ejemplo del problema. Incluso dentro de las estructuras semánticas, aunque estén fuertemente relacionadas al medio instrumental, pueden variar las nociones paramétricas del sonido o

los elementos definidos como estructuras musicales básicas (e.g. melodía, armonía, ritmo, textura). Como bien estudia Wishart en su libro *Sonic Art* (Whishart 1996), la notación musical es otro factor que actúa en ambos sentidos, como medio de representación de elementos abstractos imaginados y como condicionante de la imaginación. Pero el análisis de este autor se aproxima más a las nociones paramétricas actuales del sonido haciendo énfasis en las restricciones que propone el modelo analítico de la notación convencional debido a la concepción discreta de los elementos musicales básicos, enfocados desde el nivel temporal definido por las acciones instrumentales. La escala temporal del material analizado y considerado como atómico por la notación musical tradicional, que está en relación al diseño de los instrumentos y las posibilidades de control expresivo por parte del instrumentista, generalmente no baja del nivel de eventos sonoros y carece de precisión en la representación de comportamientos continuos incluso a mayores niveles temporales.

La música electroacústica surge del empleo de nuevos medios instrumentales pero, aunque posibilita mayor precisión en cuanto al control instrumental, hereda las concepciones de los medios preexistentes y, en algunos casos, también las restricciones. Por ejemplo, el modelo orquesta/partitura, y las restricciones instrumentales del protocolo MIDI (siglas en inglés de Interfaz Musical para Instrumentos Digitales) (Penfold 1992). En esta música no basta con definir las abstracciones equivalentes al instrumental en la música acústica puesto que, desde el principio, el control instrumental que se tiene del sonido está basado en abstracciones informáticas, las cuales deben ser definidas previamente por medio de la programación. Como metodología compositiva se puede decidir emplear una abstracción informática en vez de otra, o pre-seleccionar las técnicas de síntesis para la elaboración de una pieza, eso es una **restricción voluntaria** de los recursos instrumentales o de la representación empleada, puesto que, al ser programables, ambas pueden variar en mayor o menor medida.

Para un desarrollo más extenso sobre el tema, en relación al estudio y la clasificación de las restricciones y facilidades propuestas por el *software* y su modo de empleo, se pueden consultar los trabajos de Magnusson (2007), (2010a) y (2010b). Estos son tenidos en cuenta por su agudeza y precisión conceptual, pero no serán empleados sistemáticamente como marco teórico general en este trabajo.

Esta **Tesis** hace referencia a la restricción como principio de coherencia. Las restricciones se pueden clasificar simplemente por su origen: intelectual (interna/voluntaria) o instrumental (externa/impuesta). Las restricciones intelectuales son voluntarias y están definidas por la elección del compositor. Determinar el empleo de un tipo de **material musical** en relación a una técnica o estilo (e.g. acordes, escalas, sonidos grabados, sonidos sintetizados) implica una **restricción voluntaria**. Una **restricción impuesta** es la

que provee un determinado recurso instrumental (e.g. instrumento acústico) o un conjunto finito de combinaciones posibles dentro de un contexto (e.g. acordes triádicos diatónicos).

Una **restricción voluntaria** puede implicar una **restricción impuesta** de manera causal. Por ejemplo, en la elección de un repertorio de alturas, un orgánico instrumental o una técnica de síntesis. Cada elección restringe los recursos disponibles (idealmente infinitos) y estos, a su vez, restringen las posibles manipulaciones que el compositor puede efectuar. **La restricción se convierte en principio de coherencia al definir el repertorio de relaciones posibles de manera que sea intelectualmente abarcable. Esta definición del repertorio de relaciones posibles hace explícita la elección de las relaciones incluidas en el discurso musical por sobre otras no incluidas.**

Se emplearán, entonces, dos categorías de restricciones:

- **Restricciones impuestas**: Definidas por el sistema de producción/representación empleado.
- **Restricciones voluntarias**: Definidas por el compositor dentro del conjunto definido por el sistema de producción/representación.

Si lo que se quiere es concretar un modelo de representación y organización de todos los recursos disponibles, es necesario tratar de abarcar o prever la mayor cantidad de recursos combinables para poder dejar las restricciones que surgen de la metodología compositiva a elección del compositor. En la práctica es posible definir tipos y categorías de recursos pero es imposible prever todas las combinaciones posibles, incluso con un conjunto restringido de materiales y reglas.

Es por esto que es relevante definir las abstracciones informáticas básicas de la manera menos específica posible para que el compositor tenga mayor grado de control sobre los recursos y la creación de abstracciones musicales de mayor nivel. Si las abstracciones informáticas se definen a un nivel semántico heredado de la música acústica o de un estilo musical en particular, estas van a estar actuando como **restricciones impuestas** en un medio al que no pertenecen.

Hipótesis Generales

En relación a lo antedicho, es necesario mencionar que el análisis y el modelo propuesto en esta **Tesis** no están exentos de condicionamientos. Sus principales premisas son la concepción paramétrica y la organización jerárquica que definen las estructuras

materiales básicas que, a su vez, actúan como materia prima para la composición sonora. Este enfoque implica delimitaciones del campo de estudio del dominio del problema, que están condicionadas por el conocimiento proveniente de las ciencias de la computación. Pero siendo que los recursos informáticos son en sí el medio de producción y que, como medio, afectan la concepción que se tiene sobre el dominio del problema, estas premisas principales del sistema propuesto se pueden considerar como universales dentro de este contexto.

Por su parte, la ingeniería de *software* puede considerar que las abstracciones informáticas empleadas para la solución de problemas del mundo real no forman parte del dominio de esos problemas sino que conforman el repertorio de recursos técnicos necesarios. Sin embargo, como las abstracciones informáticas se definen en relación al problema, esta **Tesis** se refiere a las abstracciones fundamentales, definidas como tales por su simpleza y eficiencia conceptual y computacional, como los recursos técnicos que conforman las primitivas de la representación informática-musical, las cuales, a su vez, tienen arraigo en los conceptos fundamentales de las ciencias exactas y el conocimiento en general.

Las principales hipótesis que guían el desarrollo de los conceptos necesarios para definir el **modelo**, abarcan distintos aspectos orgánicos relativos al conocimiento de la materia de estudio y se expresan de la siguiente manera:

Hipótesis 1: Es posible definir un conjunto acotado de estructuras de datos que, al actuar de manera interrelacionada, generen un modelo de representación de los materiales sonoros, organizados como estructuras jerárquicas, aplicable a diversos entornos informáticos de manipulación de información sonora.

Esta es la hipótesis más general de todas, tiende a organizar el conocimiento sobre la representación musical que emplea medios informáticos. Se refiere principalmente a los recursos de abstracción de estructuras de datos y procesos de más bajo nivel, como elementos que conforman los recursos técnicos informáticos y las primitivas de representación sonora y musical ligados a estos.

Hipótesis 2: Dado que los recursos de representación de bajo nivel propuestos por el ordenador, como herramienta para manipular sonido, no se pueden cambiar, mientras más cercano sea el nivel de representación de las estructuras musicales básicas a las estructuras informáticas de bajo nivel, mayor será el grado de eficiencia a la hora de manipular y concebir materiales sonoros y musicales.

Esta segunda hipótesis se complementa con la primera. Se refiere a la adaptación de las formas de organización y representación en relación a cómo el sistema lo va representar y manipular. Los recursos y las restricciones técnicas propias de las

computadoras, como herramientas de cálculo, dependen de la arquitectura de bajo nivel definida por el *hardware*. Más allá de las distintas prestaciones de potencia computacional, la arquitectura define la lógica instrumental, que al ser aplicada a la composición musical, se vuelve el instrumento musical. Las estructuras de datos y los procedimientos básicos que se emplean para generar y representar información sonora y recursos compositivos muchas veces son considerados abstracciones de bajo nivel –sin llegar al nivel de los lenguajes *assembly*– que no se suelen contemplar como abstracciones musicales. Sin embargo, el empleo extendido de lenguajes de programación de propósito general para la composición musical pone de manifiesto la flexibilidad que estos proveen en comparación con herramientas de alto nivel diseñadas como utilidades específicas. Se considera que el empleo de abstracciones de bajo nivel por parte de los compositores es lo que provee el mayor grado de libertad expresiva, entendida como versatilidad a la hora de representar, elaborar y combinar concepciones sonoras y musicales.

Hipótesis 3: Debido a que el objeto de estudio (material sonoro o musical), que se considera como unidad de síntesis o análisis, es relativo al nivel de agrupamiento paramétrico seleccionado, los materiales musicales pueden ser definidos de manera general como entidades delimitadas y diferenciadas arbitrariamente que generan unidades de significado y que pueden ser descompuestas o combinadas para formar nuevas entidades a otro nivel de complejidad.

Esta es la hipótesis de recursión que manifiesta el sistema de conocimiento. Cuando se entiende a los materiales musicales como entidades compuestas a partir de un conjunto acotado de primitivas de representación, tanto las primitivas como los materiales compuestos por estas pueden ser considerados objetos de estudio y, a su vez, los materiales compuestos pueden ser entendidos como primitivas de representación a mayor nivel de abstracción. Los casos base empíricos son las estructuras próximas a la lógica computacional de bajo nivel, las funciones, los *buffers*, los eventos y los conjuntos. Estos definen ciertas propiedades lógicas y temporales básicas que se vuelven a emplear en la construcción de abstracciones de más alto nivel. Estas propiedades se transmiten a los agrupamientos de estructuras base como nuevos objetos de estudio. La hipótesis de recursión tiende a restringir los tipos de relaciones propios de un sistema sin perder generalidad.

Objetivos

El principal objetivo de esta **Tesis** es diseñar un sistema de representación preciso y general que luego pueda ser implementado tanto en la producción de *software*

como en las técnicas de composición y análisis de música electroacústica. Tal sistema de representación debe ser elaborado en base a los principios constantes que generan las relaciones que fundamentan el conocimiento musical. Para poder lograrlo es necesario cumplir con el desarrollo de los sub-objetivos expuestos a continuación.

- Explorar el desarrollo del estado del arte, en su división teórica y práctica, con respecto a las propuestas del *software* existente, su teoría subyacente y la producción musical en estos sistemas informáticos a fin de alcanzar un entendimiento más global de los principios de representación y los recursos propios de la informática musical y, de ser posible, de la música.
- Elaborar un marco teórico general para el estudio de la música electroacústica independiente de las propuestas específicas de la gran variedad de entornos existentes. El desarrollo de esta

Tesis se basa en el supuesto de que existen factores definidos por el entendimiento humano que, al ser aplicados de manera sistemática como técnicas de composición y análisis, pueden aclarar los principios estructurales de forma general, independiente de aplicaciones informáticas concretas.

- Sistematizar los principios empleados para la producción de música electroacústica de manera tal que puedan conformar una metodología de conocimiento aplicable a la composición y el análisis tanto de obras existentes como de nuevas producciones.
- Demostrar las ventajas de este modelo en su aplicación a la representación de datos empleada para la composición de música electroacústica.
- Demostrar la eficiencia del modelo aplicado al análisis de música electroacústica.

Metodología y Aportes

La materia estudiada por esta **Tesis** es resultado de la interacción histórica de varios campos de conocimiento. Está orientada principalmente a la composición musical con medios electroacústicos digitales y, por lo tanto, se centra en el conocimiento compositivo y las técnicas informáticas. Pero dentro de estos campos se consideran fundamentales los aportes de otras especialidades como la acústica musical, la filosofía, percepción y cognición musical, las técnicas de programación, el procesamiento digital de señales y la arquitectura de ordenadores por nombrar solo los más directos.

Para el desarrollo de los objetivos de este trabajo es necesario recurrir al análisis de los recursos teóricos y prácticos desarrollados en las últimas décadas. Este recorte se debe a varias razones. La computadora que conocemos hoy en día como herramienta compositiva no existía hace más de 30 años, el uso masivo del ordenador

generó una revolución en cuanto a la cantidad y originalidad de *software*, técnicas compositivas y recursos instrumentales. Si bien pioneros como Max Mathews, Lejaren Hiller y Leonard Isaacson fundaron las bases de la síntesis de sonido y la composición algorítmica (Roads 1995), desarrollos posteriores de herramientas y metodologías fueron dando forma a la "tradición" de la música electroacústica. La historia de esta breve tradición se basan principalmente en dos fuentes: los registros sonoros y los escritos sobre la técnica de estas producciones. No fue hasta hace un par de décadas atrás que comenzó a obtener relevancia la problemática del registro de la programación y las aplicaciones informáticas como parte de la producción musical (Bernardini y Vidolin 2005) (Barthelemy et al. 2009). Estos elementos comienzan a ser considerados en cierta forma como partituras e instrumentos necesarios para reproducir una pieza musical, especialmente con el desarrollo del procesamiento y la síntesis en tiempo real.

Por otra parte, muchos de los desarrollos artísticos, científicos, tecnológicos e incluso filosóficos, que aportan conocimiento a esta materia son de carácter netamente práctico. Esto se relaciona directamente con la idea de producción de *software* musical pensado como aplicación tendiente a solucionar un problema "conocido" de antemano. Muchas de las soluciones prácticas que se dieron durante el desarrollo de las técnicas de síntesis de sonido generaron un repertorio de nuevos recursos que cambió la manera de pensar e interactuar con el sonido. La aparición del ordenador produjo un cambio elemental muchas veces obviado en las propuestas prácticas: la computadora como medio instrumental no tiene precedentes. Si las técnicas conocidas de representación y manipulación sonora están basadas en los recursos instrumentales de épocas anteriores ¿cómo puede ser que el problema de la representación musical empleando nuevos medios digitales sea "conocido"? Es razonable pensar que, si el arte es en parte técnica y que si esa técnica depende del medio instrumental conocido, al introducir un nuevo medio instrumental en el arte puedan cambiar las técnicas y que este cambio afecte a su vez a la percepción y el conocimiento circundante.

El trabajo de esta **Tesis** consiste en el análisis y el desarrollo de recursos teóricos y prácticos. Para esto se recurre al análisis bibliográfico de la teoría de la representación de la informática musical que da cuenta de los problemas propios del medio informático. Se tienen en cuenta diversos enfoques que difieren en su direccionalidad. Los que parten de los recursos informáticos adaptando la representación musical y los que adaptan los recursos informáticos a la representación tradicional. De manera complementaria al análisis bibliográfico se estudian los entornos de producción existentes como elementos prácticos de los cuales se desprenden concepciones teóricas presentes, aunque muchas veces de manera implícita, en la lógica computacional.

El **Capítulo 2** trata sobre el estado del arte, tanto teórico como práctico. Es

principalmente expositivo de la teoría y los entornos que aportan elementos al conocimiento de la composición electroacústica.

El **Capítulo 3** es la exposición del marco teórico básico desarrollado por esta **Tesis**. Está fundamentado en el análisis, la interrelación y el desarrollo de conceptos presentes en las teorías de la representación informática, las teorías gestalticas de la percepción y sus evidencias empíricas, la teoría de la cognición musical y las prácticas compositivas. Entre los **aportes originales** se cuentan el desarrollo de los conceptos de material lógico y material concreto como dos estados del **material musical** temporal que afectan y definen la concepción y los recursos de manipulación compositiva, el nivel de representación base (explicado en el siguiente capítulo), la diferenciación entre relaciones abstractas y realización, el concepto de **material musical** como entidad abstracta recursiva y la adecuación de estos conceptos a los entornos informáticos y la práctica musical.

Debido a que muchos de los desarrollos que generan aportes significativos al conocimiento de las técnicas de representación informática musical son de carácter práctico o están relacionados con las técnicas de procesamiento de señales, se recurrirá al análisis de entornos existentes, los cuales puede o no aplicar una justificación teórica. Sobre estos aspectos tratan los **Capítulos 4** y **5**. El **Capítulo 4** desarrolla una clasificación **original** aplicable al análisis de los recursos informáticos como medio instrumental o representacional de la música. Se realizan además **aportes** al desarrollo de distintos enfoques que tienen a emplearse en la ingeniería de *software* para la creación de entornos. Como resultado de esto se acuña el concepto de nivel base, el cual es aplicable a la producción y el análisis musical, lo cual se demuestra en el **Capítulo 7**, y complementa la teoría expuesta en el **Capítulo 3**.

En el **Capítulo 5** se realiza el estudio de los recursos de representación existentes en los entornos de programación para síntesis de sonido y composición algorítmica/asistida, centrado en el análisis del entorno *SuperCollider* (McCartney 2002). No se pretende analizar la inmensa cantidad de entornos existentes puesto que en su mayoría redundan en conceptos fundantes, sino discernir y analizar estos conceptos fundantes y su aplicación específica. Cuando los conceptos difieren entre distintos entornos estos son considerados en comparación al entorno principal analizado. Para este estudio práctico se prefirieron entornos actualmente en uso puesto que esto permite recurrir tanto al *software* como al registro escrito de su desarrollo. En los casos en que fue necesario estudiar un entorno actualmente obsoleto se recurrió solamente al registro escrito. El objetivo de este análisis es derivar las estructuras que se pueden generalizar como primitivas de un modelo de representación general y estudiar las características del medio instrumental.

Una vez analizados todos los factores que intervienen en la producción sonora y musical, en el **Capítulo 6** se analizan las cualidades que se desprenden de este modelo expresadas en relación a la teoría desarrollada en los **Capítulos 4** y **5**. Luego se procede al desarrollo de las primitivas de representación informática correlacionadas con la conceptualización de los materiales musicales y sus formas de interacción. En conjunto, estos elementos conforman el **modelo de representación propuesto por esta Tesis**. Por último se deja constancia brevemente de los principios de una posible implementación del modelo desarrollado dentro del entorno *SuperCollider*. Como nota metodológica, cabe mencionar que tanto el modelo de representación como la teoría circundante fueron desarrolladas paralelamente a partir del análisis teórico y práctico de los recursos existentes.

Por último, en el **Capítulo 7** se procederá a la aplicación del modelo de representación como metodología de análisis de obras electroacústicas compuestas en entornos de programación para síntesis de sonido y composición algorítmica. En la primera parte de este capítulo explicita el modo de empleo y organización de los recursos teóricos mientras que en la segunda se procede a su aplicación.

Significación y Originalidad de la Tesis

Dentro del ámbito de la música electroacústica compuesta con entornos de programación para síntesis de sonido y composición algorítmica/asistida, existe una gran variedad de herramientas que se diversifican en técnicas específicas de la programación pero comparten estructuras de representación de los materiales musicales. Esto manifiesta el problema de la relación entre la ingeniería de software y la “ingeniería” de la composición, como dos disciplinas interrelacionadas puestas a un mismo nivel de jerarquía.

El código de programación actúa como partitura prescriptiva de los procesos compositivos. A nivel estructural, da cuenta de los procesos técnicos y mentales empleados para la composición. Es ventajoso desarrollar un método de análisis basado en un modelo de representación estándar, que pueda ser empleado para analizar las producciones musicales y presentar los resultados de manera independiente a los entornos informáticos que se usaron para generarlas. Esto permite, al público no especializado en todos los entornos de programación, un tratamiento más directo de los temas específicamente musicales.

El análisis propuesto por el enfoque de esta **Tesis** se posiciona en un nivel estructural que está entre la técnica de programación específica de cada entorno y las técnicas constructivas de materiales musicales. Este enfoque proporciona un punto

de vista más adecuado a la relación que existe entre las nociones compositivas y las técnicas instrumentales provistas por la informática.

Puesto que existen distintos planos de conocimiento (técnica musical, técnica compositiva abstracta, nivel perceptivo, etc), este modelo y método de análisis estructural sirve como nivel de referencia para analizar las estructuras compositivas/cognitivas de más alto nivel. Al estar compuesto de elementos musicalmente significativos, el sistema manifiesta la relación existente entre la técnica y la teoría empleadas para la creación de conocimiento y el desarrollo artístico.

Dada la generalidad del sistema de representación propuesto, es posible utilizarlo de manera analítica para el desarrollo de la composición en entornos de alto nivel (e.g. *SuperCollider*, *ChucK*, *Pure Data*) o bien implementarlo como entorno sí mismo, lo cual es un aporte a la ingeniería de estos sistemas.

CAPÍTULO 2: Estado del Arte

Teoría Sobre la Representación

Recientemente, a partir de la primera década del siglo XXI, se produjo un incremento notable en el empleo de lenguajes de programación para la producción artística debido al éxito de entornos como *Processing* (Reas y Fry 2007) en imagen, *Pure Data* (Puckette 1996) y *SuperCollider* (McCartney 2002) en sonido y *OpenFrameworks* (Perevalov 2013) en audiovisuales interactivas, entre muchos otros desarrollos previos y posteriores. Por una parte este fenómeno se produce gracias a la evolución de las técnicas informáticas, la capacidad de cómputo y la disponibilidad de los ordenadores personales, pero también se da como una valoración de los lenguajes de programación como herramientas más flexibles y completas a la hora de representar ideas musicales.

El desarrollo informático de las herramientas de representación musical también se relaciona con la evolución de las interfaces de interacción humano-máquina (HCI por *Human-Computer Interaction* en inglés). Desde el uso de tarjetas perforadas hasta las interfaces gráficas de alto nivel y los periféricos actuales, se fueron desarrollando conceptos en relación a la naturaleza computacional del recurso instrumental y la adecuación del conocimiento y las técnicas musicales preexistentes. El entendimiento sobre la materia musical se vio transformado, con la misma intensidad que en otras disciplinas del conocimiento, por la revolución informática. Es por esto que el desarrollo teórico de la representación musical, aplicado en los entornos de producción, está estrechamente ligado a la evolución de las herramientas técnicas y las teorías computacionales.

La interacción humano-máquina tiene que ver, entonces, con la representación del conocimiento de un dominio específico en relación a los recursos instrumentales disponibles. En el caso de la composición musical, la elaboración del conocimiento del dominio, aplicado informáticamente, depende de la interacción de varios factores integrados en un mismo medio. Los lenguajes de programación actúan a la vez como herramienta de control y sistema de representación de distintas concepciones musicales.

En este contexto, los recursos de representación gráfica, si bien adquieren cualidades únicas, derivan en gran medida de la representación de las estructuras informáticas a distintos niveles de abstracción. Es por eso que esta **Tesis** se enfoca principalmente en la representación estructural, como medio o método de conocimiento, de las entidades musicales representadas, y no tanto en los recursos visuales empleados para presentarlas.

Este capítulo se orienta a la exposición de los problemas teóricos y los recursos implementados (entornos y *software* específico) por separado. Puesto que, como se explicó en la metodología (**Capítulo 1**), muchos de los aportes a este campo del conocimiento son de carácter práctico.

Tratados a continuación, los aspectos teóricos considerados relevantes se centran principalmente en los estudios realizados por Dannenberg, Honing y Desain sobre los problemas de la representación informática musical. El enfoque de estos autores es preferido puesto que analizan aspectos de la teoría, la cognición y la técnica musical en relación a los recursos instrumentales en sintonía con la línea de investigación de esta **Tesis**. Los trabajos pioneros de estos autores resumen gran parte del estado de las cuestiones estudiadas en este trabajo. A continuación se mencionarán otros enfoques, como los recursos estándar y algunas concepciones específicas de distintas propuestas estéticas.

Complementariamente, se exponen individualmente los aportes prácticos realizados por distintos entornos –los cuales no necesariamente están desprovistos de teoría– como herramientas de representación y manipulación de materiales sonoros e ideas musicales.

Los problemas de la representación

De manera conjunta, Dannenberg, Desain y Honing (Dannenberg, Desain y Honing 1997), sumarizan una serie de situaciones, descubiertas a lo largo de sus estudios sobre la materia, que surgen al representar entidades musicales, las cuales clasifican como distintos problemas. Estos problemas son, parafraseando a sus autores, el estudio de casos prototípicos de los cuales se pretende deducir las propiedades que la representación musical debe exhibir (Dannenberg, Desain y Honing op. cit.).

Las soluciones que proponen estos autores a los problemas de la representación no será tratadas en detalle sino en relación a sus principios fundamentales. Como será desarrollado conceptualmente, en los próximos capítulos, como parte esta **Tesis**, los problemas de la representación son intrínsecos del sistema de conocimiento que los compositores de música electroacústica aplican. Puesto que muchas de las soluciones propuestas para estos problemas implican abstracciones de más alto nivel, no son empleadas de manera unívoca en la actualidad. En su lugar, tienden a ser preferibles las abstracciones de más bajo nivel para construir sobre estas la representación más adecuada. Esto se puede comprobar fácilmente analizando los entornos de programación para síntesis de sonido y composición algorítmica disponibles, los recursos que proveen y el nivel de representación de **base** que se emplea para la construcción de obras y recursos instrumentales (ver **Capítulo 4**).

Entidades continuas o discretas

El primer problema es la representación de **entidades continuas** o **discretas**. Para estos autores (Dannenberg, Desain y Honing op. cit.) los materiales musicales pueden ser representados de ambas maneras según el nivel a que se los estudie.

Una nota musical es un evento discreto determinado por sus propiedades y una señal continua es la representación de la variación de algún parámetro en función del tiempo. Si se considera la nota como unidad de control, esta se define por sus parámetros discretos a nivel de "*especificación*", pero si se considera el comportamiento dinámico interno se convierte en una señal continua a nivel de "*realización*". Si del nivel de realización se desciende al nivel de "*procesamiento de señales*", la nota se vuelve una sucesión de valores nuevamente discretos.

Esta ambigüedad se produce incluso a distintos niveles de abstracción de las entidades musicales, como posibles representaciones alternativas. Los autores dan el ejemplo de un trino, como entidad musical compuesta, que puede ser interpretada como una función que alterna la altura de manera continua o como una sucesión de objetos-nota.

Como se verá más adelante en esta **Tesis**, esta distinción de entidades discretas y continuas constituye los fundamentos de toda representación de estructuras musicales y su adecuada aplicación en el proceso de conceptualización en las prácticas compositivas o analíticas determina el grado de éxito de las mismas.

Volviendo al planteo de Dannenberg et. al., en relación a su carácter temporal, las entidades representadas por los entornos de programación pueden ser representadas "*en el tiempo*", "*fuera del tiempo*" o en "*tiempo real*". Esta clasificación, que los autores toman de Xenakis (1971), sirve para prever las capacidades de un sistema computacional. La representación "*en el tiempo*" significa que el procesamiento de las entidades musicales se producen en orden temporal. Cuando la capacidad de cómputo es lo suficientemente potente como para que no exista un retardo significativo en la respuesta del sistema al efectuar los cálculos se dice que este tipo de procesamiento es en "*tiempo real*". La representación "*fuera del tiempo*", por el contrario, refiere a que el procesamiento se produce sobre representaciones temporales pero no necesariamente en orden temporal (Dannenberg, Desain y Honing op. cit.).

La representación empleada por los entornos que actúan "*fuera del tiempo*" es menos restrictiva que los que actúan en el tiempo puesto que no tienen las **restricciones impuestas** por la causalidad necesaria en estos últimos. La representación "*fuera del tiempo*" es, si se analiza retrospectivamente, la que se emplea en los sistemas de notación tradicional y la teoría musical. Las entidades representadas mediante la notación, por

ejemplo, las melodías o las series de alturas, son estructuras que actúan fuera del tiempo y pueden ser modificadas de diversas maneras.

En los entornos de programación, la información discreta se puede representar de muchas maneras. Un programa se descompone en ordenes sucesivas que representan eventos discretos. Las sentencias de un algoritmo, las cuales pueden ser invocaciones a procedimientos a los cuales se les pasan parámetros, se realizan en relación a un determinado punto de inicio temporal. La ejecución de una acción en un momento determinado es un evento discreto pese a que pueda dar inicio a una acción continua (que se prolonga en el tiempo). La invocación de procedimientos como representación de acciones discretas es útil para la manipulación de programas "*en el tiempo*". Es el paradigma que emplean entornos de programación para síntesis de sonido en "*tiempo real*" como *SuperCollider* (McCartney 2002) o *ChucK* (Wang 2003).

En estos entornos existen los recursos de *co-rutinas*, en el primero, y *shreds* (*threads* sincrónicas) en el segundo. Un programa musical consiste en la ejecución de acciones discretas ordenadas temporalmente que se efectúan en un determinado momento. Este tipo de representación de acciones discretas también es empleado para la manipulación externa de procesos de síntesis que se están ejecutando en "*tiempo real*" empleando la separación entre el recurso instrumental (aplicación informática para síntesis de sonido) y las órdenes de control (acciones o algoritmos de control externos la aplicación de síntesis).

```
Routine {
    (midinote: 60, dur: 1).play; // evento
    1.wait; // tiempo de espera en la ejecución del programa
    (midinote: 62, dur: 1).play;
    1.wait;
    (midinote: 64, dur: 1).play;
}.play;
```

Código 1 Ejemplo de co-rutina en el entorno SuperCollider.

Un recurso de representación discreta también puede referirse a la programación de acciones "*fuera del tiempo*". Por ejemplo, la programación orientada a objetos es útil para este tipo de representación (e.g. Pope 1992), puesto que los objetos contienen información persistente sobre su estado. Esto posibilita que la manipulación de las estructuras musicales esté disociada de su ordenamiento temporal. Lo que se considera discreto en este caso son los objetos delimitados temporalmente, los cuales pueden ser notas, melodías, archivos de audio, etc, sobre los cuales se pueden realizar operaciones.

Teniendo en cuenta estas diferencias, y para adecuar la terminología de estos autores (Dannenberg, Desain y Honing 1997) al desarrollo de esta **Tesis**, el término *evento discreto*, se refiere a una acción realizada en un momento determinado "*en el*

tiempo", mientras qué, la denominación *objeto discreto*, se refiere a la representación de una entidad temporal "*fuera del tiempo*". En el **Capítulo 3** se verá cómo es posible pasar de la representación "*en el tiempo*" a la representación "*fuera del tiempo*" mediante la diferencia entre **material generador** y **material generado**.

La representación de información continua se emplea para la manipulación de señales. Estas pueden ser señales de audio o de control parámetrico de generadores o procesadores. Generalmente la información continua se representa en relación a la información discreta, ya sea contenida dentro de objetos discretos o controlada mediante eventos discretos. La representación de la información continua puede diferir cuando se emplean estructuras "*en el tiempo*" o "*fuera del tiempo*". En el primer caso se suele emplear la representación del instrumento o unidad generadora como elemento discreto que representa indirectamente la señal generada como componente de un sistema de flujo de datos. En el segundo, se relaciona con la noción de archivo de audio como **material concreto** (ver **Capítulo 3**) donde el archivo es la entidad discreta que contiene la información continua.

Los problemas del *vibrato* y el redoble

El problema del *vibrato*: Consiste en la relación arbitraria que se produce entre una función de control continua, que afecta la frecuencia del sonido, los límites del objeto sonoro, afectado como entidad discreta, y de cómo varían las propiedades de la función continua en relación al objeto discreto (Dannenberg, Desain y Honing op. cit.). Es un caso prototípico de la coexistencia de representaciones continuas y discretas, a distintas escalas de representación, que se manifiesta al realizar operaciones sobre un material compuesto.

Si el objeto compuesto se manipula desde la representación discreta (nota u objeto sonoro), surge el problema de cómo se debe comportar el *vibrato* al variar la duración de la nota. Si se entiende al *vibrato* como una entidad independiente que oscila a una frecuencia determinada, al cambiar la duración de la nota esta propiedad debería mantenerse y, por lo tanto, deberían agregare ciclos a la oscilación característica. En cambio, si se considera al *vibrato* como una oscilación que coincide con las características duracionales del objeto sonoro, como si fuera un *glisando* ascendente y descendente sincronizado con el principio y el final del objeto, el comportamiento adecuado sería incrementar el período de oscilación proporcionalmente con la duración del objeto.

Otra característica que afectaría el comportamiento del *vibrato* sería el comportamiento deseado en un contexto de relaciones, por ejemplo, si se pretende sincronizar el *vibrato* de una voz con otras, al realizar un corrimiento temporal del objeto

nota no debería alterarse la fase del *vibrato* en relación a las demás voces lo que produciría una alteración de la relación de la fase con el objeto sonoro al que afecta.

El problema del redoble: Es similar al problema del *vibrato* pero en lugar de tratar la relación entre una representación continua y una discreta, trata el caso de la relación entre dos representaciones discretas, el golpe de tambor y el redoble en su totalidad.

Si el redoble fue generado con un conjunto de "*samples"* sucesivos, e.g. grabaciones de un solo golpe de tambor, al querer realizar una transformación que afecte la duración del redoble en su totalidad, nuevamente se puede proceder de distintas maneras. Un procedimiento podría ser volver a muestrear la señal para cambiar la duración, lo cual afectaría también las altura espectral del sonido grabado. Otra forma de realización sería mantener constante la frecuencia de los golpes y agregar o quitar para cambiar la duración total. La convención musical define que la duración de un redoble se modifica mediante el segundo procedimiento en relación al **modo de acción** (Karkoschka 1972)5 y los recursos del instrumento acústico definidos culturalmente.

En el caso de la música por computadora, las abstracciones informáticas que representan el sonido resultante no implican el **modo de acción** tradicional sino uno distinto aplicado a los recursos instrumentales del medio informático. La extrapolación de estas convenciones es lo que estos autores determinan como el "*conocimiento musical que deben expresar las abstracciones informáticas*" (Dannenberg, Desain y Honing op. cit.).

Para el planteo de esta **Tesis**, si se analiza el problema desde esta perspectiva, se trata de emular el comportamiento instrumental de un ámbito específico dentro de otro, lo cual genera los problemas estudiados. Sin embargo, en defensa de estos autores, puede afirmarse que son problemas válidos, puesto que manifiestan el conocimiento musical como la diversidad de recursos y representaciones elegibles que implican una decisión técnica que afecta el producto final.

La representación informática resulta mucho más versátil puesto que permite imitar comportamientos existentes o generar nuevos. Es por esto que en el desarrollo de esta **Tesis** se analizan los recursos instrumentales de la computadora como los principios que definen las entidades representables. También es importante notar que la mayoría de los problemas enumerados por estos autores se manifiestan al aplicarle una acción de transformación o comportamiento temporal a las entidades objetivadas. El carácter temporal dinámico de las abstracciones y acciones compositivas es lo que genera cambio y diversidad en múltiples planos conceptuales.

Abstracciones de comportamientos: Un comportamiento es un proceso específico que se desarrolla en función del tiempo, la abstracción de ese comportamiento consiste en su generalización como entidad independiente. Este es un concepto

desarrollado por Dannenberg dentro del entorno de programación *Nyquist* (Dannenberg 1997). Mediante las abstracciones de comportamientos se pretende mantener una "*correspondencia estricta*" entre la representación del sonido y su realización o "*realidad sonora*".

Las abstracciones de comportamientos se denomina también, en base a sus cualidades, como "*polimorfismo contextual*" (Dannenberg, Desain y Honing op. cit.). El polimorfismo implica que un comportamiento abstracto, al ser sometido a una transformación temporal, depende de la entidad a la cual se aplica, por ejemplo, un *vibrato* se comporta de manera distinta que un *glissando* al ser expandidos temporalmente. Estos comportamientos polimóficos, al ser aplicados a estructuras complejas, también dependen del contexto. Por ejemplo, el comportamiento sincrónico del *vibrato*, mencionado anteriormente, que afecta a un conjunto de notas superpuestas, está afectando el contexto a mayor nivel de agrupamiento que si afectara a una sola nota y, por lo tanto, actúa de manera distinta.

En el lenguaje *Nyquist* (Dannenberg 1997), las abstracciones de comportamiento dependen de variables de entorno de alcance dinámico6, que definen el comportamiento por defecto de distintos parámetros al producirse una transformación sobre una entidad sonora. Al anidar las estructuras de representación sonora, el alcance dinámico de las variables hace que los valores del entorno contenedor se propaguen a los sub-entornos contenidos. Los parámetros por defecto pueden ser cambiados mediante argumentos explícitos a cualquier nivel estructural y mantener así la coherencia con los niveles inferiores. En otras palabras, esta técnica hace que las estructuras musicales, representadas de manera jerárquica, tengan garantizado un comportamiento coherente con respecto a un determinado parámetro musical. Las estructuras internas de un **material musical** compuesto "*heredan*" comportamientos paramétricos específicos definidos por la macro-estructura que las engloba.

Las abstracciones de comportamientos son una solución propuesta a los problemas del *vibrato* y el redoble. Actúan definiendo la manera en que los parámetros deben comportarse al aplicarse determinadas transformaciones. Son abstracciones que entienden a las entidades sonoras como objetos temporalmente discretos, análogos al concepto de nota musical, separados de las transformaciones, como entidades dinámicas cuya delimitación temporal depende del objeto al que se apliquen.

Conceptualmente, una transformación implica un objeto dado y una acción a efectuar sobre ese objeto. La acción efectuada tiene consecuencias en el desarrollo temporal del objeto pero no representa ese desarrollo temporal. Como se verá en el **Capítulo 3**, la noción de **proceso**, opuesta a la de **objeto**, define otro tipo de representación fundamental del **material sonoro** que se complementa con este último en la generación de entidades

sonoras. Desde este punto de vista, una abstracción de comportamiento es, en realidad, la representación de una manipulación externa al **material musical** que actúa fuera del tiempo de la misma, pero es un recurso de gran utilidad para entender las operaciones que se pueden realizar de manera coherente dentro de un determinado sistema de representación.

Los problemas de la anticipación y la transición

El problema de la anticipación surge de los recursos de ornamentación en la música instrumental y, de manera general, de las acciones interpretativas que definen o modifican la estructura musical representada mediante la notación tradicional. La anticipación implica la previsión de un cambio en algún parámetro interpretativo como puede ser la duración de las notas en relación a los ornamentos. Según se ejecute la ornamentación (antes o después del comienzo de la nota principal) es necesario prever la alteración de las duraciones circundantes. Lo mismo sucede con las articulaciones en relación al fraseo musical.

El problema de la transición está relacionado con el problema de la anticipación. Por ejemplo, para realizar un *portamento* es necesario conocer de antemano la duración disponible y la nota de destino para poder efectuar el proceso de manera adecuada. El *portamento* es una acción objetivada que depende del contexto actual y posterior para su correcta realización y, por lo tanto, requiere también anticipación. Nuevamente los autores (Dannenberg, Desain y Honing op. cit.) se encuentran con el problema del **modo de acción**, pero esta vez lo expresan, de manera más directa, en relación a la objetivación de inflexiones expresivas.

De manera similar que en el problema de la continuidad expresado por Wishart (1996), la cual, según este autor, no es posible de representar mediante la notación analítica tradicional, se recurre a la representación de entidades que discretizan el continuo temporal para poder transmitirlo conceptualmente. El conocimiento musical que se manifiesta en el concepto de *portamento* es el de la transición de un parámetro, objetivada como elemento finito. Dentro de la infinidad de variaciones posibles, el *portamento* adquiere identidad como elemento discernible puesto que implica un mayor grado de cambio, en relación a un contexto de alturas más estáticas, que genera una cualidad que se percibe de manera discriminada.

Discretizar y objetivar una transición es algo que se hace perceptualmente de manera natural, sin embargo, para representarla mediante recursos informáticos es necesario prever el comportamiento de las estructuras involucradas. Si se adopta una representación basada en los **recursos instrumentales** (ver más adelante), se pueden

programar estos recursos de manera tal que posibiliten la transición entre notas. En el ejemplo Código 2 se emplea la interpolación (implementada mediante un filtro dentro de la llamada al método *lag*) del parámetro *freq* entre dos notas sucesivas, retrasando el inicio de la segunda 0.2 segundos.

```
x = { arg freq = 67.midifreq; SinOsc.ar(freq.lag(0.2)) };
x.set(\freq, 71.midifreq);
x.set(\freq, 72.midifreq);
```

*Código 2 Ejemplo de recurso instrumental, método **lag** sobre el parámetro **freq**.*

En cambio, si se adoptara una representación "*fuera del tiempo*", basada en objetos discretos que no tienen esta capacidad programada, la transición se podría realizar descomponiendo y modificando sus parámetros mediante envolventes de transición definiendo sus tiempos de inicio y final. Para esto sería necesario analizar las estructuras de datos involucradas y seleccionar los parámetros y tipos de transición.

El problema del compresor

Este problema manifiesta la interdependencia entre el todo y las partes una vez que los elementos sonoros están organizados en estructuras más complejas. El ejemplo prototípico es el de la sonoridad resultante de un ensamble instrumental, compuesta por la sonoridad individual de cada instrumento. Ajustar la sonoridad total implica ajustar la sonoridad individual de cada instrumento de manera relacionada, lo cual requiere que el flujo de datos recorra la estructura de ida y vuelta entre el todo y las partes para su procesamiento.

Esto resulta en un comportamiento similar al de un compresor de señales de audio que realza las partes de baja amplitud y limita las de mayor nivel reduciendo el rango dinámico de la señal. La principal diferencia consiste en que el compresor actúa sobre la señal como entidad monódica mientras que el ensamble consiste en el comportamiento interrelacionado de varias señales. El problema del compresor manifiesta la diferencia que existe entre los niveles de abstracción empleados para la manipulación de los recursos y las estructuras sonoras y cómo estos afectan las posibles relaciones, así como el problema del control sobre materiales compuestos.

La amplitud resultante de la suma de estructuras complejas depende de la realización, es decir, no se puede saber hasta que el sonido real es generado. La resultante depende del tiempo de inicio, la envolvente dinámica y la amplitud de cada una de las partes individuales. Esto es un ejemplo de la diferencia cualitativa entre **material generador** y **material generado** (ver **Capítulo 3**). Dependiendo de la naturaleza de las

estructuras de datos empleadas puede no ser posible extraer los valores de amplitud de las entidades componentes a cada instante puesto que pueden no haber sido calculadas de antemano.

Para estos autores (Dannenberg, Desain y Honing op. cit.) la diferencia fundamental reside entre la representación causal y no causal del tiempo. Mediante la representación no causal, es posible analizar las estructuras generadas y extraer los datos necesarios para la manipulación de los componentes individuales en relación a la resultante general. El valor en relación a la duración que adquiere cada parámetro puede ser contemplado y manipulado mediante objetos concretos.

El problema de las descripciones estructurales

Honing (1993) describe los tipos de relaciones estructurales entre objetos basándose en relaciones binarias o n-narias. Describe el tipo de relaciones "*es parte de*" para representar las estructuras jerárquicas, especialmente las estructuras de tipo árbol. Considera que esta es un tipo de estructura que se adecúa a la descripción estructural de la música puesto que posibilita la descomposición de los objetos complejos y el agrupamiento de estructuras que se comportan de manera homogénea a distintos niveles. Por ejemplo, las estructuras de frases y sub-frases o las estructuras de tiempo y metro.

Sin embargo, Honing advierte que este tipo de estructuras pueden resultar ambiguas para otros tipos de relaciones, especialmente las que se derivan de distintos análisis de una misma pieza musical o en los casos en que las estructuras internas no están delimitadas de manera concordante. En el primer caso (estructuras de frases y sub-frases) se refiere a la interpretación de estructuras a nivel formal e incluso semántico, lo cual podría estar relacionado con la percepción del analista de los fenómenos acústicos expresados en la obra. En el segundo caso (estructuras de tiempo y metro) se refiere a los procesos de imbricación o falta de coincidencia entre niveles estructurales. Por ejemplo, cuando el final de una frase musical actúa como el principio de la siguiente de manera superpuesta (se puede entender que incluso compartiendo las mismas notas a la manera de una elisión), o cuando las estructuras de frases musicales no coinciden con la estructura métrica.

Por otra parte, el autor expresa las relaciones que implican distintos pasajes musicales según sean de mayor o menor importancia para la composición (esenciales u ornamentales) y las relaciones asociativas que existen entre un tema y sus variaciones. El nivel de estructura musical al que se refiere con estos últimos ejemplos de ambigüedad parece estar en un plano más subjetivo relacionado con las concepciones estructurales que se manifiestan en la composición musical y las relaciones intelectuales que se pueden

trazar dentro de un discurso musical.

Para esta **Tesis**, es importante distinguir entre las relaciones estructurales objetivas y subjetivas/perceptivas/intelectuales, las primeras posibilitan la construcción técnica y la definición analítica de los elementos de que constituyen una composición musical, mientras que las segundas actúan a otros niveles de representación que pueden ser distintos a la estructura objetiva (véase el desarrollo de estructura objetiva y estructura subjetiva más adelante en el **Capítulo 3**).

Otro tipo de relación considerado por Honing es la relación "*es un*", usualmente denominada de *herencia* en la programación orientada a objetos. Este tipo de relación jerárquica es ortogonal (está en otra dimensión) con respecto a la representación en árbol y sirve para determinar la relación generalidad-especificidad. Por ejemplo, entre la estructura de acorde, como la superposición de notas y un acorde específico, como puede ser un acorde de séptima mayor.

Para el desarrollo de esta **Tesis** es importante notar que este tipo de relación no necesariamente existe en el tiempo, sino que resulta útil para la clasificación y organización estructural coherente de determinadas relaciones que luego deben ser implementadas o **realizadas** paramétricamente en el tiempo. Es decir, este tipo de relación implica una organización estructural abstracta de un parámetro, como puede ser la altura, en relación al repertorio de combinaciones posibles. Esto es similar a la definición de un vocabulario que luego debe ser puesto en forma para generar el discurso (véase el desarrollo de los conceptos de **relaciones abstractas** y **realización** más adelante en el **Capítulo 3**).

Con respecto a las relaciones n-narias, Honing da el ejemplo de la relación entre un acorde y la escala, el modo y el contexto sobre los cuales está construido. Aunque no da mayores explicaciones, se puede entender que se refiere a los distintos tipos de relaciones que se pueden trazar según el enfoque y la inteligencia del análisis realizado sobre el discurso musical.

Con respecto a los lenguajes de programación, en el análisis realizado por Dannenberg, Desain y Honing (1997), se discierne distintos tipos de representación: no estructurada (estructura plana), estructura de dos niveles, estructuras jerárquicas recursivas (en forma de árbol) y estructuras heterárquicas.

Un ejemplo de estructuración de dos niveles son los lenguajes de la familia *Music N* (Mathews 1969) (Roads 1995). Estos lenguajes dividen el código de programación entre dos estructuras planas de representación. Por un lado está la partitura, que consiste en una lista de eventos parametrizados y, por otro, las definiciones de instrumento en el lenguaje de la orquesta que representan los algoritmos de síntesis. Si bien cada definición de instrumento implica la representación de un objeto instrumental que consiste en la

interconexión de unidades generadoras, las cuales podrían ser consideradas como sub-nivel, los autores se refieren a las definiciones de instrumento como elementos atómico a un nivel determinado de jerarquía.

Es necesario mencionar también que en *Csound* (Vercoe 1986) es posible invocar un instrumento previamente definido desde otro instrumento o compartir información entre instrumentos mediante variables globales lo cual posibilita cierta organización jerárquica de las definiciones instrumentales aunque el lenguaje no esté enfocado a facilitar estas capacidades. En este entorno, cuando son necesarias estructuras organizativas más complejas, se suele recurrir o bien a *macros* o bien a lenguajes de programación externos de más alto nivel que posibilitan la organización en estructuras tanto de los **recursos instrumentales** como de la organización de los materiales musicales (ver más adelante en este capítulo). Estos recursos sirven para reutilizar instancias de comportamientos predefinidos y sus invocaciones pueden ser anidadas generando relaciones jerárquicas (como estructuras discretas anidadas).

Los lenguajes de programación de propósito general comúnmente soportan la recursividad tanto de estructuras de datos como de funciones. Una estructura de datos recursiva es una estructura dinámica que se define en términos de si misma. Una lista ligada o un árbol son ejemplos canónicos de este recurso. La estructura de datos que conforma la información resultante contiene tanto los datos de si misma (elemento de una lista o nodo de un árbol) como una referencia a el o los siguientes elementos de la estructura resultante (los cuales son, obviamente, del mismo tipo). Para su realización es necesario definir solo una estructura de datos y, básicamente, los procedimientos de inserción, ordenamiento, búsqueda, recuperación y borrado. Las estructuras de datos recursivas son extremadamente versátiles para representar estructuras complejas basadas en principios estructurales relativamente simples y pueden ser de tamaño teóricamente infinito.

La recursión también se refiere a la capacidad de una función de invocarse a sí misma, es un recurso que simplifica notablemente la lógica de operaciones complejas como la búsqueda de datos en estructuras de tipo árbol o, de manera más general, grafos. En relación a la representación musical, estos tipos de estructuras son muy importantes puesto que posibilitan la definición de las primitivas de representación y distintos tipos de relaciones jerárquicas variables que se organizan mediante un conjunto acotado de operaciones básicas (ver **Capítulo 3**).

Para ciertos problemas de la organización musical, por ejemplo en el problema del solapamiento de frases musicales, puede ser necesario recurrir a estructuras no jerárquicas que puedan expresar la relación de composición ("*parte de*") pero sin restringir las relaciones posibles entre distintas unidades estructurales. El problema que destacan

estos autores (Dannenberg, Honing y Desain op. cit) es que, *hacer suposiciones sobre cómo se deben representar los materiales musicales a bajos niveles de abstracción, puede motivar que los entornos resulten totalmente inútiles toda vez que una suposición no es correcta*7. Esta afirmación es significativa puesto que tanto las concepciones compositivas como los conocimientos científicos sobre cognición musical pueden ser dinámicos según el grado de desarrollo en diferentes épocas.

Como ejemplo de relaciones heterárquicas, Dannenberg, Desain y Honing (1997) exponen el problema de la ornamentación. Una ornamentación puede depender de distintos contextos al mismo tiempo, por ejemplo, del *tempo* y la duración. Consideran que la ornamentación se puede dividir en dos categorías, las que alteran la duración de las notas circundantes ("*time-taking*") y las que no ("*timeless*"). Al efectuarse transformaciones de *tempo*, los ornamentos no modifican su duración proporcionalmente sino en relación a sus cualidades como tipo de ornamento y, por lo tanto, dependen de dos niveles de organización distintos, uno en relación a la jerarquía melódica y el *tempo*, y otro en relación a las clases de ornamentaciones. Un ejemplo concreto de esta problemática, abordada como concepción compositiva, es la klavierstuck VII de Stockhausen (1965) (Di Liscia 2013) donde existen dos organizaciones temporales (*acciacaturas* y notas) combinadas.

El problema instrumental

Este problema expresa de manera clara las diferentes concepciones que un compositor puede tener sobre los materiales musicales, o bien como entidades abstractas, o bien como el resultado de los recursos instrumentales. Según estos autores (Dannenberg, Honing y Desain op. cit.), los compositores pueden pensar en las notas como sonidos independientes o como las instrucciones a un solo intérprete. Dan el ejemplo de la textura puntillista, donde cada nota es distinta y no importa si un conjunto de notas provienen de un solo instrumento, en contraposición a una frase *legato* que no puede ser ejecutada con distintos instrumentos por nota. En el primer caso la entidad fundamental es la nota mientras que en el segundo es el instrumento.

Tradicionalmente, varios autores refieren a este problema desde distintas perspectivas, Karkoschka (1972) se refiere a la "*notación de acciones*" en la música instrumental diferenciando la finalidad de la notación musical que representa instrucciones al intérprete de otras que tienden a representar el sonido resultante. De manera similar, Windsor (1995) expone, en relación al análisis musical y el nivel neutro de Nattiez, que la notación musical sirve un propósito dual, descriptivo de la música como fenómeno sonoro y prescriptivo de las acciones del intérprete, pero descarta que este tipo de relación sea

aplicable a la música acusmática puesto que no existe la mediación interpretativa y la representación no solo no es estándar sino que incluso puede estar ausente en lo absoluto.

Sin embargo, la descripción del problema instrumental parece ser un caso válido, puesto que el **material lógico** (código de programación o aplicación informática, ver **Capítulo 3**), al ser accesible e interpretable por el mismo compositor u otras personas, manifiesta de manera objetiva las concepciones compositivas adoptadas y los recursos representacionales empleados en la manipulación informática. En palabras de sus autores (Dannenberg, Honing y Desain op. cit.), *"el problema instrumental refiere al desafío de modelar recursos de producción sonora (instrumentos) y su relación con las instancias sonoras (notas)"*. Más aún, es un problema que no solo afecta la relación entre notas e instrumentos sino que también actúa internamente en la definición estructural de los instrumentos, puesto que puede generar características sistémicas que afectan la producción de los materiales sonoros.

Como otro ejemplo que concierne a este asunto de representación, considérese una definición de instrumento que genere materiales complejos. Al ser intervenido externamente mediante eventos, el instrumento como abstracción, no solo puede generar notas simples sino que también puede estar diseñado para generar materiales complejos como patrones o texturas de largo aliento, las cuales definen sus posibilidades de manipulación. La representación adoptada en relación a las **abstracciones instrumentales** dan cuenta de ciertos aspectos de la concepción compositiva y los recursos técnicos que pueden generar resultados específicos8.

La relación entre la estructura musical abstracta y la estructura instrumental es explicada mediante el modelo *recurso-instancia* (Dannenberg, Honing y Desain op. cit.) (Dannenberg, Rubine, y Neuendorffer 1991). Empleando la terminología de la programación orientada a objetos, una instancia es un objeto específico que representa una abstracción musical definida como clase. Esta puede ser un objeto, una llamada a una función, una estructura de datos, una envolvente almacenada en un arreglo, etc, que se emplea para representar la estructura instrumental o sonora *in situ*. El ejemplo que dan los autores es el de los lenguajes *Music N* (Mathews 1969) (Roads 1995) donde cada nota de la partitura es una instancia (copia) de los procesamientos definidos por los instrumentos en el lenguaje orquestal. Las instancias son objetos concretos, realizados individualmente, se manipulan como información discreta. En cambio, un recurso es una **abstracción instrumental** (ver **Capítulo 4**) que recibe la información discreta (los eventos de control), la cual es empleada para modificar el comportamiento o estado del recurso. Un recurso puede ser un canal de audio, un *buffer*, un sintetizador, etc. Es una abstracción que no implica una realización sonora concreta sino que determina sus posibilidades.

Algo que se confunde en la definición proporcionada por en el modelo *recurso-instancia* es que una instancia puede estar representando tanto a un **material musical** concreto como a un recurso instrumental y, por lo tanto, no considera el factor temporal. Los autores dan el ejemplo de una guitarra, la cual se puede instanciar como modelo instrumental. Al definir una guitarra como instancia se define un objeto atemporal, simplemente como la abstracción de un objeto material. Internamente, la instancia guitarra puede generar sub-instancias de objetos cuerda. Las cuerdas actualizan su estado mediante información discreta (órdenes de control), la cual determina la longitud de la cuerda, la posición de toque y la pulsación de la misma. El modelo se considera poderoso puesto que provee una representación natural para solucionar el problema instrumental. El modelo también es flexible puesto que *"el compositor puede decidir instanciar una guitarra por cada nota, emplear la misma guitarra pero instanciar una cuerda por cada nota musical o ejecutar todas las notas en una sola cuerda"*, lo cual genera diferentes variantes sonoras y restringe las posibilidades al mismo tiempo. Sin embargo, este modelo no representa las abstracciones sonoras basadas en **síntesis abstracta**, las cuales también emplean el comportamiento *recurso-instancia* para generar distintos tipos de realizaciones musicales mediante abstracciones no convencionales que pueden estar representando tanto el recurso instrumental como la estructura sonora según sea su modo de empleo.

El modelo recurso-instancia es estudiado en esta tesis en relación a la **interfaz instrumental** e **interfaz de composición** respectivamente (ver **Capítulo 4**). Aunque los detalles de la implementación informática no sean relevantes, es una distinción que sirve para entender las abstracciones que representan los distintos tipos de entidades instrumentales y materiales musicales que interactúan en el *universo simbólico9* del compositor de música electroacústica.

Otros enfoques

Recursos estándar

Gran parte de la teoría sobre el empleo de lenguajes de programación para la representación musical se centra la definición de los elementos del sistema de representación como estructuras de datos relacionadas con entidades musicales tradicionales y las abstracciones de sus propiedades fundamentales (e.g. nota, altura, amplitud, frase o secuencia, acorde, etc). Ejemplos de estos lenguajes son: *Nyquist* (Dannenberg 1997), *Common Music* (Taube 1991), *Kyma* (Scaletti 1989), *PatchWork y OpenMusic* (Assayag et al. 1999) (Agon 1998), *SuperCollider* (McCartney 2002) y *Elody*

(Letz et al. 1998) entre muchos otros. Los recursos de representación relacionados con la tradición musical son empleados de manera sistemática por la totalidad de entornos que emplean lenguajes de programación de alto nivel por razones obvias. Incluso aunque no estén centrados en la representación tradicional como finalidad específica, los entornos de programación de alto nivel recurren a los elementos que representa las magnitudes conocidas.

Como ejemplo paradigmático, dentro de la programación orientada a objetos, se puede mencionar el lenguaje de descripción musical *Smoke* (acrónimo en inglés de *Smalltalk music object kernel*) (Pope 1992). Este lenguaje consiste en la especificación de los tipos de datos básico del lenguaje de programación (e.g. enteros, decimales, fracciones, *strings*, etc), las estructuras de datos que representan entidades musicales (e.g. eventos, notas, frases, acordes, etc) y las operaciones posibles sobre estas estructuras de datos empleadas para relacionarlas (e.g. relaciones sucesivas o simultáneas de notas, distintos tipos de operaciones de transformación, etc.). Muchos de estos elementos serán visto de manera detallada en el **Capítulo 5** sobre el entorno de programación *SuperCollider* (McCartney 2002).

Al emplear la programación orientada a objetos, o cualquier otro paradigma de programación de alto nivel, es posible representar de manera más específica y concisa los elementos tenidos en cuenta de manera general. Los recursos de representación suelen estar supeditados a la lógica del lenguaje y la técnica de programación. La especificidad se logra mediante la definición de las estructuras de datos que se consideran como primitivas de la representación musical, las cuales restringen las relaciones posibles según su grado de especificidad.

El desarrollo teórico de estos recursos, si bien es importante, está lo suficientemente difundido dentro de las técnicas de programación actuales. Es por eso que en esta **Tesis** no serán analizados de manera particular salvo en relación a los recursos informáticos necesarios para elaborar el **modelo** de representación musical. Puesto que se considera mucho más importante discernir las concepciones que llevan al desarrollo de la representación musical informática (explícitas en los problemas de la representación ya expuestos) antes que el repertorio de técnicas actualmente estándar de la ingeniería de *software*.

Concepciones específicas

En la literatura sobre la representación musical empleando medios informáticos es habitual la exposición de enfoques particulares, derivados de, o aplicados a, composiciones específicas. Esto se debe a que el empleo de lenguajes de programación

de propósito general posibilita el desarrollo de innumerables concepciones compositivas.

Algunos ejemplos interesantes se pueden encontrar en libros como *The OM Composer's Book* (Agon et. al. ed. 2006), donde distintos autores, por capítulo, expresan sus concepciones compositivas, su metodología de trabajo y las representaciones que desarrollan sobre los materiales musicales, algunos de manera más explícita que otros.

Otro tipo de aportes se dan en relación a la notación tradicional, extendida o adaptada para representar música electroacústica. Por ejemplo, el Estudio II de Stockhausen (1956) emplea un sistema de representación que define la estructura de la realización electrónica de la pieza.

Si bien estos y muchos otros aportes similares son valiosos, los elementos que se abarcan en esta **Tesis** están enfocados desde la perspectiva de la programación como medio de representación en sí mismo, y pretenden ser generalizables. Es por esto que se trata de no adoptar conceptos de representaciones que puedan resultar específicos de producciones estéticas particulares.

Representación en Entornos Existentes

En esta sección se enumeran los enfoques prácticos que proponen distintos entornos de programación para síntesis de sonido y composición algorítmica/asistida. Debido a que, por su extensión, sería imposible mencionar todos los desarrollos presentes y pasados, fueron seleccionados los que se consideraron más relevantes, ya sea por su uso extendido o por sus aportes originales.

Lenguajes de programación

Paradigmas iniciales

Los entornos de programación para síntesis de sonido y composición algorítmica definen estructuras de datos y algoritmos que se emplean para representar objetos y comportamientos musicales. Aunque en la actualidad los entornos más utilizados unifican los recursos de síntesis y procesamiento con los mecanismos de control, históricamente estos aspectos fueron desarrollados por separado debido, en parte, a las **restricciones impuestas** por la capacidad de procesamiento. Sin embargo, esta división histórica hace explícita la separación conceptual entre instrumentos y órdenes de control.

En la categoría de lenguajes creados para síntesis de sonido se encuentran los desarrollados originalmente por Max Mathews (Mathews 1969) comúnmente denominados como *Music N* (Roads 1995). La letra "*N*" representa las distintas versiones de la

aplicación informática a medida que cambiaba su implementación para agregar nuevas características o para poder ejecutarlo en distintas computadoras debido a los recursos de la época en que fueron creados. De esta familia de lenguajes surgen posteriormente otros como *Csound* (Vercoe 1986) (Boulanger 2000), *CMix* y *Cmusic* (Moore 1990), escritos en un lenguaje de programación más moderno y portable. En la actualidad es *Csound* el que se emplea de manera más extendida y es el que se tomará como referencia puesto que incluye todas las características desarrolladas por esta familia e incorpora el recurso de señal de control como técnica para optimizar la capacidad de cómputo al emplear osciladores de baja frecuencia.

Estos lenguajes de programación dividen los algoritmos de procesamiento y control musical empleando el modelo de orquesta-partitura. La orquesta es la parte del programa que se encarga de especificar los algoritmos de síntesis interconectando las salidas y entradas de *unidades generadoras* (algoritmos modulares de síntesis) como señales de audio. En la orquesta es donde se crean los instrumentos, también llamados *definiciones de síntesis*, que son los encargados de producir sonido. La partitura especifica secuencialmente, según el tiempo absoluto de inicio de los eventos, el conjunto de órdenes y parámetros empleados para ejecutar, en tiempo real o diferido, los instrumentos definidos en la orquesta. Sin embargo, la orquesta, en esta familia de lenguajes, carece de la flexibilidad que posibilitan los lenguajes de programación de propósito general como recurso de control y expresión de las estructuras informáticas que representan materiales y estructuras musicales.

Lenguajes de control

Con respecto al control musical, existen lenguajes y entornos diseñados específicamente para generar o enviar órdenes de control que interactúan con otras aplicaciones de síntesis y procesamiento o sintetizadores por *hardware*. Dentro de esta categoría se encuentran lenguajes como *Formes* (Cointe y Rodet 1984a y 1984b), *Formula* (Anderson y Kuivila 1991) y entornos como *Blue* (YI 2001) y *AthenaCL* (Ariza 2005). Estos lenguajes y entornos actúan como secuenciadores, de eventos o de algoritmos de síntesis, y definen estructuras de más alto nivel que representan entidades musicales. Estas abstracciones generalmente representan estructuras simples como notas, voces, secuencias, acordes o comportamientos dinámicos, pero pueden incluir también estructuras más complejas como tipos de texturas particulares generadas algorítmicamente. En el caso del lenguaje *Formula*, se definen estructuras materiales como procesos concurrentes de control paramétrico que pueden ser agrupadas generando abstracciones de mayor complejidad, esto es particularmente interesante para esta **Tesis**

puesto que demuestra la importancia de las estructuras jerárquicas en relación a la organización temporal en la composición musical. Como se verá a continuación, muchas de estas características son adoptadas también por los lenguajes de programación para síntesis de sonido y composición algorítmica.

Lenguajes integrados de propósito general

Junto con el incremento en la capacidad de cómputo y el desarrollo de las técnicas informáticas, fueron surgiendo lenguajes de programación que integran la lógica de síntesis y control de sonido dentro de un mismo ámbito de representación. Las estructuras de datos y de control algorítmicas se unifican, en mayor o menor medida, a un mismo nivel estructural. Esta nueva organización posibilita un mayor grado de interacción entre las estructuras que representan señales sonoras y las que representa organizaciones musicales. Se integran los procedimientos necesarios para la elaboración musical a distintos niveles estructurales.

De la gran variedad de lenguajes que se fueron desarrollando sobre este nuevo paradigma se encuentran entornos como *SuperCollider* (McCartney 2002), *Nyquist* (Dannenberg 1997) y *ChucK* (Wang 2008), por mencionar solo algunos lo suficientemente representativos de los aspectos más relevantes.

Algunos entornos se construyeron sobre la base de lenguajes de programación preexistentes, de los cuales *Lisp* tal vez sea el lenguaje más utilizado, este es el caso de *Nyquist* entre los ejemplos citados. Otros, en cambio, desarrollaron nuevos lenguajes de programación más orientados al dominio específico, pensados para adecuar la expresividad del lenguaje a las estructuras de datos y técnicas de programación empeladas para la síntesis y la composición algorítmica. Estos entornos se pueden diferenciar por el paradigma que define la arquitectura básica.

SuperCollider, en su versión 3, está pensado modularmente en base a la separación de la aplicación informática que realiza la síntesis de sonido y el lenguaje de programación. El servidor es el programa que se encarga de generar sonido en base a la organización de *unidades generadoras*, *definiciones de síntesis*, *buses* de audio y de control y manejo de archivos de audio, pensado como una aplicación de tiempo real. El lenguaje de programación, cliente del servidor de síntesis, es el que envía las instrucciones de cómo organizar los recursos de la aplicación de síntesis.

Aunque hay una división arquitectónica clara entre la aplicación de síntesis y la aplicación de control, es esta última la que se encarga de representar y manipular todos los recursos posibilitando la interacción entre ambos niveles de abstracción. La división cliente-servidor es una decisión proveniente de la ingeniería de *software*, en la versión

anterior de este entorno (McCartney 1996) no había existía esta división y tanto los algoritmos de síntesis como los de control dinámico actuaban dentro de la misma aplicación informática.

Nyquist es otro lenguaje que integra las estructuras de control de síntesis con la lógica de control de los instrumentos mediante la definición de funciones y sus interrelaciones. De manera similar a *SuperCollider*, los elementos que representan procesos de síntesis y de control instrumental están definidos a un mismo nivel sintáctico diferenciándose por el tipo de abstracción que representan las funciones del lenguaje de programación. Un rasgo distintivo de este entorno es que define lo que denomina *abstracciones de comportamientos* (explicado anteriormente en los desarrollos teóricos), que son funciones especiales que especifican cómo deben comportarse distintos tipos de abstracciones musicales según las cualidades definidas por el usuario. Si bien este es un recurso que se puede programar en cualquier de los entornos mencionados, se destaca por la relevancia que se le otorga dentro de este entorno como un tipo de abstracción fundamental en relación a las cualidades básicas de los materiales musicales.

El lenguaje de programación *ChucK* continúa empleando los mismos paradigmas mencionados hasta el momento. Las principales diferencias son que incorpora, a nivel sintáctico, las nociones de programación concurrente temporal (Wang y Cook 2003, 2004a, 2005) (Wang 2008), como mecanismo de control temporal, y el flujo de información sonora, que son las interconexiones entre *unidades generadoras*. El modelo de concurrencia temporal hace que no sea necesaria la distinción entre frecuencias de audio y frecuencias de control, introducidas por *Csound*, para representar distintos tipos de señales según su resolución temporal. Esto unifica aún más las diferencias entre algoritmos para producción de sonido y control paramétrico puesto que el período de control temporal variable integra las estructuras de control de flujo del programa desde la lógica de la generación de sonido hasta la manipulación de sus parámetros.

Las posibilidades de control sobre la síntesis de sonido y el control paramétrico es lo que define al recurso como medio de representación de los elementos musicales de manera similar a cómo actúan la notación musical y los recursos instrumentales convencionales. Una pieza musical escrita en su totalidad con un lenguaje de programación, representa, por medio de relaciones entre diferentes abstracciones informáticas, los materiales y las estructuras musicales y su realización como fenómeno sonoro. Es en este sentido que los lenguajes de programación pueden ser entendidos como una forma de notación musical extendida (Dannenberg 1996), puesto que además de representar el contenido estático de una composición musical de manera equivalente a la notación tradicional, pueden también contener recursos

dinámicos que varían de ejecución en ejecución o en relación a la interacción del intérprete en el caso de la música mixta.

Los lenguajes de programación para síntesis de sonido y composición algorítmica/asistida son una referencia importante para el desarrollo de esta tesis puesto que, en gran medida, son los responsables del desarrollo de la concepción paramétrica dentro de la música electroacústica. Su empleo extendido en la actualidad define un paradigma de trabajo particular y diferente de las concepciones históricas previas a su aparición. Es en este tipo de programas donde se definen con mayor rigurosidad las estructuras materiales abstractas, como estructuras de datos que representan los materiales musicales, empleadas para la composición.

Lenguajes de programación y recursos gráficos

La representación gráfica de la música por computadora se basa principalmente en dos enfoques. El primero consiste en la representación de señales digitales y el segundo en la representación de las estructuras musicales. Dentro del primer enfoque, teniendo en cuenta solo los recursos que consideran la evolución temporal, se puede diferenciar entre la representación del fenómeno acústico, como variaciones de amplitud en el oscilograma o la energía en el espectrograma, y las funciones envolventes como representación analítica de la evolución de diversos parámetros en el tiempo. Estos tipos de representaciones derivan del análisis matemático del fenómeno físico y se emplean predominantemente en las aplicaciones de edición y mezcla de audio donde la señal acústica es el material trabajado. En cambio, la representación de estructuras musicales está orientada hacia otro tipo de abstracciones que denotan estructuras conceptuales, empleadas para la producción de materiales sonoros pero que no necesariamente representan el fenómeno físico resultante (e.g. los programas de secuenciación). En los entornos gráficos de programación coexisten ambos enfoques puesto que las técnicas de producción musical necesitan de ambos tipos de recursos en distintas etapas de desarrollo.

De la misma manera que sucede con los lenguajes de programación, es muy extensa la cantidad de software que emplea recursos gráficos, con mayor o meno grado de consistencia, para representar sonido y estructuras musicales. Es por eso que se realiza una selección de los que resultan más importantes teniendo en cuenta aspectos específicos y distintivos en relación a su finalidad. Por un lado están los entornos que implementan recursos de representación y secuenciación de parámetros de síntesis y posibilitan estructuras algorítmicas complejas como *OpenMusic* (Agon et al. 1999) y *Pure Data* (Puckette 1996). Otro tipo de entornos, como la *UPIC* (Unité Polyagogique

Informatique CEMAMu) (Squibbs 1996), *Iannix* (Coduys et al. 2003, 2004, Jacquemin et al. 2014) e *InScore* (Fober 2012), están más orientados a la secuenciación empleando elementos gráficos y fueron diseñados para cumplir la función de la partitura en la música electroacústica. Otro tipo de aplicaciones informática es el *Acousmograph* (Geslin y Lefevre 2004), que no fue ideado para secuenciar parámetros musicales sino para el análisis de música acusmática.

OpenMusic y su predecesor *PatchWork* (Agon et al. op. cit.), son, en la denominación de sus autores, entornos de composición asistida por computadora. Son tal vez los entornos que más énfasis hacen con respecto a la representación y manipulación de los materiales como estructuras simbólico-musicales ligadas a la tradición de la música occidental. Su diseño originalmente está orientado hacia la generación de materiales musicales y el control de los parámetros de síntesis, aunque es posible desarrollar algoritmos que realicen procesos de síntesis mediante llamadas a programas externos (e.g. *Csound*). Estos entornos, al ser un lenguaje de programación representado gráficamente, posibilitan la creación de librerías, para desarrollar tareas específicas, de la misma manera que los lenguajes de programación de propósito general.

PatchWork introduce la representación de las funciones de programación del lenguaje *Lisp* como elementos gráficos. Con respecto la representación sonora y musical, este enfoque posibilita la representación y edición visual de estructuras de datos, e.g. secuencias melódicas o envolventes dinámicas, por medio de los símbolos de la notación tradicional o la gráfica de funciones insertadas en el flujo de control del programa.

Con *OpenMusic* aparece el objeto "*maqueta*" que posibilita la integración de la representación musical temporal con la lógica de los programas. Una *maqueta* es un tipo de objeto que representa una línea temporal sobre la cual se pueden disponer programas que generan materiales sonoros o instrucciones de control, archivos de audio y secuencias MIDI de manera interrelacionada. Los objetos *maqueta*, a su vez, se pueden agrupar jerárquicamente y temporalmente para producir estructuras musicales complejas. Internamente, los distintos tipos de objetos contenidos dentro de estos objetos emplean recursos de representación visual como la notación musical, el oscilograma, y el *piano roll*. Aunque provee una gran flexibilidad en la organización temporal de materiales diversos, el enfoque propuesto en *OpenMusic* diversifica las características de los objetos representados debido principalmente a sus connotaciones simbólico musicales. Esto hace que sea difícil pasar de las estructuras de datos que representan señales de audio a las que representan estructuras o eventos musicales como señales de control o comportamientos formales. En el desarrollo de esta **Tesis** se verá como es posible solucionar este problema cambiando el nivel de representación base de las distintas entidades entendidas como materiales musicales de manera tal que todas compartan el

mismo nivel de referencia posibilitando su organización y manipulación de manera unificada.

Pure Data es un entorno de programación gráfica para síntesis de sonido muy similar a *Max/MSP*. Ambos entornos están orientados a la programación de la lógica del procesamiento de señales de audio y de control, con los recursos heredados de la familia de lenguajes *Music N*, como **abstracciones instrumentales** (Puckette 2002). Contienen herramientas para la secuenciación de sonido y pueden definir estructuras de datos de más alto nivel para representar abstracciones musicales contenidas dentro de la lógica del flujo de señales.

Dentro del entorno *Pure Data*, su autor desarrolló una herramienta de representación gráfica denominada "*Data Structures*" (Puckette 2002b) que propone una manera original de representar parámetros de control temporal mediante el empleo de figuras geométricas en dos dimensiones. Los elementos gráficos no tienen un significado predefinido sino que pueden ser asignados arbitrariamente a cualquier parámetro de una **abstracción instrumental**. La representación de los valores de control está determinada por las propiedades gráficas del objeto: posición, ancho, contorno superior y contorno inferior, pudiendo controlar uno o más parámetros al mismo tiempo. El espacio visual sobre el cual se posicionan los elementos gráficos actúa como una línea temporal pero sin predefinir estructuras ni escalas temporales. Lo que resulta interesante destacar de este trabajo es la simplicidad con que se relacionan los principios básicos de la representación gráfica de figuras geométricas con la representación paramétrica del sonido. Dentro de un conjunto acotado de elementos gráficos y principios de relación se proporcionan recursos flexibles para la representación gráfica de estructuras musicales. Sin embargo, la generalidad y la falta de especificidad de las relaciones entre los objetos visuales y los parámetros representados es lo que hace que el recurso sea difícil de interpretar. Por cada trabajo es necesario definir casi todas las relaciones que dan sentido a la organización visual lo que hace que el recuro de representación sea poco eficiente.

La *UPIC*, desarrollada por el compositor Iannis Xenakis a finales de la década de 1970 (Squibbs 1996) (Thiebaut et al. 2008) (Coduys et al. 2003), dio origen a un nuevo paradigma de interfaz gráfica para la representación sonora en la música por computadora. En los últimos años se han producido varios desarrollos, que rescatan, varían y expanden sus principales ideas, entre los cuales se destacan los entornos *HighC*, *Hyperscore*, *MetaSynth* (Thiebaut op. cit) y *Iannix* (Coduys op. cit.) (Jacquemin et al. 2014). De manera similar a las "*Data Structures*" de *Pure Data*, estos entornos se basan en el mapeo de trazos y figuras geométricas a parámetros de síntesis. El espacio bidimensional representa el tiempo en el eje de las abscisas y, principalmente, la altura en el eje de las ordenadas. Sobre este espacio se pueden dibujar recorridos que representa

variaciones de frecuencia e intensidad. Los elementos de síntesis sonora suelen ser un conjunto restringido de recursos que pueden ser provistos externamente por *software* específico. Según la técnica de síntesis empleada, el espacio de dibujo puede se entendido como contenedor de estructuras simbólicas independientes (voces instrumentales como **comportamientos abstractos**) o como un espacio espectral unificado.

Particularmente, el entorno *Iannix* emplea una concepción espacial del tiempo diferente. Este no se expresa convencionalmente en el eje de las abscisas de hizquierda a derecha sino que se representa mediante trayectorias en el espacio, las cuales pueden tener cualquier sentido y dirección. Este entorno, y de manera similar *InScore* (Fober 2012), tienen además funcionalidades extendidas como partituras interactivas y pueden ser utilizados como controladores en tiempo real.

Un rasgo común a todos estos entornos es que no representan los distintos elementos necesarios para la síntesis sonora de manera estructurada. Su diseño está orientado hacia el control paramétrico pero no integran los niveles de síntesis sonora como sucede en los lenguajes de programación antes mencionados.

Por último, otra herramienta de representación gráfica de la música electroacústica que propone un paradigma distinto a los anteriores es el *Acousmographe* (Geslin y Lefevre 2004). A diferencia de los entornos mencionados hasta el momento, el *Acousmographe* no fue diseñado para el control y la organización de la síntesis de sonido sino para el análisis de los fenómenos resultantes. La utilidad del *software* se basa en realizar anotaciones sobre la representación de las señales sonoras contenidas en archivos de audio (e.g. forma de onda y espectrograma). Estas anotaciones pueden ser dibujos abstractos o símbolos musicales que codifican distintos elementos discriminados perceptivamente, o representaciones espectro-morfológicas (Smalley 1986) que consisten en anotaciones realizadas sobre el espectrograma para resaltar distintos aspectos del comportamiento espectral dinámico. Aunque los recursos gráficos no difieren de los empleados por los entornos antes descriptos, esta aplicación aporta el recurso de la representación simbólica alternativa de fenómenos acústico relevantes para la compresión analítica.

CAPÍTULO 3: Marco Teórico Desarrollado

Introducción

En este capítulo se presentan los elementos necesarios para definir las características del modelo de representación musical que se expone en esta **Tesis**. Para el desarrollo de los distintos aspectos que actúan en la elaboración de una composición musical se toma como referencia el concepto de **material musical**.

Se definirá operativamente al **material musical** como cualquier abstracción que pueda ser entendida musicalmente tanto como sonido en sí mismo o como generadora, modificadora e incluso organizadora de sonido. La generalidad de esta definición es intencional puesto que se quiere contemplar la mayor cantidad de procesos que intervienen en la composición musical. Como se verá más adelante, está sustentada en los conceptos básicos empleados para la organización y manipulación de entidades musicales, pudiendo abarcar principios formales e instrumentales. Se considera que una obra musical se compone simplemente de **materiales musicales** relacionados y organizados temporalmente.

El modelo se basa en la interacción de distintos aspectos teóricos complementarios:

- Primero se definen los **dos tipos de materiales básicos** (**objeto** y **proceso**) según sus cualidades representacionales en relación a la teoría del conocimiento y sus características temporales.
- Luego se expone cómo el factor temporal define cualidades específicas, relacionadas con la representación tradicional de los materiales musicales, según actúen a una determinada escala temporal.
- Una vez definidas las entidades básicas y sus posibilidades de significación relativas, se procede a la exposición de los principios relacionales que posibilitan su agrupamiento en entidades más complejas.
- Complementariamente, el significado específico de las primitivas representacionales y las estructuras compuestas depende del **nivel simbólico base** que el compositor adopte para la realización del trabajo. Este último factor ya no es una cualidad objetiva universal sino una especialización de los recursos disponibles que actúa como una **restricción voluntaria**. Según la delimitación del campo de estudio adoptada en esta **Tesis**, el nivel más amplio (menos específico) que un compositor puede adoptar para la realización de una obra electroacústica es el que impone la lógica del ordenador como máquina de cómputo.

Uno de los ejes fundamentales de este análisis es la división en solo dos tipos de materiales que pueden actuar como primitivas aplicables a cualquier sistema de

representación de musica electroacústica: los **objetos** y los **procesos**. Mediante el empleo de estos dos constructos, relacionados con la tradición musical y clasificados por la teoría de conocimiento en lo referido a los recursos informáticos, se puede solucionar el problema de la escisión de la representación y manipulación entre estructuras de alto y bajo nivel de abstracción, campos representados, tradicionalmente, como síntesis de sonido y composición algorítmica o asistida. Se verá también como, al considerar los procesos como un tipo de representación de los materiales musicales, se vuelve clara la relación entre estos y la forma, puesto que ambos son entendidos como abstracciones similares a distintos niveles temporales. Mediante esta clasificación, los procesos musicales dejan de ser funciones de control como comportamientos accesorios a los objetos musicales y pasan a ser entidades que actúan, aunque de manera diferente, al mismo nivel de jerarquía. Incluso, el concepto de **proceso** puede resultar más cercano a la naturaleza del **material musical** que el de **objeto** puesto que el primero, al igual que el sonido, necesita intrínsecamente del tiempo para existir.

Composición Sonora

Tradicionalmente, la acústica trata sobre la naturaleza física de las señales sonoras y la teoría musical abarca diversas técnicas de elaboración musical. Sin embargo, existe un campo intermedio que articula ambos extremos que puede ser definido como *composición sonora*.

A raíz del surgimiento y desarrollo de la música electroacústica se fueron creando diversos marcos teóricos que abordan los procedimientos de síntesis y el análisis/catalogación/ transformación (la división original entre música electrónica y música concreta) de sonido. A partir de ellos se fueron elaborando teorías sobre la *granularidad* temporal de la elaboración sonora ("*luthería*" sonora, procedimientos y herramientas de síntesis, elaboración a escala micro temporal). En épocas más recientes, debido a los avances sobre la tecnología informática y su mayor disponibilidad para los compositores, se fueron desarrollando, de manera más difundida, diversos procedimientos relacionados a esta, como son: la composición algorítmica y la posibilidad de controlar los procesos sonoros más allá del *evento-nota* como material atómico para la composición (Roads 2002). Nótese que la composición algorítmica puede no ceñirse a los procesos automáticos sino también a la elaboración de entidades, sonoras o musicales, empleando medios informáticos. Estas transformaciones en el pensamiento teórico musical no son exclusivas de la música por computadora, se encuentran ya presentes en los razonamientos de las vanguardias musicales del siglo XX y su influencia mutua pareciera ser más probable que su transmisión de un campo al otro.

Estos nuevos elementos teóricos tienen su razón de ser en los cambios técnicos, por un lado, pero sobre todo también en la concepción de **material sonoro**. Se pone en tela de juicio qué sonidos pueden ser **materiales musicales** y se expande su espectro a cualquier sonido generado por la naturaleza o la técnica humana. Los compositores comienzan a trabajar con los límites temporales en la elaboración sonora. De aquí surgen las preguntas sobre la naturaleza de los sonidos como **materiales musicales**, sobre cómo hacen los compositores para manipularlos, darles forma y transformarlos, de qué elementos técnicos disponen y sobre qué parámetros actúan (qué variables se pueden manejar o analizar y en qué consisten).

A raíz de esto, aunque con eje en las técnicas informáticas, se puede empezar a depurar una teoría de combinación no específica, compuesta de distintas entidades fundamentales que dan forma, mediante los **recursos instrumentales**, a cualquier tipo de material sonoro. El **medio instrumental** no es una variable secundaria del problema estudiado: si se aborda este estudio entendiendo al ordenador como instrumento musical, es posible abarcar un espectro más amplio de posibilidades sonoras en cuanto a su *granularidad* temporal y su composición o análisis paramétrico.

Composición de Materiales

La división más importante para este análisis ya fue formulada mediante la distinción entre material como **proceso** y material como **objeto**10. El material entendido como proceso es una entidad que obtiene un mayor grado de objetividad al realizarse en el tiempo y el espacio. Un proceso o procedimiento11 define el comportamiento de “algo” a lo largo del tiempo. Un objeto, en cambio, es la delimitación de una entidad temporal. Es así que la materialización de un proceso de síntesis o un procedimiento compositivo puede convertirse en un objeto, o varios, según la escala temporal y estructural en la que se desarrolle. A su vez un objeto puede ser ese “algo” cuyo comportamiento está definido por un proceso y cuyas variables pueden cambiar en el tiempo o permanecer estáticas. La idea de **material musical** se vuelve ***recursiva***.

Por ejemplo, un algoritmo de **síntesis abstracta** es la descripción de un conjunto de procesos que pueden generar elementos musicales concretos. Los procesos pueden ser funciones oscilatorias, envolventes dinámicas, procesos de filtrado, etc. Al ejecutar el algoritmo, provisto de parámetros específicos, se genera una instancia sonora particular. El resultado es un objeto sonoro que puede ser significado como nota musical, la cual tiene principio, duración y características dinámicas propias. El algoritmo definió los posibles comportamientos dinámicos de los componentes internos del sonido que luego fueron puestos en forma concreta al producir una nota musical. Delimitada temporalmente

y almacenada en un archivo de audio, la nota se vuelve un elemento estático, invariable, en comparación al proceso generador. Este objeto-nota pasó de estar representado abstractamente por los procesos generadores a estar representado concretamente como señal de audio.

El concepto de ***recursión*** se integra en el siguiente paso: Si se toma el archivo de audio que contiene al objeto-nota como dato de entrada de otro algoritmo de síntesis, o como material de un procedimiento compositivo, el objeto resultante de los procesos anteriores se vuelve parte de los nuevos. Como parte de otro algoritmo de síntesis, el objeto-nota ahora puede estar cumpliendo una función similar a la que cumplían, por ejemplo, las envolventes dinámicas en el paso anterior. Si se toma como material de un procedimiento compositivo, el objeto-nota se vuelve ese "algo" manipulado que da forma a un **material musical** a mayor nivel de complejidad, como pueden ser, una melodía o una textura sonora, por dar dos ejemplos simples.

Analizando los factores temporales e instrumentales, que en el caso de la informática serían las abstracciones empleadas para representar los materiales, se pueden adjudicar algunas propiedades más a esta definición de material. A partir del sonido digital, el valor de una muestra, expresada como un solo número que representa el nivel de presión sonora en un instante, también atómico, definido por el período de muestreo, pasa a ser la entidad manipulable más pequeña. Es posible concebir procesos a frecuencias mayores que la frecuencia de muestreo, sin embargo, es necesario definir un nuevo período, más pequeño, que discretize el proceso que queremos materializar, lo mismo sucede con períodos de control pero desde una escala temporal mayor. Cualquiera sea el caso, el procedimiento no varía. Se mantiene la concepción discreta del fenómeno práctico aunque sus propiedades semánticas puedan cambiar al cambiar drásticamente de escala temporal. Por lo tanto, **a partir de un límite temporal inferior se pueden empezar a construir estructuras cada vez mayores (*agrupamientos*) que representen sonidos o estructuras materiales más complejas.**

Aunque hasta aquí pueda parecer que se hace referencia meramente a la representación física de los fenómenos sonoros, es importante recordar que se están empleando medios informáticos para la creación sonora, y que estos emplean la representación física-matemática del sonido. Sin embargo, los principios de discretización y las cualidades de las escalas temporales tienen su propia validez más allá del medio empleado.

Para la ingeniería de software, esta convivencia de dos *universos simbólicos* (García Amilburu 1998) diferentes, la representación informática por un lado y la representación musical por otro, es lo que sugiere marcar una escisión entre ambos. Esto se debe a que el dominio del problema, en términos informáticos, suele restringirse a las abstracción de

las estructuras musicales más estandarizadas. En relación a esto se puede mencionar la distinción entre representación *simbólica* y representación *sub-simbólica* (véase por ejemplo Bresson y Agon 2007). La primera hace referencia a las estructuras musicales (notas, acordes, etc.) que simplifican la complejidad de los eventos sonoros al contenerlos dentro de unidades simbólicas que definen comportamientos convencionales (predefinidos por el sistema adoptado). La segunda, se refiere a los símbolos informáticos empleados en los programas para representar las estructuras musicales y los procesos de síntesis, es decir, los datos necesarios para el procesamiento. Por ejemplo, una envolvente dinámica representada como una **lista** o **vector**, es considerada una abstracción informática (*nivel sub- simbólico*), en cambio, si se la representa como regulador dinámico, adjudicándole sus propiedades tradicionales, se vuelve elemento de representación musical (*nivel simbólico*).

El problema que puede surgir al aplicar una escisión determinante, entre lo que el compositor manipula (o lo que le está permitido manipular por la aplicación informática) y lo que le corresponde hacer a la computadora automáticamente, es la pérdida de flexibilidad. Al emplear abstracciones de más alto nivel, que simplifiquen procesos complejos pero que permitan resultados más ricos con menor cantidad de datos, se restringen las posibilidades de manipulación y elaboración de concepciones compositivas. Sin embargo, es una distinción totalmente válida en la práctica musical puesto que **cada compositor puede definir cual es su *nivel simbólico* y a partir de ello trabajar en la composición**. El *nivel simbólic*o podría ser entendido entonces como el **nivel de abstracción que se toma como base para empezar a elaborar el discurso musical** , e incluso podría considerarse la elaboración de un *nivel simbólico* como parte de la identidad musical y sonora de una obra o compositor. Aunque es evidente que las abstracciones requeridas por el *nivel sub-simbólico* son mucho más difíciles de aprender y dominar, estas no deberían, en principio, ser puestas fuera del alcance de los compositores. Por otra parte, es posible concebir la elaboración de materiales musicales mediante **niveles simbólicos variables** como se verá más adelante.

Material como Objeto o Proceso

Tipos de conocimiento

Desde el punto de vista teórico, existen dos tipos básicos de conocimiento que se diferencian en la manera de representar la información. Estos son el *conocimiento procedimental* y el *conocimiento declarativo* (véase, por ejemplo, Honing 1993). Aplicados por las ciencias de la computación, éstos definen los recursos de representación que

operan en la producción de algoritmos y datos.

A grandes rasgos, el *conocimiento procedimental* se relaciona con los procesos algorítmicos como estructuras que generan resultados mediante secuencias de acciones ordenadas lógicamente. Los datos, que pueden ser representados mediante estructuras complejas, se relacionan con el *conocimiento declarativo* en cuanto son objetos concretos que pueden ser referidos y descriptos según sus cualidades. Usualmente se los suele distinguir como el conocimiento de "cómo hacer" algo como diferente del conocimiento "acerca de" algo (Honing 1993).

Más allá de sus adecuaciones al estudio de la cognición, estas nociones delimitan los conceptos de **proceso** y **objeto**. Un **proceso** se puede definir como el método o medio por el cual se obtiene un resultado, mientras que, un **objeto** se puede definir mediante la descripción o el análisis de sus cualidades simbólicas. La diferencia radica en el carácter temporal dinámico o estático de ese algo representado. Temporalmente, si se considera el resultado de un **proceso** dinámico como la culminación en algo, ese algo resultante puede ser considerado estáticamente puesto que su existencia puede ser definida en términos de sus cualidades. Es posible describir sus cualidades desconociendo los procesos que las generaron. De esta forma, **en términos temporales se puede considerar lo estático como resultado de lo dinámico de manera causal**. A su vez, si se considera al **proceso**, como un algo compuesto de cualidades dinámicas, para poder realizarlo es necesario definir límites internos que puedan ser objetivados como acciones, y estas acciones pueden estar definidas por las cualidades y el comportamiento de objetos aún más elementales. En este sentido, **los procesos pueden ser entendidos como compuestos de distintos objetos o como objetos con cualidades variables**. Por un lado, se puede poner un **objeto** al lado de otro y comparar, la comparación es un **proceso externo** que no afecta las cualidades estáticas de los objetos pero necesita de estos para poder ser definida. Por otra parte, se puede ir cambiando el objeto de manera tal que varíen sus cualidades internas, ese **cambio de estado** es un proceso que dinamiza las propiedades del objeto volviéndose parte de este.

Un ejemplo paradigmático de esto es la interacción de los componentes de un microprocesador que dependen del flujo de energía para definir sus cualidades. A medida que se asciende en los niveles de abstracción, la diferencia entre objeto y proceso depende del punto de observación del fenómeno entendido como conocimiento y su composición puede ser definida en términos el uno del otro en distintas situaciones.

Esta concepción variable entre objeto y proceso tiene que ver con la *atención* (como foco) que se le presta a aspectos internos de una estructura que se vuelven también estructuras, lo cual se estudiará más adelante en relación a las nociones organizativas. Schaeffer (1988 pg. 168) nota este fenómeno entendiendo que cambian los criterios (como

intención) al cambiar el nivel estructural en relación a la percepción. En esta **Tesis** se pretende mantener los mismos criterios en todos los niveles estructurales, de manera recursiva, pasa simplificar y generalizar estructuralmente la variedad de entidades significativas. Este fenómeno de variación conceptual será denominado como **atención estructural**.

Esta **Tesis** se inclina por la noción de proceso como más elemental, ontológicamente hablando, que la de objeto. El objeto puede ser entendido como un momento en la transición continua de estados generados por la interacción de procesos. Como la captura de un momento específico en el espacio. El tiempo cumple un rol importante en relación a nuestra percepción. Los procesos demasiado rápidos o demasiado extensos no pueden ser percibidos.

Una montaña es un objeto, y puede ser descrito según sus cualidades aparentemente invariables. Si la delimitamos temporalmente según nuestra capacidad de percibirla, la entendemos como objeto. Pero estamos ignorando, en principio, los tiempos geológicos. Puesto que el ritmo de cambio de los procesos que generaron el objeto son demasiado extensos, cuesta entender que ese objeto en el espacio es un momento en la transición de estados que le dieron origen y que le darán fin. Solo se conoce el movimiento el cual se puede apreciar como transición de estados. En relación a esto, Sockhausen desarrolla la noción de estructura como proceso o estado del material musical (según sus propiedades dinámicas o estáticas) en el concepto de *forma momento* (Stockhausen 1989).

Lo que importa destacar aquí es que los **procesos** y **objetos** son dos conceptos que sirven para referir **acciones** y **estados**. Ambos pueden ser entendidos como el “objeto” de estudio o manipulación. Los procesos en sí mismos pueden ser representados como objetos si se prevé su comportamiento, acotado temporalmente, de manera abstracta. Por ejemplo, una definición de síntesis es un algoritmo generador que no necesariamente está delimitado temporalmente, puede generar **materiales concretos** de duraciones muy distintas. Sin embargo, si se sabe que ese algoritmo va a generar un material de una duración determinada, con parámetros conocidos de antemano, **es posible emplear la representación de los procesos abstractos, y prever su *realización*, en lugar de concretar el material sonoro para poder representarlo**. Como se verá más adelante, según se elija uno u otro tipo de representación se puede estar afectando las posibilidades de manipulación de los materiales musicales.

El material como objeto

Históricamente, el concepto de *objeto sonoro* fue desarrollado por Schaeffer

(1988). Los *objetos sonoros* son fenómenos acústicos complejos que pueden ser capturados, almacenados y reproducidos, y su definición como *entidades temporales finitas* determina su cualidad fundamental. Los *objetos sonoros* son *entidades temporales discretas* que se pueden abarcar mediante un análisis retrospectivo. Para Schaeffer, este análisis retrospectivo se basa en los principios de la escucha reducida, es decir, la percepción del sonido en sí mismo, desprovisto de connotaciones semánticas ajenas a la materia sonora y no necesariamente relacionadas con la medida acústica.

Para esta **Tesis**, el eje de la discusión no se centra en los factores perceptivos sino en las cualidades abstractas que pueden actuar de manera descriptiva y que, por lo tanto, se relacionan con el *conocimiento declarativo*. Estas cualidades abstractas se asocian con las medidas físicas de los objetos dejando de lado las estructuras perceptivas que actúan como constructos en otro plano de análisis o nivel de representación. Esto se debe a que la finalidad del tipo de representación propuesto en este trabajo busca generar puntos de referencia objetivos sobre los cuales se puedan construir distintos tipos de conocimiento sobre los **materiales musicales**. A causa de ello, en el sistema teórico de representación sonora desarrollado en esta **Tesis**, los rasgos que se disciernen como característicos de los objetos sonoros representados son sus cualidades temporales predeterminadas.

Un **material sonoro concreto**12 puede ser entendido como el resultado de una serie de procesos que lo componen, pero la visión de esos procesos se pierde dentro de la concepción de objeto. El objeto pasa a ser una entidad delimitada que actúa como punto de partida tanto para su análisis como su composición en relación a otras entidades. Es una unidad manipulable que se puede fraccionar, mediante recursos de edición básicos, en unidades más pequeñas, o se puede emplear en relación a otros materiales concretos. Mediante el análisis es posible determinar los procesos que dieron origen al objeto, pero una vez representado como objeto sus posibilidades de manipulación y transformación son distintas.

El concepto de *objeto sonoro* (Schaeffer op. cit.) es más específico que el de nota musical como unidad significante. Para Schaeffer, la nota musical se presenta como un dato que sobreentiende las “ideas” musicales (op. cit. pg. 28) pero que no actúa como entidad significante de la misma manera que los objetos sonoros en relación al material fonético. Sin embargo, no parece haber discrepancia en la naturaleza objetiva, en relación a lo temporal, entre ambos tipos de abstracciones.

De manera similar, para Wishart (1996), la notación musical es una idealización de los fenómenos sonoros. Estas abstracciones ideales actúan como una simplificación del universo sonoro y se representan de manera discreta mediante notas musicales. Para este autor, el empleo de los recursos de la notación tradicional hace que se pierda la noción básica de *continuidad* del sonido presente en las articulaciones intrínsecas del discurso

musical.

Esta dicotomía, expresada por Wishart, entre la *continuidad* del sonido y la discretización impuesta por la notación musical se relaciona directamente con la discusión sobre si la representación musical se constituye esencialmente de elementos *continuos* o *discontinuos* (véase por ejemplo Honing 1993 y el estado del arte en el **Capítulo 2**). Estas aparentes contradicciones manifiestan las dualidades que existen entre los distintos tipos de abstracciones que se emplean, de manera complementaria, para representar los fenómenos musicales. El empleo de estas abstracciones se relacionan con el enfoque del análisis de ciertas cualidades de los fenómenos estudiados y la concepción temporal aplicada.

> "The distinction between continuous and discrete is not always clear-cut, and it may alternate between levels of representation; for example, a note may be represented by a discrete description at the specification level, a continuous audio signal at the realization level, and a discrete set of sample values at the signal-processing level." (Dannenberg, Desain y Honing 1997 pg. 272)

Como se hace notar en la cita previa, el problema de la *continuidad*/*discontinuidad* de los materiales musicales depende del nivel estructural de representación empleado. Las nociones de *continuidad*/*discontinuidad* se relacionan con las cualidades de los materiales como procesos u objetos respectivamente, por la correspondencia que existente entre estos y la representación del desarrollo temporal (proceso/continuidad) de una entidad y su representación como elemento discreto de valor único (objeto/discontinuidad). Esto a su vez se relaciona con la noción de **atención estructural** (ver apartado anterior) como la acción que define la elección de uno u otro criterio.

El objeto se define en esta **Tesis** como algo de existencia previa que se analiza en el presente, claramente contextualizado y delimitado temporalmente, que puede servir como unidad de sentido para la constitución de entidades de mayor jerarquía. Pero el problema de esta noción es que no contempla los procesos generadores como elementos originales.

La noción que se considera más clara respecto de este tema es la de Deutsch, quién entiende que el objeto musical es resultado de la acción de "*los mecanismos* [perceptivos] *de agrupamiento en música sobre un conjunto 'dado' de elementos acústicos básicos*" (citado en Honing 1993). Si bien esta explicación solo provee una definición formal, resulta suficiente para abarcar los mecanismos de relación, entre objeto y proceso, que dan origen a los elementos estudiados a distintos niveles estructurales, los cuales pueden ser entendidos inherentemente de distintas maneras y pueden ser compuestos, transformados y analizados consecuentemente.

Un ejemplo de procedimiento compositivo sobre un **material concreto** (aunque expresado como **relación abstracta**13), como podría ser una sucesión14, es la

retrogradación. Una melodía o una serie se pueden retrogradar, puesto que, aunque representan entidades temporales, su representación es discreta y se conoce de antemano la duración y la totalidad de los elementos que involucran. En el caso de una serie de alturas que no especifica las duraciones de las mismas, como se da habitualmente en la teoría de la música atonal, la relación temporal es simplemente una relación de orden. Si se empleara una representación temporal causal, la retrogradación no sería un procedimiento posible, pero sí lo sería la inversión, puesto que para invertir los intervalos de una secuencia de alturas no es necesario conocer los eventos futuros sino solamente la relación con los eventos previos.

El material como proceso

Los *procesos sonoros* son la contrapartida de los objetos y actúan según las características del *conocimiento procedimental*. Históricamente los procesos musicales son referidos como *procedimientos compositivos* en un sentido particular: como la evolución temporal de un determinado comportamiento que genera forma. Esta concepción generalmente no contempla los *procesos musicales* (**materiales lógicos**15) que generan o modifican objetos sonoros (**materiales concretos**) como unidades más pequeñas, los cuales se suelen referir simplemente como parámetros en relación a los recursos de síntesis de sonido o variaciones expresivas en relación a la interpretación. Estas cualidades se relacionan con la noción de *continuidad* en contraposición a la forma discreta de representación que implican los objetos y los recursos de notación más empleados (Honing 1993, Whishart 1996).

Un *proceso* define un comportamiento que genera un resultado, e.g. un *crescendo* o un *glissando*, las inflexiones de la voz cantada, la aumentación progresiva de las duraciones en una sección o entre motivos musicales (continuidad formal), etc. Estos procesos actúan a distintas escalas temporales representando el desarrollo temporal. Afectan distintos parámetros tanto del sonido como de los materiales musicales entendidos como entidades formales abstractas. En el caso del sonido instrumental, la continuidad se define, en su forma más elemental, dentro de los eventos musicales (notas). Sin embargo, existen múltiples ejemplos de recursos y técnicas que extienden la notación musical para poder representar *procesos sonoros* continuos entre eventos musicales. Se tratan algunos de ellos a continuación:

- En la notación de *glissandi* continuos en la música de Xenakis, se emplean recursos que extienden la notación tradicional para representar la continuidad, incluso cuando el medio instrumental no posibilita la continuidad expresada en la concepción y notación de la obra. En este caso se fuerza conceptualmente al

medio de producción sonora para realizar una aproximación al ideal expresado en la partitura.

- Lo mismo sucede con la notación analógica, mediante esta se pueden representar gráficamente procesos continuos de manera natural que, aunque no puedan ser realizados con la misma precisión, en la práctica se busca una aproximación a las intenciones que se expresan de manera gráfica.
- Incluso dentro de la notación musical tradicional existen los *modificadores* (Pope 1989) que se expresan de manera analógica y definen procesos concurrentes que afectan un determinado parámetro. El ejemplo más común de esto son los reguladores dinámicos.
- En el caso de la notación de la voz cantada o hablada la continuidad en las inflexiones de la voz se producen implícitamente en relación a la emisión (notación silábica del texto, ligaduras de expresión o prolongación, nota repetida o cambio de nota entre sílabas), y estas en relación a la respiración como gesto corporal. Se definen procesos concurrentes que pueden ser entendidos como superposiciones de procesos continuos que, a su vez, conviven con eventos discretos como la pronunciación de las consonantes.
- Otro recurso de continuidad es la elaboración de texturas. Aunque algunos de los instrumentos musicales empleados individualmente no puedan concretar, de manera ideal, la continuidad del sonido, estos se pueden relacionar temporalmente de manera tal que se produzca un resultado equivalente. Un ejemplo de esto es la heterofonía, en este tipo de textura la continuidad se produce como resultado de la superposición de sonidos discretos.

En cada uno de los casos mencionados subyacen las nociones de *continuidad* y *proceso sonoro*, aunque con recursos específicos dependientes de cada nivel de abstracción considerado.

En relación a la forma, el concepto de *continuidad* puede ser expresado incluso si existen mediaciones que interrumpen la continuidad del sonido. Un determinado **material musical** complejo puede estar compuesto por un proceso que afecte alguna de sus cualidades. Por ejemplo, se puede pensar en la repetición de eventos discretos similares, los cuales van siendo progresivamente de duraciones más largas. Tanto desde la composición como desde la percepción, podemos agrupar conceptualmente los eventos por similitud y contemplar su evolución progresiva. En este ejemplo, podría incluso haber una mediación formal dada por la interpolación de otros materiales sucesivamente yuxtapuestos, pero la recapitulación como continuación del proceso de aumentación progresiva se concibe conceptualmente como proceso de continuidad formal 16. El concepto de *proceso* se adapta de la misma manera que en los casos anteriores y actúa como factor de cohesión y continuidad a nivel formal.

En base a estos ejemplos, se puede determinar que los *procesos musicales* y sonoros definen un tipo de comportamiento distinto y complementario al de los objetos

temporales.

Los procesos pueden estar actuando a distintos niveles, tanto conceptuales como temporales, y sobre distintos parámetros, tanto de entidades simples como complejas. El concepto es el mismo y se entiende como la continuidad que define la evolución de un comportamiento. Definido de manera general, es claro como puede actuar tanto internamente, dentro de objetos sonoros, como a nivel formal.

Los procesos musicales se pueden componer con las mismas técnicas que se componen los objetos. Se pueden superponer, de manera concurrente o paralela, o se pueden suceder y agrupar para generar forma.

El comportamiento resultante de un proceso puede ser entendido, *declarativamente*, como un objeto. Sin embargo, los procesos no definen cualidades temporales concretas, determinan el comportamiento, que puede tener principio y final sin especificar necesariamente su duración (e.g. un comportamiento evolutivo entre dos valores concretos que no especifica ni la duración ni la razón de cambio).

Denominaciones de material lógico y material concreto

Antes de introducir las relaciones entre las representaciones de los materiales musicales como objetos o procesos, es necesario realizar aclaraciones terminológicas y conceptuales.

Los elementos informáticos empleados para representar materiales musicales como procesos son denominados en esta **Tesis** como **material lógico**. Debido a su origen en los recursos computacionales, el término puede ser empleado también para referir, de manera general, a los programas y los datos empleados para componer y generar una pieza musical. En ambos casos el significado es prácticamente equivalente y su referencialidad queda explicita en el contexto de su aplicación.

El concepto de **material lógico** es preferido en ciertas ocasiones por su relación con la generación de datos. Un programa informático puede recibir datos de entrada y genera datos de salida. El programa puede ser considerado entonces como el generador de los datos de salida. Por lo tanto, un **material musical** entendido como proceso puede ser denominado tanto como **material lógico** o **material generador** según se quiera destacar la funcionalidad específica que cumple en un determinado contexto.

De manera complementaria, el material representado como objeto puede ser denominado como **material generado** puesto que se considera que para su existencia fue necesario un proceso de generación previo. Los materiales como objetos, según la definición previa dada en esta **Tesis**, también pueden ser referidos como la instancia concreta de un proceso generador y, por lo tanto, pueden ser denominados, de manera

general, como **materiales concretos**. Este último término tiene además sus orígenes en la denominada "*música concreta*", basada en la noción de objeto sonoro de Pierre Schaeffer (op. cit), pero no refiere específicamente a esta.

El cambio de estado y la manipulación

La representación alternativa de objetos y procesos, aplicada a los recursos musicales durante la composición, afecta las cualidades de las entidades manipulables. Al afectar las propiedades de una entidad mediante distintas representaciones, se posibilitan o inhiben distintas formas de control y esto, a su vez, puede afectar el desarrollo de distintas concepciones aplicables a la composición de los materiales.

El **material lógico** se puede concebir como **generador** de **materiales concretos** de la siguiente manera: Si consideramos el efecto de un determinado proceso al ser aplicado sobre otras entidades musicales, la resultante, tal vez más compleja, puede ser un objeto cuyo comportamiento interno está definido por el efecto de ese proceso, cualquiera sea la escala temporal. Pero el **material lógico** deja de ser el objeto principal de la representación, y pasa a ser una característica interna de un **material concreto**. Al pasar de ser el recurso que define la abstracción y su representación a ser característica interna, se pierden sus cualidades intrínsecas como entidad manipulable, las cuales se transforman en cualidades descriptivas. **El material lógico pasa de ser sintáctico a semántico al ser visto desde una determinada posición estructural**.

De esta manera, el **material lógico** es el algoritmo generador y el **material concreto** es el resultado generado y, por lo tanto, **puede se entendido como el mismo material en distintos estados según la representación empleada**. Este ***cambio de estado*** posibilita diferentes tipos de representación y, lo que es más importante, diferentes tipos de manipulación. Desde el punto de vista de la informática, no es más que la separación entre los datos y las funciones u operaciones sobre los datos. Pero debe ser tenido en cuenta que ambos, datos y funciones, actúan como medio de representación de los materiales musicales y que, en muchos casos, pueden ser representaciones alternativas de una misma entidad musical a diferentes niveles de abstracción.

Como elementos a ser manipulados, la diferencia que existe entre el **material lógico** y el **material concreto** es la misma que existe entre una función matemática y una cantidad *n* arbitrariamente finita de valores generados con esa función. Se puede modificar la función para obtener distintos resultados y también se puede modificar el resultado, si embargo, los tipos de modificaciones que son posibles de efectuar en una u otro son diferentes. En concreto, si se tiene un algoritmo que genera una secuencia armónica aleatoria en base a un repertorio de acordes, se pueden modificar los parámetros del

algoritmo, los cuales podrían ser los coeficientes de probabilidad y el repertorio de acordes. En cambio, si se trabaja sobre el **material concreto**, generado por el algoritmo y representado como una sucesión de eventos, ya no es posible modificar los mismos parámetros de la misma manera, la intervención sobre el material generado sería de tipo "manual", modificando evento por evento o mediante otro algoritmo que realice esta tarea de manera automática.

Como abstracciones que representan entidades temporales, la diferencia fundamental entre estos dos estados consiste en que el **material concreto** representa necesariamente algo finito, mientras que el **material lógico** puede representar entidades no delimitadas temporalmente. La representación de los materiales como entidades finitas posibilita distintos tipos de manipulaciones compositivas muy comunes de la tradición musical, por ejemplo, la variación, la permutación, la retrogradación o, de manera más general, la reorganización temporal de materiales preexistentes *ad libitum*. Este tipo de manipulación del material sonoro es de la misma naturaleza que los recursos de edición básicos de los editores de audio, por ejemplo: cortar, copiar y pegar. Por su parte, representar los materiales musicales como algoritmos permite una codificación más compacta y otro tipo de flexibilidad al representar estructuras temporales.

Tiempo

Aplicación de las escalas temporales

Esta sección trata sobre como las escalas temporales definen las cualidades de las abstracciones fundamentales, los materiales lógicos y concretos.

Ambas abstracciones fundamentales, los procesos y los objetos, se definen por su estado relativo. El **material concreto** existe una vez que los procesos que le dan origen se desarrollan en el tiempo y son representados de manera finita como un conjunto de relaciones constitutivas. Pero esto no explicita por qué un mismo comportamiento, aplicado a un mismo parámetro, puede generar estructuras simbólicamente tan diferentes como el timbre del sonido, el comienzo de una nota, o la dinámica expresiva de una frase musical.

Este problema, aplicado a la modulación en amplitud, es expresado por Di Liscia de la siguiente manera:

> "*Hasta cierto punto, la evolución en amplitud de una forma de onda es una característica de la señal y, hasta cierto punto, es la señal. Esto es así porque la forma de onda de una señal acústica no es otra cosa que un gráfico de la evolución de su amplitud en el tiempo*" Di Liscia (2004 pg 46)

Relacionando este fenómeno con la práctica musical, el autor diferencia tres

categorías dentro de un rango temporal:

> · "*Forma de onda:* […] *.05 seg. hasta 1/LHz (donde LHz es el límite de frecuencia audible). La evolución en amplitud no se relaciona con la intensidad del sonido, sino con su* ***espectro****.*
> • *Cualidad de superficie:* […] *1 seg. a 0.5 segundos. La evolución en amplitud no se relaciona con la intensidad del sonido, sino con su* ***cualidad de superficie****, o* ***rugosidad****.*
> · *Envolvente:* […] *cuya duración supera, aproximadamente, 1 seg. La evolución en amplitud se relaciona con la* ***intensidad*** *del sonido.*" Di Liscia (op. cit. pg 46, negritas agregadas por el autor de esta **Tesis**).

Esta clasificación demuestra como el entendimiento de la música electroacústica puede asignar distintos significados a un mismo **comportamiento abstracto** según las concepciones materiales y las técnicas instrumentales aplicadas.

La escala temporal en la cual se desarrollan los procesos sirve como indicio básico para determinar estas posibles cualidades resultantes. La relación con la percepción sonora es obvia, el rango audible del período de las oscilaciones de presión que producen sonido en un medio comienza a partir de los 0.05 segundos y desciende hasta los 0.00005 segundos aproximadamente. Cualquier proceso oscilatorio que se encuentre dentro de ese rango será percibido como sonido y probablemente se represente mediante sus cualidades espectrales. Algo similar sucede con los transitorios de ataque, las *cualidades de superficie* y la envolvente dinámica de un sonido. A mayor escala temporal, las variaciones de amplitud pasan a ser variaciones expresivas que pueden afectar grupos de objetos sonoros como expresión dinámica. Según la escala de Roads (2002), los factores enumerados pertenecen a niveles diferentes: *microsonido*, *objeto sonoro* y *meso tiempo*.

Es importante notar como un proceso se define también por su comportamiento (oscilatorio o continuo) y el parámetro al cual se aplica. Las variaciones de presión que generan sonido son un procesos oscilatorio, mientras que la envolvente dinámica de una nota musical es un proceso continuo que puede actuar sobre el proceso oscilatorio previo. A su vez, la dinámica de frase es otro proceso que puede actuar sobre un conjunto de notas musicales y, por lo tanto, sobre ambos procesos previos. Esto es un ejemplo básico de composición/transformación de **materiales lógicos**. Si bien matemáticamente el concepto es muy sencillo, es importante notar su relación con las estructuras simbólicas que generan y cómo estas definen conceptos variables que se componen a partir de un único tipo de abstracción mediante la aplicación de diferentes relaciones.

La delimitación simbólica de los objetos resultantes está implícita en la representación y es variable. Esta falta de definición explícita de los objetos resultantes es propicia como medio de representación estructural puesto que puede abarcar la totalidad

de relaciones posibles sobres las cuales se puede desarrollar el léxico musical sin sobrecargar los elementos representacionales.

Propiedades temporales de los materiales

Las entidades analizadas que se consideran **materiales musicales concretos** deben definir rasgos y relaciones temporales *implícitas*, *explícitas* (Honing 1993). Las relaciones temporales *implícitas* se deducen de valores temporales absolutos puestos en referencia a una línea temporal en común, mientras que las relaciones temporales *explícitas* son las que se especifican directamente como la relación conceptual entre dos entidades (e.g. antes de, durante, después de, etc.).

Para que estas relaciones puedan ocurrir se necesita un ámbito de referencia el cual puede ser un **contexto** en común, como en el caso de una línea temporal, o una entidad de existencia previa, por ejemplo, cuando algo está "antes de", se considera conceptualmente la existencia previa de una entidad que se toma como punto de referencia. En ambos casos la referencia es una entidad en común que puede o no actuar jerárquicamente al mismo nivel que los elementos relacionados.

Estas relaciones se diferencian de la concepción *tácita* del tiempo (Honing op. cit.) en la cual solo está definido el instante presente. En el *tiempo tácito* como **contexto**, existen relaciones temporales que se pueden definir como "antes" o "después" de un punto en grado variable, pero el instante de referencia no es fijo, cambia constantemente con el presente. La representación *tácita* es útil puesto que simplifica las relaciones temporales inmediatas y se emplea comúnmente en los algoritmos de síntesis. Como **abstracciones instrumentales** (ver **Capítulo 4**), estos algoritmos actúan en *tiempo real* recibiendo eventos de control y alterando el flujo de la señal de salida. Desde el punto de vista del control externo son entidades temporalmente "estáticas", objetos instrumentales que necesitan de una acción ajena para que el sonido producido pase a formar parte de un **contexto temporal**. Por lo tanto, las entidades analizadas que se consideran **materiales lógicos** deben definir rasgos y relaciones temporales *tácitas*.

La representación matemática abstracta de las funciones de síntesis puede ser entendida como la unidad fundamental que emplea inherentemente la representación *tácita* del tiempo. Como se explicó anteriormente, las **abstracciones instrumentales** también pueden ser descompuestas en procesos temporales mediante la previsión de los datos generados por las funciones matemáticas, pero la representación *tácita* del tiempo implica un límite temporal indefinido para las **abstracciones materiales** representables.

Material	Relación temporales	Procesamiento temporal
Concreto	Implícita/Explícita	Fuera del tiempo (no-causal)
Lógico	Tácita	En el tiempo/Tiempo real (causal)

Tabla 1 Propiedades temporales de los materiales concretos y lógicos.

En la Tabla 1 se muestra un resumen de las propiedades temporales de los **materiales concretos** y **lógicos** complementada con la clasificación de Xenakis (1971) vista en el **Capítulo 2**.

Relaciones Estructurales

Relaciones abstractas y realización

Hasta aquí, el **material musical** se considera como tal si adquiere significado temporalmente. Sin embargo, es necesario considerar también otro tipo de relaciones que se producen fuera del tiempo musical pero que contribuyen a su constitución y desarrollo por medio de la planificación de los recursos **restringidos voluntariamente** para una composición musical.

Ciertas estructuras organizativas, como las escalas o conjuntos de alturas, el repertorio rítmico, tímbrico, o de intensidades pueden ser desarrolladas en abstracto antes de ser puestos en forma dentro de una composición musical, actuando como los datos de un proceso aún no definido.

Un un *pitch-class set* (Forte 1974), como recurso de organización de las alturas, es ejemplo de este tipo de organizaciones. En abstracto, un conjunto de alturas puede no representar una relación de orden17 entre los componentes del conjunto, solo garantiza que, de alguna manera a ser definida posteriormente, estos van a actuar como un determinado tipo coherencia interválica. Un conjunto de alturas es entendido como una unidad en abstracto, sin tiempo. En sí mismo, es simplemente la relación estructural de un parámetro musical que puede formar estructuras temporales.

Otro recurso similar denominado *registración fija* (e.g. Di Liscia 2013) empleado en composiciones musicales como criterio de organización y coherencia de la altura, es un ejemplo claro que se relaciona con la noción de escala musical como repertorio de alturas restringido. La *registración fija* es la selección y disposición de un conjunto de notas en el registro, como un acorde invariable sobre el cual luego se creará un desarrollo temporal

haciendo aparecer y relacionando las notas en el tiempo.

Las organizaciones abstractas pueden especificar relaciones de orden sin especificar duraciones concretas. La relación de orden es, en cierta medida, temporal, puesto que implica una relación causal, de tiempo explícito (antes, durante o después). Si se tratara, por ejemplo, de una serie dodecafónica, existe una relación con la noción de material concreto puesto que puede ser manipulada como elemento discreto de manera abstracta. Es por esta característica que la organización formal de una pieza musical se puede determinar de antemano sin especificar el contenido. Se pueden emplear relaciones abstractas que definen la cantidad y variedad de secciones postergando la realización de su contenido. Las abstracciones del comportamiento formal comúnmente expresadas mediante letras, e.g. ABA como reexposición, ABA' como reexposición variada, ABACABA como forma de rondó, etc, expresan el orden temporal, las relación de contraste y similitud y su importancia organizativa, pero las cualidades que definen carecen de significado más allá de la relación que manifiestan. De manera similar, es común predefinir relaciones proporcionales sin especificar unidades temporales concretas (e.g. el doble, triple, la mitad, etc.).

Los recursos definidos por el diseño de un instrumento, como **interfaz instrumental** (ver **Capítulo 4**), pueden actuar como **relaciones abstractas** que definen un repertorio y posibilitan determinados desarrollos temporales posteriormente.

Este tipo de **relaciones abstractas** son muy comunes como técnica compositiva, principalmente porque son un método de **restricción voluntaria** que define los recursos disponibles. Para su concreción como **material musical** se necesita una etapa de **realización**, la cual implica la adición de propiedades paramétricas, especialmente las temporales, dentro de un determinado contexto de relaciones. Esta es la diferencia entre **relación abstracta** y **realización** como recurso compositivo18.

Las relaciones jerárquicas y estructurales que se definen en el análisis de los recursos compositivos, se valen de este tipo de abstracciones que actúan de manera distinta a las **abstracciones materiales** de **objeto** y **proceso**. El paso de la **relación abstracta** a la **realización** es en cierta forma similar al cambio de estado entre el **material lógico** y el **material concreto** pero difiere en dos aspectos fundamentales. Por un lado una **relación abstracta** no representa el factor temporal como sí lo hace un **material lógico**, las **relaciones abstractas** son estrictamente atemporales, denotan relaciones estructurales de manera estática. Por otra parte la **realización** de una **relación abstracta** implica su concreción mediante la adición de los factores temporales que están representados por los **materiales lógicos** y **concretos**.

Por estas razones, la manipulación de las **relaciones abstractas** difiere substancialmente de la manipulación de las **abstracciones materiales**. Cuando las

cualidades de los materiales musicales vienen dadas por las **relaciones abstractas**, estas son tomadas como cualidades invariantes, mientras qué, cuando son el resultado de las operaciones efectuadas, al estar realizadas temporalmente, son cualidades dinámicas.

Principio de agrupamiento

> "*The domain of music is full of wonderfully complex concepts whose precise meaning depends on the context of their use, their user (be it composeer, performer, or musicologist), and the time in history of their usage.*" Dannenberg, Desain & Honing (1997 pg. 272)

Una vez definidos teóricamente los conceptos materiales básicos del sistema de representación y sus propiedades temporales, resta determinar los tipos de relaciones que pueden producirse entre estos para formar estructuras complejas.

La concepción de relaciones estructurales jerárquicas es un recurso común en la sistematización del conocimiento sobre entidades complejas. Los principios de organización jerárquica están presentes en las teorías de análisis musical y percepción sonora de manera general. Para el análisis musical tradicional, según Deutsch (2013), las notas, implícitamente entendidas como la unidad fonética de la música, se combinan en lo sucesivo para formar motivos, los cuales se combinan para formar frases, las cuales se combinar para formar conjuntos de frases hasta llegar al nivel de la pieza musical. En lo simultáneo existen las estructuras provistas por otras ramas del conocimiento musical, las notas se combinan mediante la armonía y las frases mediante el contrapunto, la textura musical y la orquestación.

Aunque estas categorías de análisis no estén sistemáticamente ordenadas, todas reflejan la concepción de organización jerárquica de relaciones estructurales de elementos del nivel de abstracción inmediatamente inferior, la nota en relación al acorde, la melodía en relación al contrapunto y la textura que agrupa tanto estructuras armónicas como melódicas. En el caso de la orquestación, esta puede actuar incluso de manera transversal sobre estas estructuras generando distintos tipos de agrupamientos. Un buen ejemplo de esto último es la *Klangfarbenmelodie* de la escuela de Viena y, específicamente, *Farben* de Schonberg, donde las variaciones tímbricas orquestales definen la estructura de la pieza por sobre la organización de las alturas.

Si bien los estudios sobre percepción y cognición sonora y musical están fuera del alcance de esta **Tesis**, de ellos se pueden tomar como referencia los principios relacionales que definen su modelo lógico.

El conocimiento sobre la percepción musical se basa en los principios teóricamente

fundamentados por los procesos gestalticos. El extensivo trabajo de Deutsch (2013) sistematiza estos principios aplicados a la percepción sonora al tratar los distintos mecanismos de agrupamientos. Junto con Bregman (1990), estudian en detalle los agrupamientos perceptivos implicados en la percepción sonora de estructuras musicales a distintos niveles jerárquicos que generan determinados resultados perceptivos como la cohesión tímbrica de los sonidos (Deutsch op. cit) o la organización de los flujos musicales (Bregman op. cit.) partiendo de los mismo principios teóricos.

Tenney, en sus estudios sobre los principios de organización estructural de la percepción sonora, estrechamente ligados a sus concepciones compositivas, da una definición simple y general del principio de agrupamiento. Define la "*unidad gestaltica temporal*" (Tenney 1980) en términos de lapsos temporales. Una "*unidad gestaltica temporal*" es un lapso de tiempo, definido a un determinado nivel jerárquico dentro de una estructura de relaciones, que actúa como cohesivo internamente y segregativo externamente. Los factores que definen cohesión o segregación son la proximidad temporal y la similitud paramétrica.

De manera común, el mecanismo de estructuración general es el principio agrupamiento que se basa en la cohesión o segregación de elementos según criterios específicos. Cómo y por qué se agrupan determinados elementos de determinada manera para formar estructuras más complejas depende de los principios relacionales del fenómeno estudiado.

El agrupamiento y la estructuración jerárquica son principios epistemológicos que se emplean para organizar el conocimiento humano. Las técnicas informáticas hacen uso extendido de estos principios tanto para el diseño de sus herramientas (e.g. los lenguajes de programación) como para el análisis y la solución de problemas empleando estas herramientas.

Por un lado, este resumen deja constancia de la relación que existe entre los distintos campos del conocimiento que actúan interdisciplinariamente en la composición con medios electroacústicos. Pero también, hace explícitas las diferencias entre dominios específicos. Los mismos principios estructurales pueden ser empleados simultáneamente para manifestar distintos aspectos que conviven, en distintos planos del conocimiento, sobre un mismo fenómeno estudiado.

Esta **Tesis**, al referir a la estructuración musical según una concepción compositiva y no perceptiva, la cohesión o la segregación de elementos estructurales se puede dar por factores más arbitrarios y abstractos que se relacionan con las herramientas informáticas, y las técnicas y las nociones del compositor. Determinar qué estructuras sería más adecuadas para representar un fenómeno perceptivo es otro estrato de análisis que no necesariamente tiene que se coherente con las estructuras compositivas empleadas para

su realización. Sin embargo, al analizar una obra, el analista se verá obligado a buscar la razón en la elección de los agrupamientos y los criterios de cohesión y segregación en relación al resultado musical y los recursos técnicos.

Como se expondrá en el **Capítulo 6**, el principio de agrupamiento jerárquico es fundamental y actúa como el método de relación de los **materiales lógicos** y **concretos** de manera recursiva. Para el compositor, estos agrupamientos no dependen de factores desconocidos sino de concepciones aplicadas sobre ideas musicales. Para el analista, es obvio que estos factores pueden ser desconocidos y deben ser deducidos por este en ausencia del compositor.

Descripciones estructurales

Dannenberg, Desain y Honing (1997) describen los problemas que surgen de la naturaleza compleja de las descripciones estructurales. Consideran que las representaciones estructurales como frase, metro, voz o armonía, entre otras, pueden ser entendidas de manera entrelazada en forma de red. En ciertos casos, la relación jerárquica "*pertenece a*" es útil para definir la relación de estructuras anidadas del tipo frase y sub-frase cuando estas se comportan de manera homogénea y una representación estructural recursiva resulta adecuada para material analizado (Dannenberg, Desain y Honing op. cit.). Pero reconocen que, a veces, las descripciones estructurales pueden ser ambiguas desde el análisis, e incluso esa ambigüedad puede ser transmitida en la ejecución. Por ejemplo, cuando dos frases musicales se solapan y el final de una es el principio de la otra, o cuando los límites de una frase no coinciden con la estructura métrica. En estos casos la relación "*pertenece a*" no puede ser aplicada de manera consistente (Dannenberg, Desain y Honing op. cit.). Esto se debe a que las estructuras analizadas pueden pertenecer a distintos "planos" estructurales que conviven sobre un mismo fenómeno analizado y, por lo tanto, una única estructura de relaciones jerárquicas no es consistente al describir todos sus aspectos.

Si bien el principio organizativo es el mismo (la relación jerárquica de agrupamientos), su función, aplicada a un determinado aspecto del fenómeno a analizar, varía. Estos principios de organización pueden ser estudiados tanto en relación a la interpretación, percepción y el análisis sonoro y musical como a la composición. La función de los principios estructurales varía puesto que las estructuras empleadas para representar una construcción musical puede diferir de las que resultan del análisis o la percepción.

Los distintos tipos de ilusiones sonoras son un claro ejemplo de esto. La ilusión de Shepard, en la cual distintos tonos en *glissando*, que aparecen y desaparecen, forman un

único tono complejo de movimiento continuo, o la ilusión de escala (Deutsch 1974), en la cual dos melodías separadas espacialmente se fusionan perceptivamente de manera distinta. En ambos ejemplos, los agrupamientos estructurales difieren si se los considera constructivamente o perceptivamente. Esta diferencia, que se genera y queda demostrada en las ilusiones, es un aspecto fundamental a tener en cuenta en la organización de las estructuras materiales básicas.

Distintos agrupamiento y relaciones entre procesos y objetos no necesariamente reflejan un resultado perceptivo sino una **concepción compositiva**. Esta concepción está relacionada, en cierta medida, con los recursos técnicos instrumentales pero también es una dualidad propia de las construcciones intelectuales. El resultado de una pieza musical puede ser abordado desde la composición o el análisis, empleando herramientas similares, y obtenerse resultados que difieren en aspectos estructurales. El análisis de la música electroacústica se vuelve críticamente difícil con respecto a estos problemas si no se dispone de los materiales lógicos con los cuales se elaboraron las obras o una descripción previa de los procesos.

Teniendo en cuenta estos factores, la organización jerárquica de los objetos y procesos expuestos en esta **Tesis** se realiza desde el punto de vista de la composición, pudiéndose emplear en el análisis de obras de las cuales se disponga del material lógico (programas, archivos de audio, descripción o deducción de los procesos, etc.). Para poder abordar el análisis de los distintos niveles o planos estructurales, es necesario contar con una base que actúe como descripción precisa de los fenómenos estudiados, de esta manera, los objetos y procesos involucrados pueden ser descriptos y referenciados de manera objetiva. Situar este nivel de referencia dentro de las primitivas de representación instrumental-intelectual, como objetos y procesos jerarquizados e interrelacionados, parece ser la manera más eficiente puesto que se exponen de explícitamente las estructuras básicas empleadas para la construcción de conceptos de más alto nivel. Por otra parte, el análisis de planos superiores necesita de la incorporación de las propiedades culturales y perceptivas de los constructos analizados los cuales pueden ser cotejados con la concepción expuesta por las relaciones estructurales de los materiales considerados más objetivos.

Estructura objetiva vs estructura subjetiva

En función de lo expuesto anteriormente se concluye que para el análisis de música electroacústica es útil diferenciar entre **estructuras objetivas** y **estructuras subjetivas**.

Las **estructuras objetivas** son las que se derivan de la lógica constructiva de los

materiales musicales al emplear determinadas herramientas de manera abstracta. Son objetivas puesto que pueden ser entendidas mediante el análisis de los recursos instrumentales y las organizaciones explícitas puestas en forma por el compositor por medio de **estructuras materiales**. El análisis de estas estructuras se basa en el estudio del código de programación o los recursos provistos por un tipo de *software* específico. **Es el estudio del material lógico, como equivalente a la notación musical en forma de programa informático**. En otras palabras, la programación trabaja sobre un sistema de relaciones predeterminado por los recursos instrumentales y las prácticas interpretativas (modo de empleo) del cual se pueden derivar cualidades objetivas.

Las **estructuras subjetivas** son las que se construyen mediante la percepción o la interpretación del resultado sonoro. Estas pueden ser percibidas y conceptualizadas de manera distinta con respecto a las estructuras compositivas.

Desde el punto de vista del compositor, **las estructuras objetivas y subjetivas pueden ser intencionalmente manipuladas** como en el caso de las ilusiones acústicas, lo cual es un elemento que se suele tener en cuenta también en el análisis musical. Por ejemplo, en la práctica orquestal tradicional se suelen duplicar voces según las técnicas de la orquestación, la duplicación del bajo o la melodía no implica un resultado contrapuntístico sino una técnica de manipulación tímbrica y registral. Por lo tanto, la relación estructural se puede confundir debido al empleo de los mismos recursos para obtener resultados en diferentes planos.

La disposición de las voces en un acorde orquestal puede estar determinada por sonoridad resultante pretendida, e.g. los acordes que siguen la disposición de la serie armónica omitiendo la fundamental, la cual se reconstruye perceptivamente, es un recurso que posibilita una sonoridad extendida incluso fuera del registro orquestal. Una contrapartida de la orquestación en la música electroacústica podría ser el balance espectral de las técnicas de síntesis o mezcla de sonido. Lo que cambia en estos casos son las **abstracciones materiales** manipuladas. La orquestación está sujeta a los recursos instrumentales de los instrumentos acústicos mientras que el procesamiento de sonido está delimitado por las técnicas de procesamiento. Desde el punto de vista de las abstracciones empleadas, la orquesta necesariamente utiliza los recursos de nota/voz e instrumento mientras que el procesamiento de señales puede estar empleando abstracciones que imiten dicho comportamiento o que sean propias del medio electroacústico, e.g. la abstracción de canal de audio o procesamientos como la ecualización.

Otro tipo de manipulaciones de las estructuras objetivas son las que tienden a lograr resultados psicoacústicos específicos, como por ejemplo el contrapunto lineal (Bregman 1990) la orquestación tímbrica de Ravel o el uso de sonidos diferenciales como **material**

musical. En los casos del contrapunto lineal, los sonidos diferenciales y las ilusiones sonoras, la disociación entre la estructura material provista por el medio instrumental y la estructura percibida es extrema puesto que las **abstracciones instrumentales** dejan de tener relación directa con el resultado esperado. Es decir, que la estructura subjetiva resultante no se intenta representar como tal sino que está implícita en la organización de las abstracciones empleadas. Sin embargo, en estos casos es posible prever el **resultado subjetivo** en base al análisis de las **estructuras objetivas**.

Como ya se mencionó anteriormente, **distintas estructuras objetivas pueden producir las mismas estructuras subjetivas**. Por ejemplo, una mutación tímbrica se puede lograr mediante un procedimiento tan sencillo como la variación relativa de amplitud de la suma de dos formas de onda distintas (a la manera de un *crossfade*), el cual es incluso el recurso técnico empleado en la orquestación de instrumentos acústicos. En la música electroacústica, una mutación tímbrica puede ser lograda mediante las técnicas de procesamiento espectral, en particular la síntesis cruzada. Esta puede ser empleada para realizar una transición o hibridación tímbrica mediante la multiplicación espectral de distintos sonidos. Ambos recursos manipulan distintos parámetros para realizar la mutación y por lo tanto emplean distintas abstracciones para lograr un resultado equivalente19 dentro del espectro de resultados que posibilita cada técnica. Este principio de equivalencia puede ser comprobado empíricamente pero adquiere su fundamentación objetiva de las propiedades técnicas aplicadas.

Ejemplos de relaciones jerárquicas entre objetos y procesos
Notación tradicional

En el siguiente ejemplo se conjugan todos los aspectos tratados anteriormente. Por razones de simplicidad, el fragmento consiste en una melodía ejecutada por un violín elaborada *ad hoc* (Figura 3). La notación musical expresa los componentes básicos centrados en la altura y la duración a los cuales se suman los indicadores dinámicos (Pope 1989), el toque o **modo de acción** (Karkoschka 1972) instrumental (arco tirando o empujando, número de cuerda y pizzicato), la articulación (relacionada con el **modo de acción**), los indicadores de compás y el *tempo* base. En la partitura, los símbolos están expresados alrededor de las duraciones proporcionales expresadas por la notación rítmica como eventos discretos sucesivos. Mediante el análisis, si descomponemos estos factores como capas de procesos estos se pueden expresar como se muestra en la Figura 4.

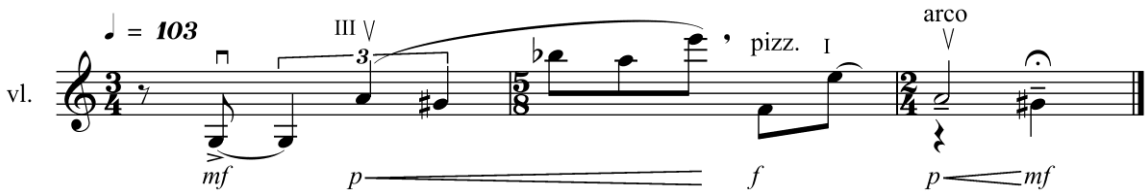


Figura 3 Notación tradicional

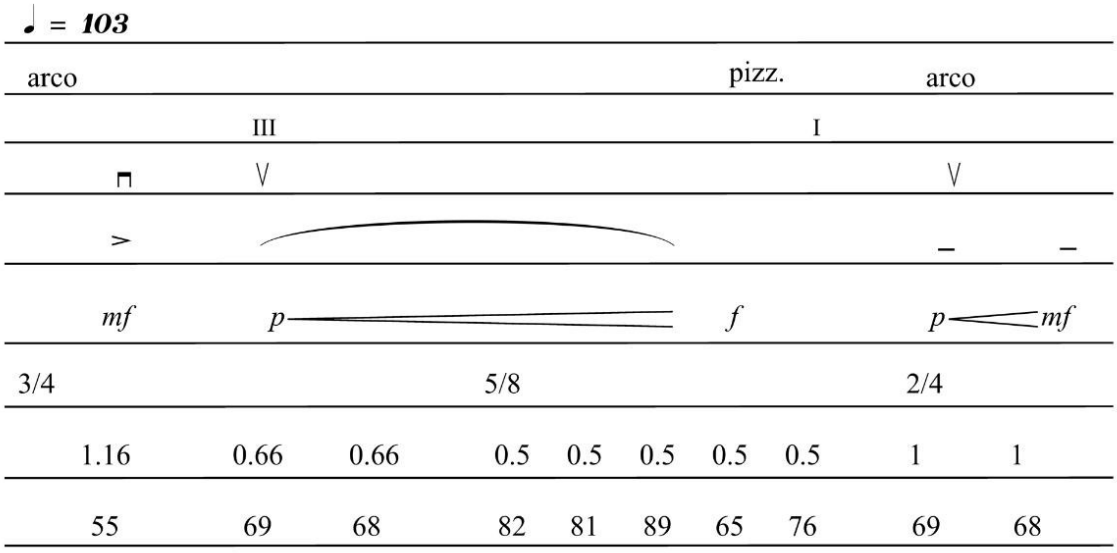


Figura 4 Descomposición paramétrica de la notación tradicional.

A diferencia de la notación tradicional, la resultante visual de la Figura 4 muestra claramente la linealidad de los procesos concurrentes. Estos consisten en sucesiones de valores de un mismo tipo (altura, intensidad, articulación, etc) expresadas, en este caso, a un mismo nivel jerárquico como componentes del agrupamiento a nivel de melodía. Mediante este procedimiento, visualmente se pierde la noción de objeto-nota expresada en la partitura puesto que no queda inmediatamente clara la segmentación de eventos. Esto es, en parte, producto del agrupamiento arbitrario que se realizó en el análisis de los parámetros. La notación tradicional realiza una jerarquización simbólica de las duraciones y las alturas que se destacan visualmente sobre del resto, en cambio, en los lenguajes de programación para síntesis de sonido y composición asistida se suelen emplear representaciones del tipo de la Figura 4 debido a la naturaleza de las estructuras de datos empleadas y su representación en estos entornos20. Pese a estas diferencias, ambas notaciones son equivalentes puesto que representan los mismos parámetros y sus relaciones, sin embargo, una parece hacer énfasis en la noción de objeto y la otra en la de proceso. Si se realiza el camino inverso y se pasa de la representación de la Figura 4 a la de la Figura 3, se puede observar como los objetos musicales se componen de procesos que generan agrupamientos, que pueden ser perceptivos o conceptuales.

Ambas notaciones contienen **relaciones implícitas**, especialmente en lo que respecta a la producción del sonido. En la música instrumental, estas dependen del instrumento musical y los procesos corporales necesarios para producir sonido. En el caso de los algoritmos de síntesis, estos procesos pueden ser especificados de manera explícita como abstracción de un sistema instrumental. Por medio de la **síntesis abstracta** (Bilbao 2009), los procesos necesarios para producir sonido se expresan con las mismas estructuras temporales que las señales de control, comúnmente asociadas con la partitura. Esta economía en el empleo de elementos representacionales está basada en la arquitectura de los entornos informáticos y la lógica matemática, pero manifiesta conceptualmente la homogeneidad de las primitivas empleadas en la representación y la importancia de los principios de agrupamiento. Es por esto que la notación algorítmica se suele considerar como más precisa y general que la notación convencional pese a que, en la práctica, el compositor no disponga de los recursos necesarios para actuar en todos los niveles de abstracción, ya sea por las **restricciones impuestas** por el empleo de una aplicación informática en particular o por la complejidad de los conocimientos adicionales necesarios.

En el caso de la representación informática, la semántica de las estructuras empleadas es variable. En la música, esta variación se da en relación a los principios de agrupamiento y las escalas temporales que se aplican según la finalidad del análisis realizado. Los agrupamientos expresados, reflejan solo una posible interpretación de los fenómenos musicales que puede ser útil como recurso sintáctico que expresa las relaciones entre **objetos** y **procesos** de manera precisa. La estructura sonora resultante puede no ser congruente con la estructura perceptiva, pero esto es considerado como otro plano de conocimiento que se puede derivar de las **relaciones implícitas** en la notación. Lo mismo sucede con la notación tradicional u otras notaciones alternativas, cada sistema hace énfasis en determinados aspectos que son necesarios y suficientes para generar el resultado deseado aunque los procesos de representación internos no sean idénticos en el análisis posterior.

La economía de estructuras básicas (**procesos** y **objetos**) resignificadas mediante los **principios de agrupamiento** posibilita un mayor grado de flexibilidad en la concepción de estructuras resultantes puesto que no está condicionada por factores externos a la lógica sintáctica. Sin embargo, la conveniencia en el empleo de determinadas estructuras de mayor especificidad agiliza el proceso de producción para determinadas concepciones técnicas y musicales. Estructurando los principios básicos de manera adecuada se generan estructuras de datos y algoritmos específicos mediante librerías, como sucede, por ejemplo, en entornos como *OpenMusic* y *SuperCollider*.

Notación en texto plano

El siguiente ejemplo es un programa breve elaborado en el entorno *SuperCollider* por el científico y músico experimental Nathaniel Virgo21 (Código 3). Pertenece a un género definido por el medio de representación y transmisión denominado *sctweet* por la comunidad de *SuperCollider* en internet22. Los *sctweets* consisten en un programa breve, de un máximo de 140 caracteres, para ser difundido mediante el sitio web *tweeter23*. El límite en la cantidad de caracteres y el formato de escritura (en una sola línea) están determinados por el medio de difusión, tweeter permite publicar mensajes de un máximo de 140 caracteres lo que hace necesarios diversos recursos de economía en la escritura del programa notable en la ausencia de espacios, nuevas líneas o indentación . Estos programas son escritos para ser ejecutados en el entorno *SuperCollider* y producir sonido en tiempo real.

Este ejemplo emplea una representación diferente, en varios aspectos, a la notación musical tradicional. La primera diferencia clara es que su representación es algorítmica. El código de programación expresa la lógica de producción del sonido y determina las estructuras musicales resultantes de manera precisa y compacta. El énfasis de la notación está puesto en la representación de los procesos generadores y sus relaciones, y no en las estructuras sonoras resultantes a las cuales se aproxima más la notación tradicional. Al estar orientada a la producción de sonido mediante técnicas de síntesis, la notación no está pensada para su interpretación y realización por parte de otra persona en el sentido tradicional de la práctica musical. Refleja tanto las concepciones técnicas, empleadas para la producción de sonido, como las estructurales, que definen las relaciones entre procesos y parámetros que resultan en comportamientos musicales específicos.

```
play{Splay.ar({Pluck.ar(BPF.ar(f=product({|i|
product({LFPulse.ar(2**rrand(-9,1),2.rand/2)}!(i+2))/(1+i)+1}!
8)*86,f).sin,Saw.ar,1,1/f,9)}!9)}
```

Código 3 sctweet de Nathaniel Virgo

El autor de este programa provee una explicación de su construcción en internet24, aquí solo se analizarán las estructuras empleadas en relación a la representación paramétrica de procesos generadores. El programa consiste en 9 unidades generadoras (*Pluck*) que producen sonido mediante el algoritmo Karplus-Strong (Karplus y Strong 1983). Seccionando las estructuras del programa en relación a los parámetros altura y duración, se puede apreciar que ambos son generados mediante la combinación de osciladores de baja frecuencia que producen ondas cuadradas (la unidad generadora *LFPulse,* que produce pulsos de baja frecuencia y ancho constante en este caso). Por cada unidad generadora *Pluck*, se define una señal de control que afecta las frecuencias (alturas) y los ataques (duraciones) del sonido. Estas señales de control, a su vez, están compuestas por la multiplicación de las señales producidas por varios osciladores *LFPulse*, a distintas frecuencias, para generar secuencias de alturas y ritmos (el cambio de altura determina un nuevo ataque). En resumen, yendo desde los parámetros de control a la producción de sonido, la pieza se compone de osciladores de baja frecuencia que son relacionados mediante distintas operaciones matemáticas para generar una señal que actúa como flujo de control de alturas y duraciones. Es decir, una función compleja definida como la relación entre funciones más simples. Las funciones simples, que son señales de audio, contribuyen a al comportamiento específico de los parámetros pero no tienen, desde la perspectiva de la representación musical, significado directo por si mismas. En este caso, las funciones están actuando a una determinada escala estructural que define el comportamiento de la señal resultante que controla la altura y el ritmo a mayor escala.

En este ejemplo se hace evidente la relación entre nuestra concepción de **proceso** y lo que se denomina tradicionalmente como **procedimiento**. Literalmente los mismos procesos son los que actúan en la producción de sonido (la señal acústica a nivel de micro estructura), los materiales musicales y la forma. La composición de osciladores de baja frecuencia da como resultado una señal de control compleja en su desarrollo, a frecuencia de audio, que contiene la información que se emplea para: generar el sonido, puesto que es usada como señal de excitación del algoritmo de síntesis aunque filtrada y distorsionada; para generar el material que puede ser considerado motívico, e.g. ostinatos o melodías; y para generar las variaciones a nivel formal, e.g. por transposición y variación motívica, variaciones en la densidad cronométrica y en las relaciones entre las voces.

Aunque pueda parecer complejo por la escritura condensada que lo contiene, el algoritmo es relativamente sencillo, sin embargo, las estructuras musicales resultantes son relativamente complejas y variadas. Las mismas funciones generadoras actúan a distintas escalas temporales y afectan las estructuras musicales a nivel de *objeto sonoro*, definiendo los comienzos (que pueden resultar en acordes entre las voces) y el timbre, las estructuras *meso temporales*, e.g. los ostinatos, las frases musicales y el comportamiento imitativo, y las estructuras *macro temporales* que consisten en la variación a nivel formal, e.g. transposición y variación de las estructuras *meso temporales*.

Es interesante destacar cómo la combinación de procesos a distintas escalas temporales y estructurales generan, al actuar concurrentemente, distintas estructuras sonoras y musicales. Los elementos representacionales empleados son mínimos y, mediante su combinación específica, generan una resultante compleja a distintas escalas temporales.

Las estructuras musicales resultantes, conceptuales y perceptuales, no están reflejadas en el algoritmo de manera directa. Estas quedan fuera de los elementos representados por el algoritmo y pasan a pertenecer a otros planos de representación. Para poder estudiarlos, es necesario realizar un análisis de las posibles relaciones resultantes o concretar los procesos, es decir, tomar la salida del programa como dato. En este caso, el algoritmo contiene variables estocásticas que pueden modificar considerablemente el resultado entre ejecuciones. Es interesante notar como esta característica produce una diferencia entre el conjunto de posibles resultados, como la esencia general del algoritmo, y una realización particular. Esto garantiza la coherencia entre distintas ejecuciones y, a su vez, genera una variedad interesante de resultados distintos como si se tratase de un estilo musical que abarca un conjunto de piezas. Es la relación entre el principio de variación y el repertorio de materiales a nivel de pieza o género musical.

Por otra parte, este algoritmo es un ejemplo de cómo los recursos técnicos y, sobre todo, la representación empleada para la composición, afectan las posibilidades de manipulación de las estructuras resultantes. Las abstracciones empleadas y sus relaciones dispuestas posibilitan la variación de parámetros globales. De manera sencilla se podría modificar la cantidad de voces (unidades generadoras *Pluck*), lo cual afectaría en consecuencia la densidad cronométrica y armónica; la cantidad de osciladores que intervienen en la composición de cada una de las señales de control, lo que produce una variación en la complejidad del movimiento melódico; la frecuencia de referencia, lo cual produciría una transposición global del conjunto de alturas posibles; y el tiempo de *decay* del algoritmo de síntesis, lo cual afecta la articulación entre las notas (e.g. *legato* o *staccato*). Sin embargo, si se quisiera modificar el repertorio de alturas sería

necesaria una modificación un poco más profunda del algoritmo y cambiar las relaciones matemáticas que se aplican incrementalmente en las iteraciones. Pero lo que resulta más importante es que no es posible, ni aún especificando los valores aleatorios de manera concreta, controlar individualmente las estructuras motívicas y formales resultantes, puesto que son producto de las relaciones que se producen mediante la composición de osciladores de baja frecuencia según la disposición específica en este algoritmo. Si se quisiera tener un mayor grado de control sobre este aspecto sería necesario cambiar casi la totalidad el algoritmo y las estructuras de datos empleadas. Incluso siendo posible, hay que tener en cuenta que un cambio de esta naturaleza modificaría la concepción misma de la pieza, puesto que se modificarían sus relaciones de coherencia estructural aunque sus elementos constitutivos podrían pasar a ser parte de un proceso de elaboración más general.

CAPÍTULO 4: Teoría Subyacente de los Recursos de Representación Existentes

Interfaces Musicales

El acervo de conocimientos respecto de las técnicas de composición musical es vasto. Por cada aspecto de los factores involucrados en una composición musical existe un tratado que los desarrolla extensamente. En la música por computadora, particularmente en el desarrollo de *software*, esta variedad es difícil de abarcar. Es por esto que las aplicaciones informáticas para componer música suelen tener una funcionalidad específica delimitada a algunos aspectos de la producción musical que se ve reflejada en los distintos tipos de *software* musical estandarizados.

En relación a las cualidades de los elementos representables y su organización temporal se propone en esta **Tesis** una clasificación que diferencia entre **interfaz instrumental** e **interfaz de composición**, la cual resulta práctica para la contextualización de los recursos empleados por distintos tipos de aplicaciones. Esta distinción parte de la misma relación que se expresa en la división orquesta/partitura clásica de los lenguajes *Music N* y similares (Roads 1995) pero actualizada a la realidad híbrida de los entornos más recientes.

Interfaz instrumental

Se define operativamente el concepto de **interfaz instrumental** como una abstracción informática considerada temporalmente "estática" que actúa metafóricamente como un instrumento físico.

Una **interfaz instrumental** puede ser un algoritmo de síntesis manipulado externamente, un banco de sonido, una consola de mezcla virtual o un controlador MIDI (Penfold 1992) u OSC (Wright, Freed y Momeni 2003). La abstracción informática actúa como objeto en sentido físico, necesita de un estímulo/evento de entrada para poder generar o procesar sonido según su programación interna. Desde el punto de vista del control externo, este tipo de aplicación emplea la representación *tácita* tanto del tiempo como de la estructura (Honing 1993), es una entidad que responde a uno o varios eventos/flujos de entrada generando uno o varios eventos/flujos de salida.

Sin embargo, si se analizan internamente este tipo de aplicaciones, se puede observar que existen abstracciones que representan el tiempo de manera implícita. Por ejemplo, en las secuencia programadas o *loops* que se disparan mediante una interfaz

MIDI, existe una organización temporal interna predefinida por el usuario, pero el nivel de abstracción al cual se manipula externamente la secuencia, no necesariamente toma en consideración su constitución interna.

Dentro de esta categoría está el concepto de *definición de instrumento* en entornos como *Csound* (Vercoe 1986) y *SuperCollider* (McCartney 2002) o *patch* (exclusivamente de síntesis o procesamiento) empleado en los lenguajes de programación visual como *Pure Data* (Puckette 1996). Una definición de instrumentos es, principalmente, un programa que actúa como **abstracción instrumental** que procesa o genera señales continuas tanto de audio como de control en relación a otro programa que provee características de secuenciación u otro tipo de interfaz de control en tiempo real. La noción de flujo de datos (*data-flow* en inglés) en oposición a la de flujo de control (Duignan et al. 2005), es característica de este tipo de **abstracciones instrumentales** y está en estrecha relación con la noción de proceso como **material musical**.

Una **interfaz instrumental** se caracteriza por las cualidades temporales de su control externo, puesto que, una vez que la abstracción fue programada, las relaciones temporales internas no son directamente accesibles. Este tipo de abstracciones son empleadas, generalmente, para el procesamiento de sonido en tiempo real, representando el tiempo de manera *tácita* y empleando el paradigma de flujo de datos.

Interfaz de composición

Una **interfaz de composición**, en cambio, actúa como medio para disponer y combinar acciones y procesos temporalmente de manera temporalmente *implícita* o *explícita* (Honing 1993). Dentro de esta categoría se encuentra la programación de "*disparadores*" de *samples* y *loops*, los "*secuenciadores lineales*" (multipistas) y los lenguajes de programación visual y textual (Duignan et al. 2005).

Una **interfaz de composición**, entonces, implica un **contexto** de realización temporal de los **materiales lógicos** y **concretos**. Actúa de manera similar a la partitura de la música convencional pero no define elementos de representación estandarizados. Los editores multipista y los secuenciadores son un ejemplo de aplicaciones gráficas que proporciona una estructura contextual basada en el tiempo. Por su parte, los lenguajes de programación pueden proveer las mismas características mediante la representación textual de los conceptos involucrados.

Así como los **materiales musicales** entendidos como **procesos** y **objetos** se relacionan entre sí como: entidades relativas a distintas concepciones teóricas, recursos simbólicos, escalas de representación, cualidades como agrupamientos y cualidades temporales; las nociones de *instrumento* y *partitura* se comportan de manera similar. En la música por computadora, un instrumento puede generar una textura compleja de largo aliento (véase ejemplo Código 3 en **Capítulo 3**) y una secuencia programada se puede emplear como evento instrumental simple (e.g. un DJ que combina secuencias musicales de largo aliento). La diferencia radica principalmente en la escala a la cual se esté actuando creativamente en relación al nivel material base (nivel de referencia definido arbitrariamente por el compositor, véase más adelante en este capítulo), la concepción compositiva que implican las abstracciones empleadas y el control que se tiene sobre las mismas.

De esta manera, las **interfaces instrumentales** pueden ser representadas dentro de las **interfaces de composición** si se les asigna las propiedades temporales de los posibles **materiales generados**. Es decir, que las interfaces instrumentales pueden ser representadas como **materiales lógicos**.

Ambas interfaces, instrumental o de composición, emplean los mismos recursos recursos computacionales que, combinados, generan abstracciones informáticas muy similares, pudiendo variar en el grado de especificidad o personalización. Para la música electroacústica, la diferencia histórica entre *orquesta* y *partitura* es una distinción teórica necesaria para organizar la técnica y el trabajo compositivo. Los recursos que posibilitan actualmente las computadoras superaron ampliamente sus restricciones técnicas iniciales y comienzan a poner en evidencia las necesidades de desarrollar nuevos marcos conceptuales.

Control y representación

Los editores de partitura se centran en la edición gráfica de la notación tradicional, pudiendo reproducir la música anotada mediante instrumentos MIDI, pero el resultado sonoro no suele ser satisfactorio debido a la falta de control sobre ciertos parámetros de las inflexiones de los instrumentos musicales. Los entornos para síntesis de sonido posibilitan la elaboración del sonido como material en sí mismo pero suelen carecer de herramientas de representación a *nivel simbólico*- formal. Si se emplean de manera combinada, ambos tipos de aplicaciones informáticas enfocan la atención sobre aspectos complementarios haciendo uso de distintas representaciones de los materiales musicales.

La edición de una partitura, aborda la representación simbólica tradicional que define objetos abstractos los cuales se pueden combinar para formar estructuras musicales.

Pero esta representación no define el material sonoro concreto sino una idealización basada en la simplificación propuesta por el concepto de nota musical (Wishart 1996), cuya realización concreta depende de la versatilidad en el control de los algoritmos de síntesis (bancos de sonidos, síntesis abstracta, modelado físico) o una ejecución instrumental. Para poder sistematizar el mapeo de los parámetros de la partitura tradicional con los algoritmos de síntesis es necesario tener en cuenta y asignar el significado específico de los símbolos, especialmente los que representan *notaciones de acciones* (Karkoschka 1972).

Un proceso de estas características suele ser demasiado engorroso e incluso imposible de realizar debido a que el *software* no fue diseñado para abarcar todas las variantes posibles. La notación paramétrica empleada por los algoritmos de síntesis es más directa con respecto a la representación del sonido resultante pero no necesariamente respeta la jerarquización de los parámetros musicales propuesta por la partitura. Incluso es mucho más difícil representar ciertas acciones instrumentales, que en el caso de la partitura se anotan con relativa simplicidad, puesto que pueden implicar el control simultáneo de un conjunto considerable de parámetros de síntesis.

La situación ejemplificada es consecuencia del empleo de distintos sistemas de representación que no resultan totalmente compatibles. Por un lado está la representación tradicional y, por otro, la representación de los recursos instrumentales informáticos. Ambos ideados en distintas épocas, en relación a medios de producción diferentes.

Unificar el control y la representación de los materiales musicales, de manera que sea lo suficientemente conveniente como para satisfacer ambas necesidades es una de las metas de esta **Tesis**. Emplear de manera integrada las **interfaces instrumentales** y la **interfaces de composición** es la solución propuesta por este trabajo. Como se verá en el **Capítulo 6**, mediante la representación de **materiales lógicos**, es posible unificar los parámetros de control temporal con los algoritmos de producción sonora en un mismo espacio de trabajo achicando la brecha que existe en la actualidad.

Simplificación impuesta por el software

En comparación con la notación tradicional, los secuenciadores de *samples* y *loops* suelen adoptar una representación mucho más simplificada en cuanto a la cantidad de parámetros que representan y controlan. A diferencia de la notación tradicional, basada en la altura y la duración como parámetros *morfofóricos* de la representación, los secuenciadores se basan en la noción de evento y duración. La noción de evento reemplaza a la noción de nota. La diferencia radica en que el principal parámetro del evento no es necesariamente la altura. Un evento puede representar un *sample* y dentro

de este pueden haber sonidos estructuralmente simples o complejos en simultáneo o sucesivamente, e.g. una secuencia melódica breve, un acorde, un sonido con morfología espectral desarrollada, etc. El evento puede estar representando estructuras/agrupamientos perceptivos/compositivos (de manera similar a la notación neumática del canto llano). Por su parte, los *secuenciadores lineales* simplifican al máximo la representación estructural definiendo simplemente los conceptos de *pista* y *clip* de audio (Duignan 2005). Una *pista* no representa la estructura de los materiales musicales sino de la funcionalidad específica de la herramienta que fuera diseñada para manipular flujos de audio.

El empleo de un conjunto de elementos definidos por las **relaciones abstractas** (e.g. un repertorio de sonidos), y las **abstracciones instrumentales** (como objetos compuestos de comportamientos pre-definidos), pueden actuar como las abstracciones informáticas que representan acciones complejas con respecto a una inflexión instrumental. Estos recursos se suelen emplear como técnicas que simplifican la complejidad de la manipulación de los materiales musicales, a costa de la pérdida en la felxibilidad en la manipulación de sus detalles. De manera similar a como sucede con la notación musical que omite especificar gran cantidad de parámetros sonoros y expresivos de manera precisa.

Según el paradigma que implemente un determinado entorno de representación de la música electroacústica, este puede estar propiciando la simplicidad en pos de la facilidad en la manipulación de las abstracciones musicales sacrificando la flexibilidad, o puede propiciar el control comprensivo de los parámetros musicales en deterioro de la facilidad de uso. Esta distinción es muy común entre los entornos de creación musical basados en interfaces gráficas y los lenguajes de programación para síntesis de sonido y composición algorítmica/asistida.

Este problema no parece tener solución así planteado. Sin embargo, mediante el empleo de lenguajes de programación es posible programar las abstracciones de alto nivel que luego serán empleadas mediante recursos de control simplificados, como técnica de desarrollo en dos etapas. Esto no es más que un ejemplo de la relación que existe entre las **restricciones impuestas** y las **restricciones voluntarias**. Al elegir un sistema de *software*, a menos que se pretenda de antemano un uso muy específico, es teóricamente correcto optar por el que permita el mayor grado de flexibilidad a costa de la simplicidad e implementar las técnicas de desarrollo necesarias para elaborar las distintas etapas de una composición electroacústica. Incluso, este enfoque define rasgos estilísticos particulares entre distintos compositores en base a sus concepciones compositivas.

Recursos Informáticos

Los lenguajes de programación como medio de representación musical

> "Acceptance of the limitations of music notation only leads to a deeper entrenchment of old concepts and a difficulty imagining what lies beyond." (Dannenberg 1996)

Los lenguajes de programación de propósito general son un medio de representación del conocimiento expresado con la precisión de la lógica matemática-computacional que pueden ser entendidos como "*herramientas de pensamiento*" (Iverson 1980). Pese a que para la representación musical muchas veces no es necesario, ni pretendido, un grado extremo de precisión en la notación de las ideas, es claro que los lenguajes de programación pueden actuar como medio de representación musical de la misma forma que sucede con estos en otros campos del conocimiento.

El empleo de lenguajes de programación para componer música puede ser entendido como una forma de *notación musical extendida* (Dannenberg 1996) capaz de expresar cosas que la notación tradicional no puede. Como *notación extendida*, los lenguajes de programación posibilitan la creación de programas como partituras interactivas y la hibridación entre la notación, la composición y la interpretación al incorporar la toma de decisiones y la aleatoriedad (Dannenberg op. cit.). En la actualidad, esto se demuestra en la música mixta, que emplea ampliamente el procesamiento de señales en tiempo real como parte indisociable de la composición y la interpretación.

Como recurso instrumental, en prácticas más recientes como el *live coding*25 (Collins et. at. 2003) (Wang y Cook 2004), los lenguajes de programación para síntesis de sonido en tiempo real son empleados como instrumento musical en un sentido más estricto. Los programas que generan la música son escritos sobre el escenario, muchas veces proyectando el código de programación de manera que sea visible para el público (McLean et. al. 2010). Existen incluso orquestas de *laptops*26 (Trueman et. at. 2006) (Bukvic et. al. 2010) como conjunto instrumental en relación a la tradición del concierto.

El empleo de lenguajes de programación garantiza una forma de representación musical, pero no especifica sus cualidades.

> "*While the function a program is to perform can influence the choice of language in wich the program is written, it is also true that a programming language can influence the conception of a program's function. In a more general sense, programming concepts can suggesst functions that might not occur to one outside of the context of programming. This is of signal importance in music composition, since the integration of programming concepts into the musical imagination can extend the boundaries of the imagination itself.* " (Chowning 1995)

Chowning expresa que estas cualidades pueden variar según el diseño particular de distintos lenguajes. ¿Hasta qué punto la lógica de los lenguajes de programación, las estructuras de datos y el modelo computacional inciden en nuestro conocimiento de la materia estudiada? Son notables las relaciones que existen entre los modelos de la teoría del conocimiento, los cuales pretenden universalidad, y los modelos computacionales, aunque muchas veces los segundos derivan de los primeros es posible que exista una retroalimentación. Eso es lo que parece expresar Chowning al final de la cita. El conocimiento musical previo, al encontrarse con un nuevo medio, adquiere nuevos recursos influenciados por sus características. Esto no implica connotaciones negativas sino que se acepta como un hecho natural, puesto que se podría realizar el mismo razonamiento en relación al desarrollo de los recursos de representación previos a la aparición del ordenador. Es el mismo conocimiento el que crea las herramientas y las interpreta.

En cuanto a la discusión sobre el empleo de herramientas informáticas de mayor o menor nivel de abstracción, suele haber consenso en que las técnicas compositivas que pretenden emplear el ordenador de manera creativa dentro de un proceso artístico deben tratar de buscar el mayor grado de flexibilidad por sobre la especificidad/comodidad. En relación a la composición asistida, Bresson y Agon lo expresan de la siguiente manera:

> "*An important idea of computer-aided composition is that as compositional processes can be (at least partially) carried out and represented by programs, then programming languages are the most relevant compositional supports for computer-musicians. Composers should then be considered as programmers rather than just as simple users of a given program.*" (Bresson y Agon 2007)

De manera más específica Dannenberg, Desain y Honing dan un ejemplo de cómo los recursos computacionales se manipulan en relación a las concepciones compositivas y las abstracciones informáticas, y de cómo estos pueden ser adaptados de distintas maneras:

> "[...] when building a chord function it may be parametrized naturally by the pitch contour of the root, but it has to be known in advance whether at some time in the future we will want to de-tune on specific note of the chord. Once foreseen, it is easy to write a chord function with an extra parameter for this capability, but in general it is hard to decide on the appropriate amount of encapsulation in defining compound musical objects for general use. (This is also one of the reasons why we believe in composition systems in the form of general programming languages with their capabilities of abstraction; there are so many styles of music composition that any pre-wired choice of the designer of a composition system is doomed to be wrong for some composer at some time.)" (Dannenberg, Desain y Honing 1997 pg. 289, negrita agregada por el autor de esta **Tesis**).

Al emplear lenguajes de programación de propósito general, estas adaptaciones se pueden realizar tantas veces como sea necesario. Una característica de los lenguajes de programación de propósito general es que, en teoría, pueden representar cualquier idea concebible. Esto es lo que los convierte en un medio de representación extremadamente flexible tanto para la composición musical como para otras ramas del conocimiento.

En relación al análisis musical, se podría pensar que, disponer del programa que genera una pieza musical equivale a tener la representación de (muchos de) los procesos mentales que el compositor empleó para crearla. Aunque en realidad la relación pueda no ser tan directa, es evidente que se dispondría de un documento que aportaría cierto grado de precisión en relación al objeto de estudio.

El diseño modular en relación a la composición

La modularidad, como principio epistemológico aplicado al diseño del software, se ve reflejada en el modo de empleo de las herramientas de más alto nivel. Tanto los entorno para síntesis de sonido como el empleo complementario de distintas aplicaciones informáticas para la manipulación de señales y la puesta en forma de una pieza musical, implican una concepción modular.

En el desarrollo interno de una **interfaz instrumental**, la combinación lógica de unidades de procesamiento de señales implica la utilización de módulos. Concepción que se aplica, desde sus inicios, en el desarrollo de entornos para **síntesis abstracta**.

Lo mismo sucede en el desarrollo de una composición mediante una **interfaz compositiva**. El empleo de bancos de sonidos, sintetizadores, secuenciadores y editores multipista implica la combinación de herramientas que fueron diseñadas para interactuar de manera modular. Por ejemplo, en los editores multipista, los archivos de audio, las cadenas de procesamiento (*efectos*), las automatizaciones y las secuencias MIDI son las **abstracciones instrumentales y materiales** que se usan para dar forma a la técnica musical empleando medios digitales.

Estas herramientas actúan como el *universo simbólico* (García Amilburu 1998) que los compositores manejan para la producción musical electroacústica. Muchas de ellas fueron diseñadas en analogía con los principios de la música instrumental, por ejemplo, el empleo de secuencias y bancos de sonido, guarda una estrecha analogía con la composición pianística y la orquestación del resultado, característica del período romántico. Lo cual, a su vez, se relaciona con el concepto de nota como entidad musical idealizada (y simplificada) de Wishart (1996).

La concepción modular de los procesos de síntesis y las herramientas de control actúan como **abstracciones instrumentales** y expresivas (performáticas) de una pieza musical (Borin, De Poli y Sarti 1997). Las nociones empleadas implican un cambio respecto de las concepciones tradicionales puesto que se concibe a la composición electroacústica como comprensiva de etapas anteriormente divididas entre la composición y la ejecución.

La modularización es el método que integra las abstracciones necesarias para la producción musical desde las **relaciones abstractas** hasta la **realización** sonora. Los módulos representan estructuras a distintos **nivel de abstracción** posibilitando mayor o menor grado de control. Pueden ser implementados para solucionar una problemática técnica específica, aplicable a un estilo compositivo en particular, o como parte de una concepción holística sobre el conocimiento musical al ser usados en conjunto. Como concepción holística, la modularidad es la definición de las primitivas de representación y sus principios relacionales, lo cual puede dar indicios de los procesos y las técnicas creativas empleadas en una determinada composición musical mediante el análisis de las herramientas empleadas incluso si actúan como **restricciones impuestas**.

El diseño modular en relación a la ingeniería de software

Dada la generalidad del ordenador como máquina de computo, la ingeniería de *software* separa técnicamente los recursos computacionales del dominio de un problema específico. Implícitamente se suelen considerar al ordenador y a los lenguajes de programación de propósito general como meta herramienta y meta lenguajes empleados para la solución de problemas del mundo real. El dominio del problema a solucionar computacionalmente es delimitado y definido por el alcance del análisis realizado previamente por los desarrolladores. Es así que surgen herramientas computacionales de dominio específico, como lenguajes y aplicaciones musicales, cuyo alcance está restringido a la realización de tareas específicas. El problema de la generalidad se puede solucionar mediante la complementariedad modular de los algoritmos.

Sin embargo, en la práctica, la teoría no se cumple incondicionalmente. El desarrollo de sistemas es un trabajo arduo y costoso y, por razones de mercado, muchas veces se tienden a ver soluciones parciales que intentan abarcar una cierta cantidad recursos en base a los requerimientos de los usuarios que realizan tareas específicas. Por ejemplo, la edición de partituras, la grabación y la mezcla de audio en DAWs (acrónimo en inglés de *Digital Audio Workstation*, también referidas en esta **Tesis** como editores multipista), la secuenciación de archivos de audio, la síntesis y el procesamiento de sonido, etc.

Los editores de partitura tienen la finalidad principal de facilitar la edición visual de la notación tradicional. Dada la generalidad de la computadora, la siguiente meta lógicamente fue poder secuenciar la música escrita, generar el sonido a partir de la grafía convencional, y agregarle recursos expresivos a esa secuenciación que imiten la calidez de un intérprete y sean fieles a los recursos instrumentales. Esta simple extensión resultó ser una tarea muy difícil de realizar por razones técnicas varias pero, sobre todo, porque el diseño original del software está basado en el dominio de la edición visual de partituras en base a la notación tradicional. Las estructuras simbólicas necesarias para la realización visual difieren substancialmente de las empleadas para la expresión y la síntesis de sonido.

Paralelamente se desarrollaron herramientas de secuenciación de archivos de audio cuyas interfaces se adaptan mejor a la lógica del dominio de esta tarea puesto que su origen prácticamente coincide con el de la informática. Para la comunicación entre instrumentos de síntesis y secuenciadores se desarrollo el estándar MIDI (Penfold 1992) que actúa como protocolo de comunicación de **abstracciones instrumentales** basadas en la lógica de los sintetizadores por *hardware*. Al añadir la capacidad de secuenciación a los editores de partituras, sus recursos se vieron afectados por las restricciones del protocolo MIDI y los recursos de síntesis y *sampleo*. La restricción de los 16 canales MIDI y la exclusividad de ciertos canales como la percusión hace impráctica la secuenciación de conjuntos instrumentales amplios, e.g. una orquesta, puesto que no es posible siquiera representar más de 16 voces monódicas con comportamientos dinámicos independientes. Si bien la modularidad es un principio extremadamente versátil, este es un ejemplo paradigmático de cómo una implementación demasiado específica puede dificultar tareas que el ordenador es capaz de solucionar sin problemas. El inconvenientes es que el dominio del problema se va ampliando, desde lo específico a lo general y la interacción modular debe ser planificada como un dominio general más amplio que pueda satisfacer las necesidades específicas y sus posibles relaciones.

El desarrollo de los *samplers* y sintetizadores en si mismo siguió un camino similar. El diseño original necesitaba cumplir con determinados objetivos planteados en la definición del dominio del problema. Pero este dominio fue demasiado restringido desde sus principios por razones prácticas. El control de las técnicas de síntesis y *sampleo* y la expresividad del sonido se ampliaron y fueron necesarias extensiones, sintetizadores diseñados específicamente para satisfacer estas nuevas necesidades (e.g. Lindemann 2001). La aplicación de estos nuevos recursos no solo no resulta estándar en la actualidad sino que es restrictivo de desarrollos particulares destinados a plataformas y aplicaciones específicas como los *plugins* VST27.

El protocolo OSC (Wright, Freed y Momeni 2003), es un estándar alternativo no

oficial en la industria adoptado por una enorme cantidad de desarrolladores de *software* puesto que soluciona muchos problemas inherentes al protocolo MIDI. Su diseño es mucho más general que el de su predecesor puesto que no estructura la información en abstracciones de alto nivel.

Otro caso paradigmático es el de las DAWs, o editores multipista, que en la actualidad de emplean para un uso distinto al cual fueron concebidas. La finalidad original de las DAW es la de grabar y mezclar pistas de audio en base al modelo de la consola de grabación. Su gran innovación son las líneas temporales que representan visualmente los *canales* de audio en el tiempo. Esta característica hace posibles los recursos de edición y secuenciación de audio, de manera complementaria a los editores, puesto que están orientadas a su organización temporal empleando edición no destructiva en contraste a los editores de audio que se basan en la premisa de la modificación del archivo como dato de entrada, el cual se procesa y da un resultado de salida. En la actualidad muchos editores multipista incorporan las funcionalidades originales de los editores de audio e incluso incorporan *canales* de secuenciación MIDI que se pueden manipular *samplers* y sintetizadores incorporados como *plugins*, generalmente representados visualmente como *pianoroll*. Incluso existen sistemas híbridos que incluyen la edición de partituras para facilitar la edición de canales MIDI (e.g *Nuendo28*, *Rosegarden29*) y editores de partituras que desarrollan ciertos recursos de los editores multipista (e.g. *NoteAbilityPro30*), como la secuenciación de archivos y las envolventes de control (automatizaciones). También existe otro tipo de software que hibrida las características de los secuenciadores MIDI con los editores multipista y el control en tiempo real (*AbletonLive31*, *LMMS32*). En si estos programas no añaden funcionalidades radicalmente distintas sino que difieren en la dirección sobre la cual se plantea la hibridación, estos programas priorizan la secuenciación adoptando capacidades de edición y procesamiento de audio en tiempo real.

Al emplear un editor multipista para la composición musical nos encontramos con un caso de adaptación de la finalidad original del *software*, una evolución que deriva del modo de empleo en relación a una finalidad específica. La música concreta, como corriente estética, se basa en la grabación, edición, procesamiento y mezcla de sonido como sus recursos fundamentales. Un editor multipista resulta ser la herramienta adecuada puesto que las abstracciones que emplea coinciden con las **abstracciones materiales y compositivas**. Aunque su finalidad y modo de empleo difieren sutilmente de los de la grabación y mezcla de un conjunto instrumental estos resultan adecuados a los recursos necesarios para la composición. Sin embargo, para muchos otros casos sus recursos y abstracciones son limitados.

Un editor multipista se basa en las abstracciones de *canal de audio*. Un *canal de*

audio representa simplemente un *stream* de datos conformado por la sucesión de archivos de audio posicionados en una línea temporal. El *canal* como abstracción no discrimina su contenido, el cual queda agrupado a un mismo nivel jerárquico manipulable como unidad. El **nivel de abstracción** inmediatamente inferior es una referencia a un archivo de audio (denominado *clip* en algunas aplicaciones) el cual puede diferenciarse como **entidad material** distinta de los otros componentes del *canal*. Aunque parezca trivial, estas abstracciones implican agrupamientos conceptuales y condicionan el modo de empleo **restringiendo** los recursos instrumentales específicos. En los entornos para síntesis de sonido la abstracción de *canal* suele estar disociada de la generación de sonido. Una unidad generadora o una *definición de instrumento* puede direccionar individualmente su salida a un *bus* sin necesidad de estar agrupada en un *canal de audio* (aunque puedan estarlo, no es un principio sistémico). Esto implica un mayor grado de individualidad de las **abstracciones instrumentales** en comparación con los archivos de audio en un editor multipista. Por ejemplo, un *canal de audio* puede ser enviado a un *canal de procesamiento* lo cual afecta a todo el conjunto de archivos contenidos por más que estén representando distintos materiales. En cambio, las *definiciones de instrumentos* pueden direccionar su salida individualmente a distintas unidades de procesamiento. El *canal* como abstracción implica un agrupamiento predefinido por la lógica del sistema, si bien es posible subsanar esta restricción organizando y automatizando el contenido de los canales, los recursos no son técnicamente idénticos.

Existen tecnologías que adoptan la noción de *canal* o *bus* de audio para poder direccionar la información sonora entre aplicaciones con la intención aplicar la concepción modular al *software* de audio de manera similar a como actúan los protocolos MIDI y OSC con las señales de control. *Jack33*, *Soundflower34* y *Rewire35* son tres tecnologías similares que actúan como interfaz de audio entre aplicaciones para distribuir los recursos de generación, manipulación y procesamiento de sonido. Estas herramientas están pensadas para poder realizar las tareas específicas de cada aplicación en combinación con las demás. La idea es que en lugar de tener todos los recursos centralizados en una sola aplicación estos se distribuyan entre aplicaciones específicas que realicen tareas más simples. La diferencia entre tener un diseño centralizado o distribuido es en parte una decisión práctica para el desarrollo de *software* puesto que de esta forma distintos equipos de desarrollo pueden trabajar en funcionalidades específicas que luego pueden ser integradas sin esfuerzo a un conjunto de herramientas existentes. Excepto *Jack* en *Linux*, ninguna de estas tecnologías tuvo gran difusión en la actualidad puesto que la tendencia comercial predominante es la de hibridar la mayor cantidad de prestaciones dentro de un único paquete o *suit* de *software* complementada mediante la tecnología de *plugins*. La centralización de los recursos tiene una finalidad productiva técnica. Distintos desarrollos

pueden emplear distintas convenciones para representar abstracciones similares lo que resultaría en un mayor grado de complejidad para el usuario final. Sin embargo, desde un punto de vista aparentemente utópico, se podría plantear que es posible la sistematización de un conjunto de recursos tecnológicos y abstracciones prevalecientes que podrían ser sistematizadas de manera estándar, lo cual facilitaría el desarrollo modular independiente del estado del arte de los recursos técnicos.

En relación a estos recursos y otros factores que influyen en el desarrollo de *software*, una de las finalidades de esta **Tesis** es contribuir a tal sistematización. En principio, analizando los recursos existentes que actúan como primitivas representacionales y sus posibles comportamientos. Si hay algo que queda claro en todo este desarrollo es que las **abstracciones instrumentales** adoptadas por distintos emprendimientos es variable según necesidad, ya sea en el desarrollo de las herramientas como en el modo de empleo que propician.

Nivel Base

Definición

Existen entornos de programación que buscan integrar la mayor cantidad de herramientas disponibles según enfoques específicos como es el caso de *OpenMusic* (Agon et al. 1999), *Nyquist* (Dannenberg 1997), *SuperCollider* (McCartney 2002), *PWGL* (Laurson et al. 2009), *ChucK* (Wang 2008), entre muchos otros. La principal característica que comparten todos estos entornos es su capacidad de diseñar las abstracciones *ad hoc* empleando lenguajes de programación de alto nivel. En la generación y manipulación de materiales musicales, la definición de las cualidades abstractas determina los recursos combinatorios disponibles y sus capacidades de transformación, los cuales son variables dentro de los límites impuestos por las abstracciones informáticas de bajo nivel.

Para la composición y el análisis de música electroacústica es necesario discernir el **nivel base** al que trabaja el compositor, el cual puede variar, entre compositores, de pieza en pieza o dentro de una misma pieza. El nivel base puede estar definido por factores conceptuales o imposiciones del *software*.

Como se anticipó en capítulos anteriores, el **nivel base** está en relación a lo que Bresson y Agon (2007) denominan *nivel simbólico*, que es el nivel de abstracción que adquiere significado como recurso organizativo o entidad musical para el compositor. Estos autores distinguen entre *nivel simbólico* y nivel *sub-simbólico* adjudicando solo al primero la incumbencia compositiva.

Según el desarrollo de estos autores, la representación del sonido a nivel de muestra

musical no es significativa de la estructura sonora. En cambio, las técnicas de síntesis, como la síntesis aditiva, sustractiva, el modelado espectral, las técnicas de análisis y resíntesis, la síntesis granular, la síntesis por FM y el modelado físico, proveen una representación estructurada del sonido. Por ejemplo, la síntesis aditiva consiste en la representación de los parciales y su evolución temporal, o los modelos espectrales que estructuran el sonido en bandas (*bins*) mediante el análisis de Fourier empleando la transformada de tiempo reducido (STFT, pos sus siglas en inglés) como herramienta de representación temporal del espectro, o la representación mecánica de los modelos instrumentales realizados mediante modelado físico. Sin embargo, consideran que ninguno de estos modelos asegura ningún significado composicional. Por estas razones adoptan la noción de *representación simbólica* que es la que emplean los compositores en el pensamiento compositivo.

> "*Generally speaking, a symbolic representation consists of the application of a set of signs used to substitute more or less complex realities in order to reduce, structure, and organize the information (Chazal, 1995). Symbols help memorize this information, but also to read, to understand or impart knowledge, to think about the represented object, or —and particularly in computer systems— to perform calculus on it.* " (Bresson y Agon op. cit.).

La *representación simbólica* a la que refieren es la que forma parte de los procesos mentales del compositor a diferencia de la *representación simbólica* propia de la máquina de computo (que involucra la arquitectura de bajo nivel sobre la cual se van construyendo símbolos a mayor nivel de abstracción). Por ejemplo, los *bits* "*que se corresponden con un fenómeno físico interpretado como valor binario*" (Bresson y Agon op. cit.) y que sirven para la representación numérica o de caracteres como símbolos de más alto nivel, no resultan representativos del pensamiento musical como expresan a continuación:

> "[...] *neither a bit nor a number will usually be considered as musically relevant symbols likely to be interpreted and handled by a composer. However, their combination creating a note-like representation (pitch value, amplitude, etc.) might constitute a musical representation. This representation is compact and structured enough to carry meaningful information and to be integrated mentally in compositional processes. It corresponds to a musical tradition and to a well-defined musical element.* " (Bresson y Agon op. cit.).

Es evidente que entre el código binario y la noción de nota musical hay una diferencia simbólica abismal. Sin embargo, en la práctica hay ejemplos de creaciones sonoras realizadas mediante la manipulación de *bits*, sin emplear ningún otro tipo de abstracción de alto nivel (véase el ejemplo de minimalismo computacional en el **Capítulo 6**). El problema de la definición de estos autores es que no especifica precisamente los

principios que hacen que un elemento pase a formar parte de un determinado *nivel simbólico* puesto que no se puede considerar que una abstracción deba ser mas compleja para adquirir dicho estatus. Una nota musical puede ser representada mediante guarismos sin expresar ni la duración ni la amplitud ni ningún otro parámetro e incluso referir a la sensación de una altura determinada. Una sucesión de alturas, una disposición armónica o un *pitch-class set* puede ser representados mediante cifras cuyas relaciones refieren a aspectos específicos de distintas estructuras musicales. Estas cualidades y sus relaciones pueden ser la abstracción material primitiva que el compositor emplea para la elaboración musical, especialmente en el desarrollo de **relaciones abstractas** que luego serán realizadas en un medio instrumental (ver **Capítulo 3**).

La noción de *nivel simbólico* musicalmente significativo a la que Bresson y Agon se refieren depende de un contexto específico:

> "*Indeed, the notion of symbolic representation for music composition is often related to the traditional musical notation system and to the corresponding score representation. This representation corresponds to instrumental practice and to a formal conception of musical sound divided in timed events, essentially defined by a pitch (i.e. notes) organized (particularly in time) by compound structures (chords, voices, etc.) The musical objects and structures in the score are symbolic representations for they carry a musical sense and are likely to be handled (read, understood, written) in a given (musical) context (e.g. composition or music interpretation.)*" (Bresson y Agon op. cit.).

El contexto en el cual se produce el *nivel simbólico* para estos autores es un sistema complejo desarrollado en la tradición musical. Influyen los recursos instrumentales y las prácticas interpretativas, la notación musical y la concepción formal del sonido como eventos temporales (noción de objeto-nota) que organizados en el tiempo generan estructuras compuestas. De esto se desprende que el *nivel simbólico* se define por la práctica (tradicional), los símbolos son elementos discretos que tienen significado dentro de un contexto complejo de relaciones.

Para esta **Tesis**, el problema argumentativo de esta definición es que no se considera a la máquina de cómputo como un instrumento, puesto que no está inserto dentro de la tradición musical, y descarta las abstracciones de bajo nivel relegándolas al nivel *sub-simbólico*:

> "*On the contrary, the numbers that constitute a digital waveform in a straightforward representation of sound are not considered as symbolic elements. Individually, these elements do not hold any specific meaning for a composer and can hardly be put in relation with one another in the compositional process. We could relate these representations to the subsymbolic domain discussed in (Leman, 1993; Camurri, 1990) if (with some simplification to the original concept) we consider that this domain brings together processes and representations that carry relevant meanings for the bare computer processing of the data, while symbolic*

representations holds meaning for the system user (e.g. a composer)." (Bresson y Agon op. cit.).

Desde sus inicios, la informática actual no fue, por razones cronológicas obvias, un medio instrumental inserto en la tradición musical, aunque existen ejemplos históricos de técnicas y recursos considerados computacionalmente como *procesos formales en música* (Roads 1995). Pero las concepciones informáticas tienden a definir el dominio del problema de manera restringida a la tradición, dificultando la conceptualización de las capacidades del medio informático. Un nuevo medio desarrolla su tradición con el tiempo. La exploración y una conceptualización abierta propician el descubrimiento de nuevas técnicas, como de hecho ha sucedido con los modelos de síntesis a los que estos autores refieren. Las orquestas de *laptops* (Trueman et. at. 2006) (Bukvic et. al. 2010) integran paradigmas tradicionales con nuevos recursos instrumentales, los principios organizativos son claros, pero los recursos disponibles son distintos y así su modo de empleo (la técnica instrumental). Los sistemas de difusión, por dar otro ejemplo, son otro factor que influencia el desarrollo, no solo del espacio sonoro y los conjuntos instrumentales sino de las técnicas compositivas e interpretativas.

Bresson y Agon (op. cit.) abogan por la noción de modelado sonoro/compositivo la cual no es muy diferente del planteo de esta **Tesis**. Consideran que la composición sonora se divide en niveles de abstracción que pueden variar según el enfoque del compositor.

"*First we must consider sound as the intentional object of a compositional process, and therefore as a structure likely to be represented by a program. Through the general structuring of the programs via the abstractions or hierarchical constructs, compositional models are created, made of organized component and structural aspects and corresponding to particular situations or compositional approaches. In this context,* ***sound is therefore not considered only from the acoustic signal production point of view, but rather as the product of a compositional process that aims to create this signal****. By determining particular components and structural contents of these models, certain aspects of the programs are put forward of the representation, thus establishing implicit symbolic abstraction levels within this representation.*" (Bresson y Agon op. cit., énfasis agregado por el autor de esta **Tesis**).

Un modelo sonoro puede referir tanto a las técnicas de síntesis empleadas como a su organización y relación con otras estructuras que modelizan el "*proceso compositivo*". Se tiende a integrar todos los aspectos de la creación sonora como elementos representativos del sonido mismo de manera similar a como actúa el contexto en la música tradicional. Esta concepción deriva de las nociones de "*concepto sonoro*" (Eckel y González-Arroyo 1994), referida más adelante en esta **Tesis** (**Capítulo 6**) y de "*modelo sonoro*" de Eckel (1993) que "*permite la producción de sonido y su representación*

formal".

Puesto que la terminología puede resultar confusa, es necesario aclara que la noción de *proceso compositivo* expuesta por Bresson y Agon (op. cit.) no está relacionada con la de **proceso** como material sonoro desarrollado en esta **Tesis** (**Capítulo 3**). El proceso *compositivo refiere* a la labor creativa del compositor y la formalización de los elementos que constituyen su discurso y representan las estructuras sonoras mediante modelos (sistema de significados y relaciones) dentro del campo de la composición asistida. Es importante destacar que el concepto de proceso compositivo puede actuar como reemplazo del sistema de relaciones que garantiza una tradición musical. Mediante el desarrollo de modelos compositivos se pueden sistematizar tanto los recursos y las técnicas instrumentales como la notación de las mismas adaptándolos a nuevos conceptos sonoros. Pero para un desarrollo de tales sistemas no se puede disociar la naturaleza computacional del ordenador como instrumento musical que impone determinados patrones de desarrollo. Esto refiere a la técnica específica que deriva de la constitución casi física, predefinida, del nuevo medio instrumental, el cual se complementa con las nuevas concepciones abstractas que posibilita. Esto es sin más la relación dialéctica entre teoría y praxis artística que resurge en los cambios del medio de producción sonora.

De manera más simple, Winsor y DeLisa (1991) diferencian dos enfoques a la hora de abordar la problemática de la música por computadoras en relación a los lenguajes de programación y la representación de sus elementos como entidades musicales o técnicas: el modelado en base a la representación simbólica/gramatical y el modelado en base a la representación procedimental/formativa. A grandes rasgos, el primero se encarga de representar las estructuras definidas por la retórica de un sistema musical predefinido (cuyos componentes podrían ser: notas, acordes, las nociones de melodía y armonía, etc.), mientras que el segundo tiende a dejar de lado las estructuras simbólicas para enfocarse en las cualidades de la computadora como herramienta de cómputo empleada por el compositor para realizar sus tareas (Winsor y Delisa op. cit.).

Esta distinción da cuenta del problema de la conjunción de dominios que se produce entre la música y la informática. La representación simbólica/gramatical estaría más ligada al concepto de *nivel simbólico* mientras que el enfoque procedimental/formativo está en relación al nivel *sub-simbólico* de Bresson y Agon. El supuesto teórico de esta **Tesis** con respecto a este tema es que no existen tales disociaciones entre el dominio informático y musical. Al emplear la computadora como instrumento musical esta pasa a ser parte de un dominio más amplio que integra ambas disciplinas.

Es fácil relacionar la noción procedimental/formativa a los procesos compositivos y los procedimientos musicales, los cuales pueden referir a la puesta en forma de las

nociones compositivas, empelando medios informáticos, y el desarrollo de los estructuras musicales que luego ascienden al *nivel simbólico*. La asignación de estatus *simbólico* de las estructuras de datos básicas (ver **Capítulo 6**), que surgen de la representación informática como estructuras *sub- simbólicas* es en gran medida arbitraria, se define por la relevancia que le otorga una **restricción voluntaria** y el **nivel de abstracción al cual se manipulan** las entidades como elementos sonoros y musicales.

Puesto que la noción de *nivel simbólico* implica una conceptualización abstracta y relativamente arbitraria de los recursos informáticos como medio instrumental, aunque no se descarta su validez, para el desarrollo de esta **Tesis** se adopta la noción de **nivel base**.

Definición: *El **nivel base** puede ser cualquier elemento que represente un **objeto o proceso musical** dependiente del recurso o **abstracción instrumental** empleado que define sus capacidades de manipulación, combinación y transformación.*

El **nivel base** puede referir entonces: al concepto de nota musical, empleado extensamente en los análisis de música tradicional por su adecuación a la notación tradición y la percepción de los recursos instrumentales característicos relacionados con el **modo de acción** instrumental (**abstracción instrumental** empleada); un **objeto** o **proceso**, simple o compuesto, considerado como material; un **agrupamiento** gestaltico (Tenney 1980) (Deutsch 2013); una entidad formal o estructural de más alto nivel como motivo, frase, acorde, textura; puede incluso ser una obra musical completa (el caso de un DJ); técnicamente puede ser un archivo de audio, una definición de instrumento, un evento o un conjunto de eventos en un lenguaje de partitura o una unidad generadora.

El nivel de abstracción que define el **nivel base** no está predefinido por el medio y varía según las concepciones compositivas y analíticas. Lo que define al nivel base es su empleo como elemento combinatorio/compositivo dentro de un discurso o técnica musical. La técnica se relaciona así con la teoría por medio del entendimiento de las herramientas y los recursos y abstracciones que posibilitan.

Para la música concreta, el nivel base está dado por los objetos sonoros materializados técnicamente como fragmentos de cinta o archivos de audio. Es posible cambiar el **nivel base** y emplear algoritmos de procesamiento que transformen internamente un objeto sonoro, pero la técnica compositiva se basa principalmente en la combinación de estas entidades discretas. Como ejemplos paradigmáticos, un *sampler* relativamente simple puede manipular objetos sonoros haciendo un *mapeo* del timbre y la organización de alturas de un instrumento real o simplemente definiendo un conjunto de objetos sonoros asignados a una acción (un evento que dispara un archivo de audio). El **nivel de abstracción base** es la nota musical como objeto. En un sintetizador analógico, es posible manipular los parámetros de evolución internos del sonido en tiempo real, en

este caso coexisten las **abstracciones paramétricas** con la noción de nota musical, el nivel base abarca ambos tipos de control sobre la resultante sonora. El cambio gradual en la frecuencia de corte de un filtro resonante, recurso muy común en la música electrónica, puede actuar delimitado temporalmente por la nota mediante el empleo de envolventes o puede actuar sobre un conjunto de notas e incluso una textura musical compleja representada como señal de audio. La abstracción que define al filtro como recurso instrumental puede actuar a distintas escalas temporales y en relación a otros **niveles base**.

El ***nivel simbólico base*** se elige al seleccionar o diseñar una **abstracción instrumental** o un conjunto de **relaciones abstractas**. Una definición de instrumento puede actuar como **nivel base** si se la manipula externamente como elemento discreto pero también puede estar representando un **material musical** complejo que incluso puede estar actuando formalmente como una sección musical (ver **Capítulo 6**). En este último caso, una unidad generadora o una envolvente dinámica pueden actuar como el elemento que define el comportamiento sonoro o musical pretendido.

El **nivel base** empleado técnicamente en una composición no necesariamente coincide con el resultado percibido, el cual puede generar otro *nivel simbólico* perceptivamente, e.g. mediante la abstracción de notas musicales se pueden generar transiciones/trasnformacions tímbricas entre sonidos empleando la técnica del *crossfade*, la cual se puede percibir como una mutación continua; una abstracción similar se puede lograr mediante al análisis y resíntesis de sonidos y diversas manipulaciones espectrales. Dos técnicas representadas de manera totalmente distinta pueden generar resultados perceptivos simbólicamente equivalentes. La percepción del fenómeno resultante puede incluso diferir de la concepción u organización compositiva como se desprende de los experimentos de Deutsch (1974)36.

A continuación se enumeran las características que definen el **nivel base**:

- Es independiente del **nivel de abstracción**.
- Se define recíprocamente junto con las **abstracciones materiales e instrumentales**.
- Define los elementos combinables (como elementos manipulables).
- Puede ser variable a lo largo de una pieza.
- Se relaciona y desprende de las **abstracciones instrumentales** y **compositivas**.
- No necesariamente coincide con las abstracciones conceptualizadas perceptivamente.

Relación con los procesos compositivos

El ordenador puede ser empleado simplemente como herramienta de cálculo (auxiliar) de las **relaciones abstractas** que facilite su ordenamiento para una posterior **realización**. Durante la etapa de **realización**, el ordenador puede ser utilizado para organizar las **abstracciones compositivas**, tanto de recursos informáticos (e.g. composición algorítmica y síntesis de sonido) como tradicionales (e.g. instrumentos acústicos y edición). En esta etapa, la composición algorítmica se considera inmanentemente computacional, y por lo tanto, se relaciona con la representación de **materiales lógicos y concretos** (puesto que se manifiestan sus cualidades temporales). Implica la definición de materiales representados explícitamente como entidades computacionales, dentro de los cuales, la síntesis de sonido puede actuar a nivel las **abstracción instrumental**. Por lo tanto, la composición algorítmica, que genera el material sonoro dentro del mismo medio instrumental, abarca en conjunto las **relaciones abstractas**, las **abstracciones instrumentales** y los **materiales lógicos y concretos**.

A cada etapa del desarrollo compositivo, el **nivel base** es diferente. Para la organización de **relaciones abstractas**, el nivel base se sitúa en relación a estructuras musicales atemporales como recurso de coherencia estructural. En cambio, en la composición algorítmica como conjunto, el **nivel base** se puede situar a cualquier nivel entre las **relaciones abstractas** y las **abstracciones materiales** a cualquier escala temporal. A nivel de síntesis de sonido, el **nivel base** se sitúa en los elementos que componen las **abstracciones instrumentales**.

En los entornos de programación de propósito general, es posible variar el **nivel de abstracción base** arbitrariamente puesto que las estructuras están unificadas en un mismo plano representacional. Sin embargo, esto puede depender tanto de las **restricciones impuestas** por el *software* como de la conceptualización propia de cada compositor o los requerimientos del tipo de manipulación de las estructuras musicales que se pretenda lograr. Estos dos últimos factores pueden estar relacionados pero no ser empleados de manera sistemática o consciente (véase más adelante las distintas formas de encapsular unidades generadoras en relaciona a las definiciones de instrumento, el material resultante y sus recursos de manipulación, **Capítulo 6**). Determinar el **nivel base** propuesto por el entorno o por el compositor en relación a la organización de los recursos, es fundamental para comprender por qué determinadas estructuras musicales se comportan de manera específica. Cuando se emplean aplicaciones informáticas, como los editores de audio o los editores multipista, para la composición musical, se suelen emplear **niveles base** superiores a los empleados en los lenguajes de programación. Esto es lo que genera **restricciones** en la manipulación de las entidades que intervienen y en la

concepción de **material musical** que el compositor emplea. Las consecuencias pueden observarse tanto a nivel técnico como estético.

Las computadoras pueden ayudar al compositor en su labor como auxilio en el cálculo, o puede actuar como el instrumento que genera el material musical resultante mediante procesos algorítmicos. La realización materiales complejos estocásticos o puntillistas, por ejemplo en Xenakis y Boulez, pueden emplear procedimientos que se logran un resultado similar de distintas maneras. El **nivel base** de abstracción musical al que trabaja Boulez en *El martillo sin dueño* (Boulez 1957) es el de nota musical. Esto es una elección que se relaciona tanto con el medio de producción instrumental como con la concepción compositiva. Si su desarrollo fuera computacional, Boulez necesitaría que el *software* empleado pueda actuar solamente en relación a ese nivel de abstracción, las abstracciones de nota y *canal* (o instrumento) son suficientes para lograr el resultado deseado. Por el contrario, Xenakis, en *Concret PH* y *S.709* (Xenakis 1997 y 1971) manipula abstracciones que generan textura y movimiento melódico para los cuales la abstracción de nota e instrumento convencionales no tienen el mismo sentido y cuyos recursos de control son diferentes. El **nivel de abstracción base** en estos casos puede ser la textura como objeto sonoro (en *Concret PH*) y el *stream* melódico (en *S.709*) y las envolventes de control. Sin embargo, los resultados perceptivos pueden ser similares y pueden ser variables según el oyente, el **nivel de abstracción base** es conceptual, no perceptual.

Conceptualmente, los espectralistas (Fineberg 2000) conciben la textura musical resultante como el desarrollo de los procesos internos de un sonido. Para su realización orquestal, sin embargo, es necesario recurrir a la abstracción de nota musical como **nivel base**, debido a que el medio de producción así lo condiciona, y componer el sonido resultante. La concepción resultante es un objeto (a escala *meso temporal*) que se desarrolla técnicamente, mediante la relación de notas musicales (a escala de *objeto sonoro*), la altura en relación al tiempo. El recurso instrumental no coincide con la concepción compositiva. Sin embargo, esta correspondencia se podría lograr mediante el empleo de medios informáticos. Esto no es casual, ya que los espectralistas basan su estilo en conocimientos sobre el timbre musical desarrollados mediante la informática.

CAPÍTULO 5: Análisis de los Recursos de

Representación Existentes

Introducción

Para el estudio de los recursos de representación disponibles en la actualidad, se procederá a analizar en detalle el entorno *SuperCollider* (McCartney 2002). Es relativamente fácil de comprobar, para el lector entendido en distintos entornos de programación, que los recursos que se exponen y analizan a continuación existen, tal vez con variaciones en la interfaz o en la especificidad o priorización de distintas funcionalidades, en distintos entornos. Solo se mencionaran los casos pertenecientes a otros entornos que aporten herramientas o concepciones que no estén disponibles o que difieran substancialmente de las encontradas en *SuperCollider* y que representen tipos de recursos relevantes para el análisis.

SuperCollider como Caso de Estudio

Este entorno es generalmente descripto como lenguaje de programación orientado a síntesis de sonido en tiempo real y composición algorítmica. En palabras de su creador:

> "*Es un lenguaje para describir **procesos sonoros**.* […] *permite representar conceptos musicales como **objetos**, trasformarlos por medio de funciones y métodos, componer transformaciones contenidas en **bloques de construcción** de alto nivel, y diseñar interacciones para manipular música en tiempo real, desde las estructuras de más alto nivel de una pieza descendiendo hasta el nivel de la forma de onda*" (McCartney 2011, traducción y negritas por el autor de esta **Tesis**)

Con respecto a la ingeniería de software, el entorno está deseñado, a partir de la versión 3, con arquitectura de cliente-servidor (McCartney 2002). El servidor se encarga de realizar los procesos de síntesis y el cliente de representar y manipular las abstracciones, tanto del servidor como de los elementos musicales y sonoros, mediante un lenguaje de programación de propósito general específicamente diseñado para tal fin. En la actualidad existen dos implementaciones alternativas del servidor, por un lado está la original, *scsynth*, diseñada por James McCartney, y *supernova*, diseñada por Tim Blechmann en base a la primera, que agrega la capacidad de realizar computación paralela empleando múltiples procesadores mediante la técnica SIMD (Blechmann 2010, 2011). El lenguaje de programación se llama *sclang*, es la aplicación informática por defecto que actúa como cliente encargado de generar las instrucciones necesarias para controlar el servidor mediante abstracciones de más alto nivel. Para la comunicación entre el cliente y el

servidor se emplea el protocolo de comunicación OSC (Wright, Freed y Momeni 2003). Debido a que la aplicación de síntesis actúa como proceso independiente que se comunica mediante un protocolo estándar, han surgido diversas implementaciones de aplicaciones cliente, generalmente en distintos lenguajes de programación de propósito general, con el propósito de explorar las capacidades expresivas de distintos lenguajes, por ejemplo: *ScalaCollider*37 (empleando el lenguaje *Scala), Overtone*38 (empleando el lenguaje *Clojure*), *supercollider.js*39 (empleando el lenguaje *JavaScript*), Scrubi40 (empleando el lenguaje *Ruby*) y *hsc341* (empleando el lenguaje *Haskell*).

La arquitectura de este entorno es de particular interés para esta **Tesis** puesto que manifiesta de forma explícita la lógica del programa de síntesis y la lógica computacional necesaria para controlarla. El compositor, al emplear este entorno, debe tener conocimiento de su funcionamiento como instrumento informático, de la misma manera que se tienen conocimientos de los recursos sonoros y las técnicas de ejecución de un instrumento acústico. La técnica interpretativa está ligada a la manipulación expresiva de recursos abstractos, codificados y transmitidos en forma de programas. *SuperCollider*, a diferencia de otros entornos como *Pure Data* (Puckette 1996), *Csound* (Vercoe 1986) o *ChucK* (Wang 2008), hace explícita físicamente, mediante distintos espacios de memoria, la relación entre las **abstracciones compositivas e instrumentales** en la separación de la aplicación de síntesis y la aplicación de control, aunque esto no es más que el resultado de su diseño específico, expresa claramente distintos **niveles de abstracción**. Sobre las abstracciones básicas se construyen luego las abstracciones de más alto nivel que son empleadas para representar la lógica musical y las capacidades de los recursos instrumentales en relación a estas. Aunque no es necesario interactuar directamente, a bajo nivel, con el programa de síntesis, es posible si se lo desea. De esta manera se posibilita el desarrollo de distintos entornos y lenguajes de programación sobre una base instrumental común como se mencionó anteriormente.

Entornos como *ChucK*, *Pure Data* y *Csound*, encapsulan ciertas abstracciones de más bajo nivel de distinta manera. Por ejemplo, los *buses* que conectan unidades generadoras y definiciones de instrumentos, son abstraídos metafóricamente mediante operadores en *ChucK* o “cables” en *Pure Data* y mediante variables en *Csound*. Aunque las abstracciones sean distintas cumplen la misma función. En cambio, en *SuperCollider* un *bus* de audio o de control se abstrae como representación directa de un *bus* de datos y actúa como **abstracción instrumental**. *SuperCollider* es muy rico en cuanto a la variedad de formas en las cuales se pueden expresar los recursos instrumentales, lo cual posibilita un cierto grade de flexibilidad que, como contrapartida, produce un cierto nivel de complejidad para el usuario. Esto no le quita generalidad al diseño de los otros entornos, las abstracciones básicas se definen mediante metáforas que pueden afectar de distintas

maneras los recursos de control, no necesariamente de manera restrictiva, en base a la premisa de simplificación de las acciones para el usuario final. En la actualidad no existen estándares de alto nivel, cada lenguaje expresa los recursos de manera sutilmente distinta según distintos desarrollos conceptuales, facilitando cierto tipo de comportamientos pero afectando la generalidad o practicidad de otros.

Un *bus* de audio en *Pure Data* no es expresado como entidad independiente. El cable como metáfora del mundo físico queda sujeto a las abstracciones visuales que conecta y no puede ser manipulado algorítmicamente. Aunque existan distintos métodos para organizar abstracciones equivalentes en cada uno de los entornos, o incluso se puedan desarrollar extensiones o mejoras que provean recursos faltantes, el modo de empleo para el cual fue diseñado originalmente cada entorno condiciona la flexibilidad y la facilidad de uso de los distintos recursos. En *SuperCollider* un *bus* es la abstracción de una abstracción informática, diseñada con posibilidades extendidas en comparación a la metáfora del cable. Puesto que es la abstracción de un espacio de memoria compartido se puede manipular algorítmicamente como tal y no está condicionado por las restricciones de niveles de abstracción superiores.

En *ChucK* es más fácil interconectar abstracciones de más alto nivel a un mismo nivel de jerarquía estructural, puesto que, por su diseño, todo el procesamiento se realiza de manera unificada y sincrónica, similar al diseño de *SuperCollider* en su versión 2 (McCartney 1996), donde el lenguaje de programación estaba integrado con los algoritmos de síntesis en un mismo espacio de memoria. En *ChucK*, tanto las estructuras de control de la programación como la lógica de procesamiento de los algoritmos de síntesis están integradas, lo cual facilita la manipulación a resolución de muestra de audio de forma clara y sencilla desde el lenguaje de alto nivel. Sin embargo, este entorno implementa también la misma lógica de interconexión de unidades generadoras que en los demás entornos. En *ChucK*, las unidades de síntesis están encapsuladas en abstracciones de más alto nivel de las cuales se puede acceder a sus valores instantáneos a resolución de muestra. En cambio, en entornos como *SuperCollider*, *Pure Data* y *Csound*, se suelen emplear distintos tipos de unidades generadoras que definen recursos de manipulación a nivel de muestra pero la interacción de estas estructuras depende del período de control y la lógica de la composición de las *definiciones de instrumentos*.

La implementación del período de control es un recurso restrictivo introducido por Vercoe en el programa *Csound*42 (Roads 1995). Reduce la resolución temporal de las señales de control para ahorrar recursos computacionales. Su invención es positiva puesto disminuye el tiempo de cómputo de manera significativa sin degradar la calidad del sonido resultante de manera crítica, si es usado debidamente. Perceptivamente, las señales de control, en especial las envolventes, no necesitan de altas resoluciones temporales para

generar resultados correctos. El período de control consiste en definir un período de procesamiento mayor a la frecuencia de audio e implica el procesamiento en bloques de las señales a frecuencia de audio. En *Csound* el período de control se manipula mediante la frecuencia de control (*control rate*), en *SuperCollider* se lo denomina bloque de control (*control block*) y se lo suele abreviar con las letras "*kr*". En ambos entornos el período de control se puede igualar al período de muestreo para obtener la máxima resolución a costa de una menor eficiencia computacional. Cuando la capacidad de cómputo estándar era restringida, este procedimiento agilizó el tiempo de espera entre los datos de entrada y los datos de salida, luego, al incrementarse la capacidad de las computadoras, se empleó como recurso para la implementación de aplicaciones en tiempo real. El procesamiento en bloques de las señales de audio implica una **restricción**, no demasiado significativa, para el control externo de los algoritmos de síntesis y la manipulación de señales. Además, esta **restricción** hace sutilmente más compleja la manipulación algorítmica para casos específicos. Esto es lo que resuelve el entorno *ChucK* mediante la sincronía a nivel de muestra de todos los objetos y estructuras de control en demérito de la eficiencia (Wang 2003, 2008).

Sin embargo, en *ChucK*, a la hora de representar abstracciones que existen y se objetivan a mayor escala temporal, por ejemplo las envolventes, la manipulación a nivel de muestra resulta más compleja (requiere la organización de una mayor cantidad de recursos) que la provista por *unidades generadoras* específicas que controlan su duración global de manera automática. El equilibrio y la interacción entre distintos **niveles de abstracción** y **escalas temporales** es un problema que, de ser posible de solucionar, aún no ha sido resuelto. El período de control es notablemente útil en parte por su correspondencia con la **escala temporal** de interés en las señales que representa. A continuación se explicarán brevemente los recursos técnicos instrumentales, que actúan como **abstracciones instrumentales**, propios de *SuperCollider* desde los cuales se puede trazar un paralelo o comparación con otros entornos. Pese a su organización particular en relación a esta aplicación, los recursos expuestos son en sí generales y derivan de las técnicas informáticas de la práctica actual.

Aspectos del Servidor

El servidor es un programa por línea de comandos que define la lógica arquitectónica de la máquina de síntesis y sus recursos instrumentales, los cuales pueden ser manipulados dinámicamente desde un cliente. Según su documentación43 se compone principalmente de los siguientes elementos.

Unidades generadoras

Una *unidad generadora* es un algoritmo de síntesis entendido como elemento unitario en la composición de señales. Puede ser un oscilador de acceso a tablas, un lector de *buffers*, una unidad de síntesis granular, un filtro etc. Las unidades generadoras están implementadas como *plugins* que se cargan al iniciar el servidor (no es posible cargarlas en tiempo de ejecución). Su programación se realiza en lenguaje C/C++ mediante una API estándar, por lo que solo pueden ser programadas y compiladas como extensiones. Una *unidad generadora* puede recibir señales de audio, de control o de inicialización y generar señales de audio o de control. Las *unidades generadoras* no se pueden emplear por sí solas en la máquina de síntesis, sino que deben estar contenidas dentro de un *nodo* de síntesis, el cual se encarga de recibir la información de los *buses* de entrada, interconectar las *unidades generadoras* que realizan el procesamiento y escribir el resultado en los *buses* de salida.

Definición de instrumento (*SynthDef*)

Es la descripción en *bytecode*, entendible por el servidor, de las interconexiones necesarias entre unidades generadoras para crear un *nodo de síntesis*. Es el equivalente a una definición de *instrumento* en *Csound* y otros entornos similares y, en esta **Tesis**, se la nombra de manera equivalente. Una vez que se carga una definición de instrumento, esta es utilizada para instanciar tantos *nodos de síntesis* como se requiera. Las definiciones de instrumento pueden ser cargadas en el servidor en tiempo de ejecución o pueden ser almacenadas en disco para ser cargadas al iniciar la aplicación. A diferencia de las *unidades generadoras*, las *SynthDef* son elementos que se programan y manipulan de forma dinámica en tiempo real.

Nodo de síntesis

Un *nodo* es un elemento direccionable dentro del árbol de procesamiento de la máquina de síntesis. Estos pueden ser de dos tipos: *grupo* y *synth* (se describen a continuación). La organización en forma de árbol define una estructura jerárquica de procesos de síntesis que se encarga de solucionar el problema del orden de ejecución. En entornos como *Pure Data* y *ChucK*, el orden de ejecución está dado por la interconexión realizada directamente entre unidades generadoras, en el caso del primero, este orden puede depender de qué cable se conectó manualmente primero a una determinada entrada o salida lo que puede llevar a errores difíciles de depurar. *SuperCollider* emplea la

organización en forma de árbol de *definiciones de síntesis* lo cual actúa a nivel organizativo de los algoritmos. Los *nodos de síntesis* no necesariamente deben estar en una única relación jerárquica predeterminada. Puesto que se comunican mediante *buses* globales de audio o control, es posible interconectar nodos de distintas ramas. El árbol de síntesis sirve para ordenar eficientemente el procesamiento de señales y para garantizar el orden de ejecución entre los componentes.

Grupo

Un *grupo* es un tipo de *nodo* que contiene otros *nodos* en una lista ligada. Es el elemento principal empleado para organizar los algoritmos de síntesis. Este tipo de *nodo* puede contener tanto s*ynths* como otros *grupos*. Los *nodos* pueden ser agregados en el servidor a la cabeza o a la cola de un *grupo* o en relación a otros *nodos*. Los *nodos* que están antes en la lista se ejecutan antes que los demás en el mismo nivel jerárquico, los *nodos* contenidos en sub-grupos de ejecutan en el orden del *grupo* que los contiene. Los nodos contenidos dentro de un *grupo* pueden ser controlados de manera conjunta, incluso, si comparten los mismos argumentos de entrada, distintas *synth* pueden ser controladas en conjunto. Al iniciar el servidor se crea un *nodo raíz* a partir del cual se pueden iniciar todos los demás nodos.

Esta organización jerárquica no está relacionada con la estructura musical sino con las cualidades instrumentales del procesamiento de señales específico de este entorno. Sin embargo, sus características facilitan determinados modos de uso aunque, en general, resulta ser extremadamente versátil. La organización jerárquica de los nodos no se debe confundir con la organización jerárquica de estructuras musicales las cuales se pueden representar, de diferentes maneras, del lado del lenguaje.

Synth

Un *synth* es un *nodo de síntesis* creado a partir de una *definición de instrumento* (*SynthDef*). Consiste en una organización de *unidades generadoras* que actúan de manera conjunta. Los *synth*, al igual que los *grupos*, se pueden direccionar y controlar dinámicamente mediante comandos al servidor. Contienen unidades generadoras especiales que actúan como entradas de parámetros de control y pueden recibir y enviar señales de audio o de control por medio de los *buses* globales.

Bus

Un *bus* es el elemento en el cual se escribe y del cual se lee la información del procesamiento de señales. Los *buses* en el servidor pueden ser de audio ("*ar*") o de control ("*kr*") y actúan como los cables en *Pure Data*, las variables globales en *Csound* o los operadores de conexión en *ChucK*. En otros entornos como *Nyquist* (Dannenberg 1997), e incluso dentro de las definiciones de instrumento en *SuperCollider*, los *buses* suelen ser abstracciones encapsuladas a las cuales no se tiene acceso sino que se emplean como parámetros.

Los *buses* en *SuperCollider* son globales y están numerados a partir del 0. Los primeros buses de audio se reservan para las salidas y entradas físicas de la computadora, mientras que los restantes se pueden emplear para interconectar procesos de síntesis. Esto provee la ventaja de que, al no conectar estáticamente los *nodos* entre sí, estos no necesitan información sobre los demás y pueden leer o escribir la información necesaria para cualquier otro tipo de *synth* o las salidas del *hardware*. Los *buses* de control son solo internos (no se comunican con el *hardware*) y se comportan de la misma manera que los *buses* de audio excepto que la información es transmitida a frecuencia de control. Las señales de control pueden ser interpoladas internamente por las unidades generadoras para obtener transiciones más suaves.

Buffer

Un *buffer* es un elemento común a todos los entornos de audio. Consiste en un espacio de memoria consecutiva en la cual se almacenan señales de audio. Estos pueden ser empleados para la lectura de archivos o tablas de audio a usar en distintos procedimientos de síntesis. La memoria de los *buffers* se suele reservar dinámicamente con una capacidad razonable. De todos los componentes de la máquina de síntesis, este es el único elemento que podría estar representando un **material concreto** (ver **Capítulo 3**), en oposición a la **interfaz instrumental** representada por los demás. Pero el *buffer*, a este nivel de abstracción, es entendido como el **elemento instrumental** que posibilita la representación de materiales concretos en los procesos de síntesis a niveles más altos.

Aspectos del Cliente

El principal cliente del servidor de *SuperCollider* es el intérprete del lenguaje *sclang*. El intérprete es la aplicación a la que se integra el lenguaje, compila el código de programación y envía los comandos OSC al servidor. En general se refiere al *intérprete* y al *lenguaje* simplemente como el lenguaje *sclang*.

sclang un lenguaje de programación completo de alto nivel, orientado a objetos y dinámicamente tipado. Mediante la programación orientada a objetos, se abstraen los elementos constitutivos del servidor (incluido este), como son los *buses*, *buffers*, *nodos*, *synth* y *grupos*. El lenguaje de programación se encarga de traducir los objetos representados en comandos OSC para poder enviarlos al servidor y controlarlo. Esto proporcionar una interfaz de programación más simple y versátil comparada con la escritura manual de comandos OSC los cuales, sin embargo, pueden ser empleados directamente. Las distintas abstracciones interactúan mediante los principios de la programación orientada a objetos y pueden ser combinados y programados por completo.

Relación intérprete / servidor

La principal desventaja del diseño cliente-servidor consiste en que la transmisión de mensajes OSC agrega una capa de abstracción intermedia que, aunque se puede ocultar empleando la interfaz de la programación orientada a objetos, trae aparejado cierto grado de complejidad. Las clases en *SuperCollider* no son una representación directa, en cuanto a espacio de memoria, de las funciones del servidor, sino abstracciones que representan los componentes del servidor como objetos programables. Los objetos que representan los recursos del servidor, internamente generan los mensajes OSC necesarios y los envían como instrucciones a través un protocolo de red.

La transmisión de mensajes implica que las acciones programadas en el lenguaje pueden no ser sincrónicas. Esta es la característica que agrega mayor grado de complejidad a la programación puesto que son necesarios recursos adicionales para realizar y sincronizar las acciones. Estos recursos generan complejidad en los algoritmos entendidos como medio de representación musical. Por ejemplo, para cargar un archivo de audio y reproducirlo, es necesario emplear una abstracción que represente el *buffer* que lo contiene en memoria. La clase *Buffer* es frecuentemente utilizada para generar las instrucciones necesarias para que el servidor vaya a buscar y cargue en memoria un archivo de audio. La información del archivo de audio reside en el servidor, mientras que la clase *Buffer* mantiene una referencia al *buffer* del servidor como estructura de datos en el cliente. Es decir que el lenguaje de programación mantiene una referencia al recurso instrumental del servidor que contiene la información de audio.

Las operaciones que implican lectura de archivos y reserva dinámica de memoria suelen tardar más tiempo que las operaciones aritméticas realizadas en el procesador. El intérprete, mediante la clase *Buffer*, simplemente envía las instrucciones necesarias, lo cual requiere menor tiempo de procesamiento. Una vez que se enviaron las instrucciones, el flujo de control del programa en *sclang* puede continuar en paralelo mientras el

servidor abre el archivo y carga el *buffer* en memoria. Una vez que el servidor completó la tarea requerida puede enviar de vuelta un mensaje confirmando que las acciones ya fueron realizadas y el recurso está disponible (*completion message* en Inglés).

```
(
b = Buffer.read("path/archivo.wav");
c = 2;
)
```

Código 4

El conjunto de sentencias del ejemplo Código 4, genera las instrucciones al servidor para que cargue el archivo "archivo.wav" y asigna la instancia de la clase *Buffer* resultante a la variable *b*. Inmediatamente después, el intérprete, procede a ejecutar la siguiente sentencia y asigna el valor 2 a la variable *c*. En este caso no hay problema en la ejecución del programa, porque solo se pidió que el servidor cargue un archivo de audio. En cambio, si se quisieran ejecutar las instrucciones en Código 5, habría un problema. Inmediatamente después de que fueron enviadas las instrucciones al servidor se le pide al *buffer* que sea reproducido y esto no es posible porque la instrucción *b.play*, es ejecutada antes que el recurso esté disponible. Las soluciones a este problema son variadas, la más simple es primero cargar los recursos y luego ejecutar el programa que dependa de los recursos precargados.

```
(
b = Buffer.read("path/archivo.wav");
b.play;
)
```

Código 5

En otros entornos, si bien la lógica algorítmica puede ser distinta a nivel de superficie, nos podemos encontrar con problemas similares. El siguiente ejemplo en *ChucK* (Código 6) demuestra el problema de la lectura de archivos en tiempo real.

```
"archivo.wav" =>
string filename;
SndBuf buf => dac;
// la operación buf.read debería ser
efectuada aquí. while( true )
{filename => buf.read;
    1::second => now;
}
```

Código 6

En este caso, debido a que tanto el lenguaje como la máquina de síntesis están integrados, la ejecución del programa se detendrá, si la carga del archivo de audio no llega a efectuarse a tiempo, hasta que el recurso esté disponible. Esto, sin duda, produce otro tipo de error que consiste en la interrupción de toda la cadena de procesamiento de audio, lo que generaría interrupciones indeseadas en el sonido. Obviamente este problema también se soluciona precargando los recursos necesarios. En este caso, la instrucción *buf.read*, de uno o más archivos, debería efectuarse antes de iniciar la iteración temporal. Pero la latencia de la carga del archivo no desaparece sino que se produce antes de entrar en la etapa de reproducción.

Estos ejemplos sirven para demostrar varias cosas. Primero, que las restricciones en el tiempo de procesamiento, que son propias de las computadoras y no de los entornos, inevitablemente afectan la implementación de los algoritmos que se ejecutan en tiempo real. Segundo, que una aplicación que se ejecuta en tiempo real implica que el procesamiento requerido se puede efectuar en un período de tiempo definido. Por lo tanto, tener mayor capacidad de computo, si bien siempre es preferible, para ciertas acciones no es necesario.

Existen corrientes estéticas que emplean recursos tecnológicos, con capacidad de cómputo restringida, de manera intencionalmente minimalista (véase por ejemplo Heikkilä 2010 y Schmidhuber 1997). Estas corrientes aceptan las restricciones físicas del ordenador como **restricciones impuestas** de los recursos instrumentales. Las incorporan como una técnica que provee el resultado estético buscado. Al emplear nuevo *hardware* (sin restricciones de potencia), el minimalismo tecnológico necesita emular la capacidad de cómputo anterior, ya sea empleando algoritmos menos sofisticados, recursos de síntesis más simples e incluso bajando la resolución en *bits* y la frecuencia de muestreo del audio procesado.

Esto demuestra que, incluso en el caso de la música por computadora, las abstracciones no solo dependen de los recursos y la técnica instrumental sino que se desarrollan en relación a las concepciones humanas sobre el dominio técnico y artístico. Si bien de múltiples maneras, las concepciones que se tienen sobre un **medio instrumental** afectan su modo de empleo.

Las relaciones que existen entre el cliente y el servidor de *SuperCollider*, respectivamente como interfaz de control y recurso instrumental, posibilitan la visualización de los distintos tipos de elementos que actúan como **abstracciones compositivas** (**interfaz de composición**) y **abstracciones instrumentales** (**interfaz de instrumental**).

SynthDef y Synth

De lado del cliente, una *SynthDef* (definición de instrumento) es una abstracción programable. Es el elemento que representa las abstracciones instrumentales del servidor. Según el grado de especificidad asignado al instrumento diseñado, este puede actuar musicalmente como generador de recursos materiales a distintas escalas temporales y estructurales.

Una *SynthDef* es la definición "estática" de un recurso instrumental específico que genera **materiales lógicos** al ser instanciada como *nodo de síntesis* (*Synth*). Si bien su concepción está relacionada al *modelo recurso-instancia* (Dannenberg, Rubine, y Neuendorffer 1991) (Dannenberg, Honing y Desain op. cit.), las abstracciones resultantes que pueden representar sus *instancias* no están definidas estructuralmente. Compositivamente, la instancia de una abstracción instrumental puede generar materiales simples (e.g. notas) o complejos (e.g. texturas) a distintas escalas temporales.

En relación a su estructura lógica, una definición de instrumento no implica un único nivel de **abstracción material**. Una definición de instrumento puede estar compuesta por un número indefinido de *unidades generadoras* relacionadas de manera compleja, o puede ser simplemente la envoltura de una única *unidad generadora* que se emplea en relación a otras *definiciones de instrumento* actuando a un mismo nivel estructural.

Tanto la estructura instrumental como la estructura material de las definiciones instrumentales son programables de manera no restrictiva. La decisión del nivel de abstracción que representa en relación a una composición musical es variable y depende de las **restricciones voluntarias**. Sin embargo, debido a la lógica específica de los entornos de programación, la elección de **restricciones voluntarias** puede generar **restricciones impuestas** por las abstracciones implementadas (véase en **Capítulo 1**).

Procesamiento en el servidor

La estructura impuesta por las abstracciones empleadas para manipular los recursos de síntesis: *grupos*, *synths*, *buses* y *buffers* son **abstracciones instrumentales**. La estructura se representa como metáfora de elementos "físicos". Del lado del cliente, estas son referenciadas por las clases *Node*, *Group*, *Synth*, *Bus* y *Buffer*.

Según el repertorio de elementos "físicos" representados por el sistema y sus posibles interacciones, se posibilitan distintos tipos de **abstracciones materiales** y sus posibles relaciones. Por ejemplo, en *SuperCollider*, la arquitectura de *nodos* comunicados mediante *buses* posibilita la realización de los procesos de síntesis como entidades individuales. Esto difiere de la implementación de los editores lineales (multipistas) donde la abstracción de *canal* genera una restricción más global que agrupa los materiales musicales dentro de un contexto con cualidades definidas. Los *nodos de síntesis* se

pueden direccionar arbitrariamente y su organización temporal está dispuesta en otro plano de abstracción, en cambio, los *canales de audio* definen el direccionamiento de los elementos contenidos dentro de un agrupamiento estructural de más alto nivel definido como línea (espacio) temporal. Es decir, se integran las **abstracciones instrumentales** (síntesis) con las **abstracciones compositivas** (secuenciación) de manera restrictiva. El concepto de *canal* es equivalente a la organización estructural de los nodos de síntesis en relación a los *grupos* y los *buses* de salida de una única manera predefinida, lo cual genera una **restricción impuesta** por el sistema (ver en **Capítulo 3** *El diseño modular en relación a la ingeniería de software*).

A diferencia del *software* de utilidad específica, los entornos de programación posibilitan la representación de estructuras a menor nivel de abstracción. Esto es lo que genera un mayor grado de flexibilidad a la hora de expresar las concepciones compositivas y organizar los recursos instrumentales necesarios. La flexibilidad de los entornos se define por las posibilidades de adaptar el **nivel base** a distintas necesidades técnicas y compositivas. Mientras más generales sean los componentes de un sistema y sus posibles relaciones, mayor será su capacidad de adaptarse a distintos **niveles base** y generar abstracciones específicas a mayor nivel de abstracción. En esta **Tesis**, esto es considerado como una propiedad necesaria de los entornos de producción musical como medios de representación universal de distintas concepciones compositivas (pasadas, presentes y futuras).

Composición, programación y representación

Con respecto a las **abstracción compositivas**, *SuperCollider* cuenta con un repertorio de recursos extremadamente extenso. La representación de elementos musicales y los recursos de secuenciación abarcan los distintos paradigmas actuales. A continuación, se analizarán solo algunos que resultan relevantes para el desarrollo de esta **Tesis** puesto que su tratamiento *in extenso* está fuera del alcance de la misma.

Las *rutinas*, son funciones especiales cuya ejecución puede ser controlada mediante un reloj externo y actúan como mecanismo de secuenciación básico. En el ejemplo Código 7 se emplea una *rutina* (clase *Rutine*) para secuenciar linealmente una sucesión de eventos que genera nodos de síntesis en el servidor.

```
(
Routine {
    (freq: 440, sustain: (2/3)).play; // ejecuta un
    evento (2/3).wait; // suspende la ejecución (2/3)
    * unidad temporal (freq: 880, sustain:
    (1/3)).play;
    (1/3).wait;
    (freq: 1320, dur: 1).play;
}.play;
)
```

Código 7

Mediante el empleo de estructuras de control (e.g. condicionales, iterativas), los recursos de secuenciación se vuelven mucho más versátiles que en el *software* de utilidad específica puesto que su especificidad no está delimitada por funcionalidades predefinidas. En el ejemplo Código 8 se muestra una secuencia que se ejecuta indefinidamente y genera un evento de frecuencia aleatoria (seleccionada entre 440Hz, 880Hz y 1320Hz) a intervalos temporales también aleatorios (entre 0 y 1 unidades temporales).

```
(
Routine {
    loop {
        (freq: [440, 880, 1320].choose, sustain: 1).play;
        0.5.rand.wait;
    }
}.play;
)
```

Código 8

Con respecto a la versatilidad de los lenguajes de programación como i**nterfaces instrumentales y compositivas**, en el ejemplo Código 9 se muestra la realización de un *arpegiador*, recurso generalmente disponible en los controladores y secuenciadores MIDI. A diferencia de un controlador MIDI, donde el tipo de arpegio se selecciona (generalmente) de entre un repertorio disponible, en un lenguaje de programación este se puede programar cambiando la lógica interna de la función de cualquier manera imaginable.

```
(
~acorde = [60, 64, 67];
~dur = 1/8;

Routine {
    loop {
        ~acorde.pyramid.do { arg nota;
            (midinote: nota, sustain: ~dur).play;
            ~dur.wait;
        }
    }
}.play;
)
```

Código 9

En cuanto a los distintos tipos de relaciones temporales, la secuenciación de eventos simultáneos se puede programar de distintas maneras. Por ejemplo, agrupando los eventos dentro de un mismo tiempo de espera (agrupamiento armónico) o superponiendo distintas rutinas en paralelo (agrupamiento contrapuntístico) como se muestra en el ejemplo Código 10. Este ejemplo demuestra que las **abstracciones compositivas** y musicales (e.g. voces y acordes) se pueden representar de distintas maneras según como se combinen los **materiales lógicos**.

```
(
// agrupamiento armónico
(simultáneo) Routine {
    (midinote: 60, sustain:
    3).play; // voz 1 (midinote:
    65, sustain: 1).play; // voz
    2
    1.wait; // se ejecuta el acorde
    (midinote: 64, sustain: 0.5).play; // voz 2
    0.5.wait;
    (midinote: 62, sustain: 0.5).play; // voz 2
    0.5.wait;
    (midinote: 48, sustain:
    1).play; // voz 1 (midinote:
    63, sustain: 1).play; // voz
    2
}.play;
)

(// agrupamiento
contrapuntístico (paralelo)
Routine {
    (midinote: 60, sustain: 3).play; // voz 1
```

```
        2.wait;

        (midinote: 48, sustain: 1).play; // voz 1
    }.play;

    Routine {
        (midinote: 65, sustain: 1).play; // voz 2
        1.wait;
        (midinote: 64, sustain: 0.5).play; // voz 2
        0.5.wait;
        (midinote: 62, sustain: 0.5).play; // voz 2
        0.5.wait;
        (midinote: 63, sustain: 1).play; // voz 2
    }.play;
    )
```

Código 10

Para desarrollar este tipo de abstracciones basadas en la secuenciación de eventos, *SuperCollider* dispone de una librería de *patterns*, que consiste en una colección de distintos tipos de estructuras secuenciales predefinidas, sobre las cuales se pueden realizar distintas operaciones, y un mecanismo estándar para reproducirlas. En comparación con las rutinas, la librería de *patterns* es un conjunto de abstracciones de más alto nivel que agiliza la programación de estructuras musicales específicas. En el ejemplo Código 11 se muestra el equivalente al ejemplo Código 10 empleando *patterns*.

En comparación con las *rutinas*, la interfaz de programación de la librería *patterns* es de carácter *declarativo*, es decir que, encapsula los procesos de control necesarios para la secuenciación, para enfocarse en los datos que representan la secuencia. Este tipo de recursos actúa como una forma de “ocultar” las **abstracciones instrumentales** priorizando la representación de los datos de entidades conocidas.

```
(
Ppar([
    Pbind(
        \midinote, Pseq([60, 48]),
        \sustain, Pseq([3, 1]),
        \dur, Pseq([2, 1])
    ),
    Pbind(
         \midinote, Pseq([65, 64, 62, 63]),
         \sustain, Pseq([1, 0.5, 0.5, 1]),
         \dur, Pseq([1, 0.5, 0.5, 1])
    )
]).play;
)
```

Código 11

Como **interfaz compositiva**, la librería de *patterns* restringe los recursos de representación a la lógica subyacente de su implementación. En el ejemplo Código 11, la clase *Ppar* es una estructura que representa la relación simultánea de dos secuencias de eventos representadas por la clase *Pbind*. Internamente, la clase *Pbind* representa los parámetros de la sucesión de eventos, los cuales son componentes **simultáneos**, mediante listas de valores (clase *Pseq*), por separado para cada parámetro. Aunque dependa de la implementación, este tipo de agrupamientos implica una concepción compositiva. Los eventos se componen de *streams* como **procesos paralelos** (instancias de *Pseq*) por cada parámetro de *Pbind*, de la misma manera que la simultaneidad representada por *Ppar* con respecto a las instancias de *Pbind*. Esto es equivalente a la segunda variante del ejemplo Código 10 (agrupamiento contrapuntístico).

Según el marco teórico de esta **Tesis**, las *rutinas* y los *patterns* son representativos de los **materiales lógicos**. El material musical está definido por diferentes estructuras que representan la lógica de generación de los **materiales concretos**, los cuales se **realizan** al ejecutar el código de programación y producir el sonido. Sin embargo, los eventos, que se representan como instrucciones entre paréntesis, pueden ser considerados como materiales concretos dentro del nivel de abstracción propuesto por la *rutina* que los contiene. Esto se debe a que los eventos representan valores concretos con respecto al nivel de abstracción inmediatamente superior (ver **Capítulo 6**).

Los eventos, en *SuperCollider*, además representan, de manera implícita, una *definición de instrumento* (organización lógica de *unidades generadoras*) y su mecanismo de ejecución como *nodo* de síntesis. La funcionalidad asignada a los eventos encapsula las **abstracciones instrumentales** simplificando la escritura del programa. Pero si se analizan los elementos encapsulados se puede entender que estos son **materiales**

lógicos con respecto al evento como nivel de abstracción inmediatamente superior. Esto ejemplifica la naturaleza **recursiva** de las abstracciones informáticas en relación a las cualidades de los materiales musicales estudiados por esta **Tesis**44.

Es importante mencionar aquí que no debe confundirse la estructura lógica de los programas implementados, con la estructura de los **materiales lógicos** y **concretos** que estos representan. Las estructuras de datos y de control pueden agruparse en sub-estructuras, las *rutinas* pueden ser llamadas dentro de funciones en distintos momentos de la misma manera que estas llaman a los eventos, pero esto no necesariamente es equivalente a la estructuración musical resultante. En los casos expuestos, la relación es clara puesto que los ejemplos fueron elaborados para que así fuera, pero en programas más complejos es necesario analizar qué estructuras informáticas son las que coinciden con las estructuras musicales representadas como **materiales lógicos** o **concretos** y cuales son producto de las técnicas de organización del código de programación.

En el siguiente ejemplo (Código 12), un poco más extenso, se analizan estos factores. El ejemplo fue elaborado *ad hoc* para exponer los aspectos considerados relevantes, pero su contenido no difiere de las prácticas habituales puesto que las abstracciones informáticas son invariantes. En esta sección del libro, solo se tratarán los aspectos representacionales en relación la estructura del lenguaje de programación. Luego, en el **Capítulo 7**, este ejemplo se volverá a analizar desde la perspectiva de la representación de los materiales musicales aplicando el modelo propuesto por esta **Tesis**.

```
1 (
2 // definiciones de instrumento (abstracción instrumental)
3 SynthDef(\playbuf, { arg out = 0, pan = 0, buf, amp = 1;
4     var sig;
5     sig = PlayBuf.ar(1, buf) * amp;
6     sig = Pan2.ar(sig, In.kr(pan));
7     Out.ar(out, sig);
8 }).add;
9
10 SynthDef(\mod, { arg out, freq = 5, amp = 1;
11     var sig;
12     sig = LFNoise2.kr(freq, amp);
13     Out.kr(out, sig);
14 }).add;
15
16 SynthDef(\sine, { arg out = 0, freq = 440, amp = 0.2, pan = 0;
17     var sig, env;
18     sig = SinOsc.ar(freq.lag(0.2), 0, amp.lag(0.2));
19     env = EnvGate.new;
20     sig = Pan2.ar(sig * env, In.kr(pan));
21     Out.ar(out, sig);
```

```
 22 }).add;
 23
 24 // nodo grupo (abstracción instrumental del servidor)
 25 ~grupo1 = Group.new;
 26 ~grupo2 = Group.after(~grupo1);
 27
 28 // bus (abstracción instrumental del servidor)
 29 ~bus1 = Bus.control;
 30 ~bus2 = Bus.control;
 31
 32 // buffer (abstracción instrumental que contiene
 33 // una abstracción material concreta)
 34 ~archivo = Buffer.read(s, "archivo.wav");
 35 )
36
37 (
38 // ~funcSine organiza el código, no los materiales musicales
39 ~funcSine = { arg synth;
40     var ret = [];
41
42     // rutina de control (material lógico)
43     ret = ret ++ Routine {
44         loop {
45             synth.set(\freq, [60, 62, 67, 69, 72].choose.midicps);
46             [0.5, 1].choose.wait;
47         };
48     }.play;
49
50     // rutina de control (material lógico)
51     ret = ret ++ Routine {
52         loop {
53             synth.set(\amp, 1.0.rand);
54             0.2.wait;
55         };
56     }.play;
57
58     ret;
59 };
60 )
61
62 (
63 Routine.new {
64     var playbuf, mod1, sine, mod2, rout;
65
66     s.bind {
67         mod1 = Synth(\mod, [out: ~bus1], ~grupo1);
68         playbuf = Synth(\playbuf,
69             [buf: ~archivo, pan: ~bus1], ~grupo2);
70     };
```

```
71
72      10.wait;
73
74      s.bind {
75          mod2 = Synth(\mod, [out: ~bus2], ~grupo1);
76          sine = Synth(\sine, [fadeTime: 10, pan: ~bus2], ~grupo2);
77          s.sync;
78          rout = ~funcSine.value(sine);
79      };
80
81      20.wait;
82
83      sine.set(\gate, 0);
84      8.wait;
85
86      rout.do(_.stop);
 87      playbuf.free;
 88      mod1.free;
 89      mod2.free;
 90
 91      // necesario para la programación
 92      ~bus1.free;
 93      ~bus2.free;
 94      ~archivo.free;
 95  }.play
 96  )
```

Código 12

Puesto que su descripción sería demasiado extensa, del ejemplo Código 12 solo se explicarán las abstracciones relevantes para la representación musical dejando de lado varios detalles, especialmente sintácticos, que son propios de los recursos de programación. El significado de estos detalles no afecta la compresión del ejemplo.

Entre las líneas 2 y 22 del ejemplo Código 12, se realizan tres definiciones de instrumento: *playbuf*, *mod* y *sine*; *playbuf* es una abstracción instrumental que se encarga de reproducir un *buffer* de audio y disponerlo en el espacio estéreo (*pos*); *mod* es una **abstracción instrumental** que genera una señal de control aleatoria de la cual se pueden controlar los parámetros del movimiento, frecuencia (*freq*) y amplitud (*amp*), del proceso estocástico; y *sine*, que es un instrumento que genera sinusoides que varían su frecuencia (*freq*) e intensidad (*amp*) y su posición el en campo estéreo (*pan*).

Entre las líneas 24 y 34 del ejemplo Código 12, se definen **abstracciones instrumentales** del servidor necesarias para organizar los algoritmos de síntesis (definiciones instrumentales) y control (secuenciación) posteriores. Se definen dos *nodos grupo*, para organizar las instancias instrumentales y dos *buses* de control para interconectar los *nodos de síntesis*. En la línea 34 se define un *buffer* y se carga un archivo

de audio (“archivo.wav”). El *buffer* es una abstracción instrumental que pasa a representar el **material concreto** contenido en el archivo de audio.

El bloque de código comprendido entre las líneas 37 y 60 (Código 12), contiene la definición de una función que actúa como elemento organizativo del programa. Esta función encapsula la representación contrapunstística de dos materiales lógicos que pueden controlar la frecuencia y la amplitud de un *nodo de síntesis* mediante el envío directo de parámetro discretos de control (instrucciones *synth.set* en las líneas 45 y 53). Los elementos de secuenciación de estos parámetros están representados mediante *rutinas* del lado del cliente y actúan como material lógico de manera equivalente a la señal generada para la definición de instrumento *mod* (linea 10). El ejemplo fue realizado de esta manera para demostrar como distintas abstracciones informáticas pueden estar representando un mismo tipo de **abstracción material** musical.

El bloque de código entre las líneas 62 y 96 (Código 12) es el algoritmo de secuenciación de las **abstracciones instrumentales** y los **materiales lógicos y concretos** empleados. Lo lógica de este algoritmo está contenida dentro de una *rutina* (líneas 63 a 95) que es la **abstracción material lógica** de más alto nivel en este ejemplo.

Dentro de esta *rutina*, entre las líneas 66 y 70 se instancia un instrumento de control *mod* (línea 67) cuya señal de salida se dirige a través del *bus*, almacenado en al variable *~bus1*, al parámetro de entrada *pos* del la instancia del instrumento *playbuf* definida en la linea siguiente. La instancia *playbuf* es la que se encarga de reproducir el material concreto contenido en el archivo de audio “archivo.wav”. Ambos nodos de síntesis (*~mod1* y *~playbuf*) son instanciados dentro de distintos grupos que definen la relación de orden del procesamiento. *~mod1* va al *~grupo1* y se procesa primero que ~playbuf que se dispone en *~grupo2*. Del lado del servidor, los procesos de síntesis contenidos en *~grupo1* se ejecutan antes que los de *~grupo2* puesto que las señales generadas en *~grupo1* deben estar disponibles para los parámetro de entrada de los *nodos de síntesis* del *~grupo2*.

Luego de 10 unidades temporales de espera (línea 72), en la línea 75 se instancia otro instrumento *mod* pero ahora en relación al parámetro de espaciliazación de la instancia de *sine*. La lógica organizativa es exactamente igual que en el caso anterior excepto por la llamada a la función *funcSine* a la cual se le pasa como parámetro el *nodo de síntesis ~sine*. La función *~funcSine* genera, a partir de ese momento, las *rutinas* de control contrapuntístico de la altura y la amplitud del instrumento *sine*.

El algoritmo se continúa ejecutando durante 20 unidades temporales (linea 80) con los comportamientos establecidos hasta el momento. El resultado sonoro es la superposición de dos estratos, un sonido complejo y dinámico espectralmente, expresado como **material concreto** proveniente del archivo “archivo.wav” y una sinusoide que varía

en altura y amplitud expresada como la combinación de varios **materiales lógicos** que actúan como generadores del sonido y su comportamiento dinámico. Ambos materiales varían estocásticamente su posición espacial controlada por el **material lógico generado** por los *nodos mod*.

En la línea 83, se le envía una orden de control al nodo *~sine* que hace que este comience su tiempo de extinción (definidos en 10 unidades temporales en la línea 76). 8 segundos después de esto se detiene la ejecución de las *rutinas de control* del *nodo ~sine*, el nodo *~playbuf* y los *nodos* de control *~mod1* y *~mod2* (líneas 86 a 89). El sonido se desvanece por completo 2 unidades temporales después al completarse el *fadeout* y liberarse el nodo *~sine*.

Las líneas 92 a 94 liberan los recursos de las abstracciones instrumentales reservados por el algoritmo de síntesis y no cumplen otra función que la de finalizar correctamente el programa.

El análisis realizado expuso los recursos lógicos empleados para la composición empleando lenguajes de programación. Los resultados musicalmente significativos que estos recursos implican, serán analizados en detalle en el **Capítulo 7**.

Quarks

Al ser un lenguaje orientado a objetos derivado de *Smalltalk*, *sclang* depende de una librería de clases que provee la funcionalidad básica necesaria para la programación. Dentro de la distribución estándar del entorno vienen incorporadas librerías de clases para realizar operaciones sobre distintos tipos de objetos, como las abstracciones del servidor, los eventos o los *patterns*, entre muchas otras abstracciones específicas del dominio musical. Que el lenguaje de *SuperCollider* se considere orientado a la síntesis de sonido y la composición algorítmica se debe a que implementa esa funcionalidad mediante librerías específicas del dominio musical y el procesamiento digital de señales.

Mediante el lenguaje de programación es posible extender la funcionalidad estándar agregando librerías específicas. Estas librerías específicas se denominan *quarks45* y pueden ser incorporadas al entorno simplemente haciendo que sean visibles por el intérprete (*sclang*) al inicializarse. Es conveniente notar que los *quarks* no implementan algoritmos de síntesis sino abstracciones de alto nivel que se ejecutan del lado del cliente. La implementación de algoritmos de procesamiento de señales (*unidades generadoras*) se implementan como *plugins* (librerías dinámicas) del lado del servidor, mediante una API en C/C++.

El desarrollo de *quarks* posibilita la implementación de nuevas **abstracciones compositivas** que pueden interactuar con la lógica del programa de síntesis y las

estructuras musicales básicas de la librería estándar.

A los fines de esta exposición, de entre los muchos *quarks* que existen, interesa destacar tres librerías que implementan los distintos tipos de recursos compositivos clasificados en esta **Tesis**: 1) La librería *pcslib-sc* (Samaruga 2013), que implementa **relaciones abstractas**, 2) la librería *JITLib*46, que implementa una **interfaz instrumental** y, 3) la librería *CTK47* que implementa una **interfaz de composición**.

La librería *pcslib*-sc48 implementa la funcionalidad necesaria para la composición con *pitch- class sets* y *matrices combinatorias* (Morris 1987). Está orientada a la realización de operaciones y la organización de **relaciones abstractas** que luego pueden ser **realizadas** con los recursos de secuenciación y síntesis propios de *SuperCollider*. Implementa las clases necesarias para representar y manipular los *pitch-class sets*, las *matrices combinatorias* y las operaciones de la teoría atonal. Los datos procesados existen únicamente del lado del cliente y no representan **materiales lógicos** ni **concretos** por sí mismos sino relaciones de coherencia estructural. Es un ejemplo los tipos de abstracciones informáticas que se emplean para generar estructuras atemporales.

La librería *JITLib* (acrónimo de *Just In Time Library*) realiza un aporte interesante relacionado con la estructura del servidor. Cosiste en la implementación de objetos de más alto nivel que encapsulan las **abstracciones instrumentales** de niveles inferiores pero de manera tal que agregan funcionalidad y flexibilidad al empleo de las estructuras propias de *SuperCollider*. La librería se basa en la definición de nodos de síntesis y de control que adquieren mas propiedades facilitan la manipulación y la variación de las estructuras dinámicamente en tiempo real. Unifica las **abstracciones instrumentales** básicas de *SuperCollider*, tanto para la producción de síntesis como la secuenciación del lado del servidor, pero emulando los recursos instrumentales de la máquina de síntesis y encapsulando algunos recursos (e.g. los *buses*) para simplificar la manipulación pero sin perder flexibilidad. Esta librería fue diseñada, y es muy empleada, para el *live coding49* (Collins et. at. 2003) (Wang y Cook 2004). Manifiesta la concepción de **interfaz instrumental** puesto sus elementos adoptan la representación “tácita” del tiempo (Honing 1993). Es un ejemplo de cómo el ordenador es entendido como recurso instrumental. El compositor es el intérprete/improvisador que emplea el lenguaje de programación como instrumento musical sobre el escenario.

Por último, la librería *CTK* (acrónimo de *Composers ToolKit*) expone la noción de **interfaz de composición**. De manera similar a *JITLib*, esta librería encapsula las abstracciones del servidor de manera tal que puedan interactuar con mayor flexibilidad a un mismo nivel de abstracción entre si. Integra la manipulación de los recursos de síntesis con los recursos de secuenciación y la producción de abstracciones temporales que representan **materiales musicales lógicos** o **concretos**. Puede ser

empleada para la síntesis en tiempo diferido puesto que simplifica notablemente la complejidad que surge del empleo de las abstracciones instrumentales de más bajo nivel y posibilita la organización y la manipulación temporal de los **materiales lógicos y concretos**. Esta librería abstrae las **abstracciones materiales** de las **abstracciones instrumentales**, facilitando las técnicas de composición orientadas hacia la manipulación de **abstracciones materiales**.

Estos son simplemente tres ejemplos de como el lenguaje de programación *sclang* posibilita la realización de distintas **concepciones compositivas** dentro de un entorno integrado para síntesis de sonido y composición algorítmica. Los niveles de abstracción y las técnicas de composición pueden ser muy variadas y su representación depende de la implementación particular de abstracciones de más alto nivel.

Conclusiones

Debido a que el lenguaje *sclang* integra, mediante la programación orientada a objetos, las abstracciones que actúan en ambos planos (compositivo e instrumental), es posible desarrolla una composición musical dentro de un medio de representación integrado. Sin embargo, las abstracciones desarrolladas por un entorno pueden diferir, y en muchos casos lo hacen, de las desarrolladas por otro, aunque su finalidad sea equivalente. Es por esto que en esta **Tesis** se considera conveniente adoptar un **modelo** (ver **Capítulo 6**) de representación que pueda ser empleado para representar estos mismos elementos de manera generalizable a cualquier entorno.

En la actualidad y en relación a la producción pasada, tal **modelo** no garantizaría la unificación de las herramientas informáticas existentes sino que posibilitaría la expresión de sus recursos de manera universal, tanto técnica como teóricamente. Para el análisis de obras electroacústicas, que emplean lenguajes de programación, el analista debe ser un estudioso de los conceptos del entorno específico en el cual fue compuesta la obra para poder discernir los conceptos generales y poder exponer sus resultados de manera concisa.

La diferencia que existe entre distintos entornos de programación para composición algorítmica/asistida y síntesis de sonido es similar a la que se produce entre distintos medios instrumentales (e.g. tradicionales, analógicos o digitales) o entre distintos instrumentos de un mismo medio. Lo que cambia notablemente es la implementación del instrumento y su modo de empleo. Sin embargo, el ordenador, al ser capaz de incluir las **abstracciones compositivas**, agrega explícitamente otro nivel de representación que puede ser considerado y analizado como recurso de notación y expresión particular.

CAPÍTULO 6: El Modelo de Representación

Introducción

El modelo de representación de los materiales musicales propuesto por esta **Tesis** consiste en la representación de las **estructuras musicales en función del tiempo**. Es una representación de las **abstracciones compositivas** que integra las **abstracciones instrumentales** como representación temporal de los datos y los procesos algorítmicos.

Las estructuras representadas son generales a los recursos de organización informática de los materiales musicales, los cuales son entendidos como **objetos** (**material concreto**/**generado**) o **procesos** (**material lógico**/**generador**) integrados y relacionados estructuralmente. La integración se produce mediante **relaciones jerárquicas** de **agrupamientos lógicos**.

Con respecto al marco teórico, son importantes los siguientes elementos:

- La **concepción paramétrica** del sonido y los materiales musicales (**Capítulo 1**).
- La definición de los materiales como objetos (**concretos**/**generados**) y procesos (**lógicos**/**generadores**), su capacidad de cambiar de **estado** y sus formas de relación (**Capítulo 3**).
- Las escalas y las propiedades temporales (**Capítulo 3**).
- Las relaciones abstractas, jerárquicas y estructurales: la diferencia entre **relaciones abstractas** y **realización**, los principios de agrupamiento (teóricos, informáticos y perceptivos), la relación entre estructura objetiva y subjetiva (**Capítulo 3**).
- La diferencia entre **abstracción instrumental** y **abstracción compositiva** (**Capítulo 4**).
- La definición de **nivel base** (**Capítulo 4**).

El modelo en sí representa la realización temporal, como abstracción compositiva, de los materiales musicales en referencia a un **nivel base** variable. Los niveles de referencia se anotan mediante subíndices en relación al nivel cero. Es decir: *n0*, es el nivel base arbitrariamente definido; *n1*, *n2*, etc, son las abstracciones que representan niveles estructurales más complejos en relación al nivel de referencia y; *n-1*, *n-2*, etc, son las abstracciones que representan niveles estructurales más simples en relación al nivel de referencia.

Las primitivas (**estructuras materiales básicas**) que actúan, **recursivamente** son las siguientes:

- Evento
- Lista
- Vector
- Conjunto
- Función

Estas definen, de manera no redundante, los principios estructurales básicos que se emplean para la composición con medios electroacústicos empleando medios digitales. Su especificidad será explicada en detalle en las secciones siguientes, por el momento, solo se exponen como definición de los elementos que componen el modelo. Su denominación refiere a tipos de estructuras empleadas comúnmente en la programación pero no implica el comportamiento específico de ningún entorno en particular.

Como se explicará más adelante, un **material musical**, definido según las estructuras básicas de este modelo, puede actuar como **contexto** o **parámetro**. Como contexto, un determinado nivel de abstracción actúa como **nivel base** y, a partir de este, se elaboran:

- Estructuras de más alto nivel (tendientes a la simplicidad).
- Estructuras de más bajo nivel (tendientes a la complejidad).

El siguiente apartado provee un visión general del **modelo de representación** a partir de sus principios de **agrupamiento estructurales** como **entidades recursivas**. Luego se analizan las características de las estructuras informáticas en relación a las estructuras musicales y las cualidades que de estas se desprenden, se exponen las **estructuras básicas** en detalle y sus propiedades al actuar como contexto o como parámetro y los principios relacionales de agrupamiento y estructuración jerárquica. Por último, se exponen brevemente definiciones básicas para una posible implementación en *software*, las cuales quedarán como referencia para trabajos posteriores a esta **Tesis**.

Representación Estructural Recursiva (Contexto o Parámetro)

Para desarrollar la explicación de los **agrupamientos recursivos** se emplearán los siguientes recursos gráficos que representan los **materiales básicos** en dos dimensiones. La explicación detallada de estos materiales será expuesta posteriormente en este capítulo.

Las estructuras de datos se representan como rectángulos. La dimensión *y*, eje de las ordenadas, adquiere significados variables según el **contexto** que generan las abstracciones, puede representar la estructura o una **propiedad paramétrica** de los elementos contenidos. La dimensión temporal se representada de manera tradicional sobre el eje de las abscisas puesto que los elementos de representación están orientados hacia las **abstracciones compositivas**.

El **evento** se representa como en la Figura 5. El eje de las ordenadas no adquiere el significado de ningún parámetro interno y su extensión depende de las necesidades visuales del contexto que lo contiene.

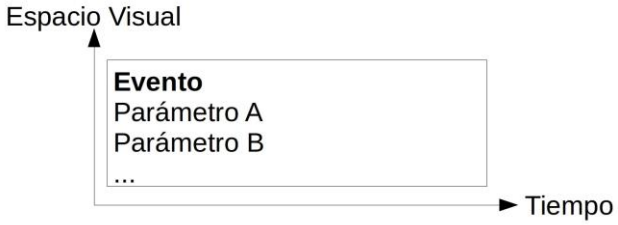


Figura 5 Evento: el eje de las ordenadas puede representar espacio visual u organización estructural.

La **lista** es una estructura que puede determinar un **contexto** (relaciones internas) que signifique el eje de las ordenadas, los elementos contenidos pueden estar ordenados verticalmente el valor de un determinado parámetro. Si no se le asigna relación de orden vertical a los elementos contenidos esta dimensión es meramente visual (ver **conjunto** a continuación). En la Figura 6 se muestra una lista que contiene una sucesión de eventos ordenados verticalmente.

Figura 6 Lista: el eje de las ordenadas representa una relación de orden escalar u arbitraria.

El **vector** representa cambios de valores continuos y, por lo tanto, el eje de las ordenadas adquiere significado escalar. El contenido puede estar representado si resulta

simbólicamente significativo. En la Figura 7 se muestra un **vector** que representa una señal de audio expresada como oscilograma.

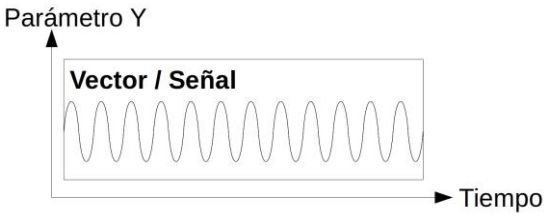


Figura 7 Vector: el eje de las ordenadas representa una relación escalar.

El **conjunto**, al igual que la lista, también puede determinar un **contexto** interno que signifique el eje de las ordenadas. Como se verá más adelante en el apartado “Recursos gráficos y programación”, la relación vertical puede ser de distintos tipos, ordenados o no. En la Figura 8 se muestra un **conjunto** que contiene eventos homogéneos los cuales están ordenados verticalmente según algún parámetro escalar.

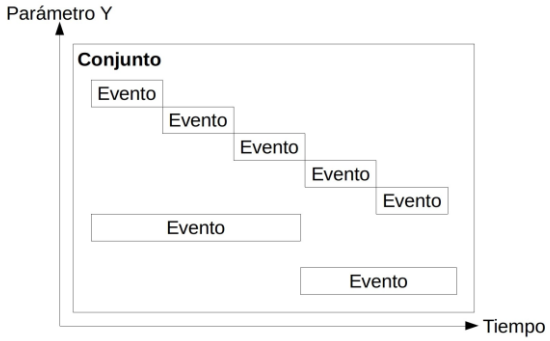


Figura 8 Conjunto: el eje de las ordenadas representa una relación de orden escalar u arbitraria.

La **función** (Figura 9) se representa de la misma manera que los eventos, el eje de las ordenadas es una necesidad visual y puede contener texto que explique la lógica de procesamiento del elemento representado. Si la **función** representa un **material lógico** (ver definición más adelante en este capítulo) que puede no haber cambiado a estado concreto, se emplea la delimitación temporal estimada al momento de su concreción mediante un rectángulo o una línea punteada que indique el tiempo de actividad de la función dentro de un contexto temporal.

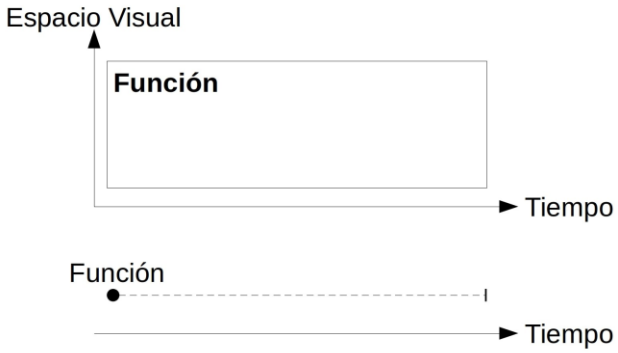


Figura 9 Función: el eje de las ordenadas es simplemente espacio visual.

La representación **recursiva** de estructuras en función del tiempo se puede expresar gráficamente como se muestra en la Figura 10. En el gráfico se muestran tres estructuras anidadas dentro del rectángulo externo entendido como **nivel de referencia**. Las flechas indican el parámetro representado por los ejes cartesianos dentro de cada nivel estructural. Se emplea una escala de grises como ayuda visual para identificar las estructuras internas, mientras más obscuras más profundas, pero esto no adquiere ningún significado relevante más allá de esta función visual.

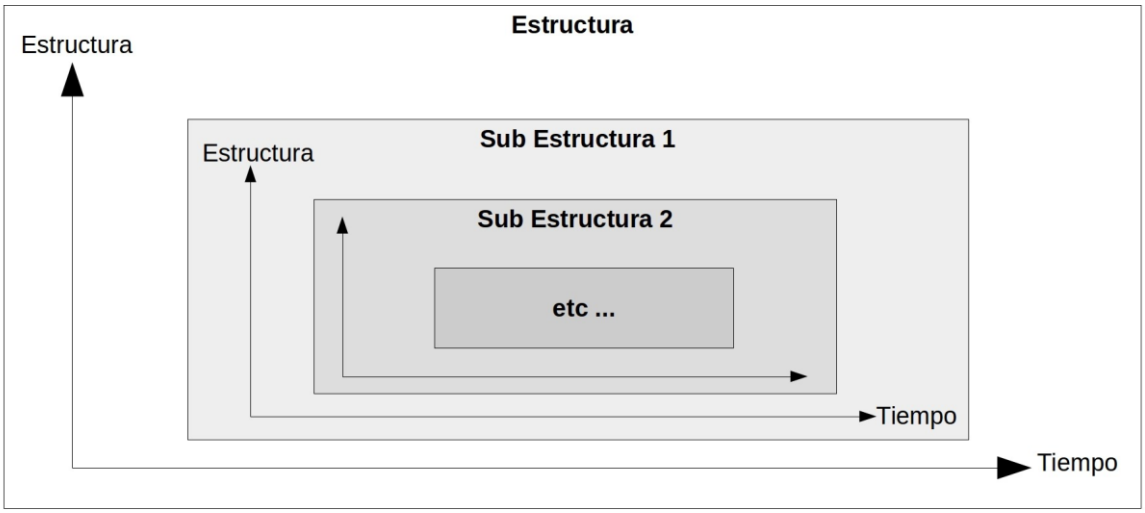


Figura 10 Representación estructural recursiva expresada gráficamente.

Las estructuras externas actúan como **contexto** de las estructuras internas o contenidas. Un contexto es una **escala de referencia que relaciona los elementos contenidos y define las propiedades temporales de estos**. Como se verá más adelante, son las estructuras primitivas (**evento**, **lista**, **vector**, **conjunto** y **función**) las que actúan como contexto de los elementos contenidos, los cuales, a su vez, pueden ser cualquiera de estas primitivas y sus posibles combinaciones.

En la Figura 10 el contexto está representado mediante el eje de las ordenadas que, en este caso, siempre refiere a la estructura de los elementos contenidos y no especifica ordenamiento alguno. Es decir, actúa como organizador de las estructuras en el espacio visual.

En las Figura 11 y Figura 12 se muestra como las estructuras abstractas adquieren significado musical en relación al **nivel base** (índice 0). El rectángulo externo de la Figura 11 representa la obra musical en su totalidad a escala macro temporal. Los rectángulos internos representan los distintos **niveles de abstracción** y sus índices son positivos a mayor nivel de abstracción y negativos en el caso opuesto. En este ejemplo (Figura 11), el **nivel base** está siendo visualizado desde la macro estructura de la obra musical y, por lo tanto, corresponde a un nivel de abstracción intermedio dentro del gráfico, con el cual se relacionan dos niveles de abstracción superior (“Obra Musical” y “Nivel Base +1”) y uno inferior (“Nivel Base -1”) (Figura 11).

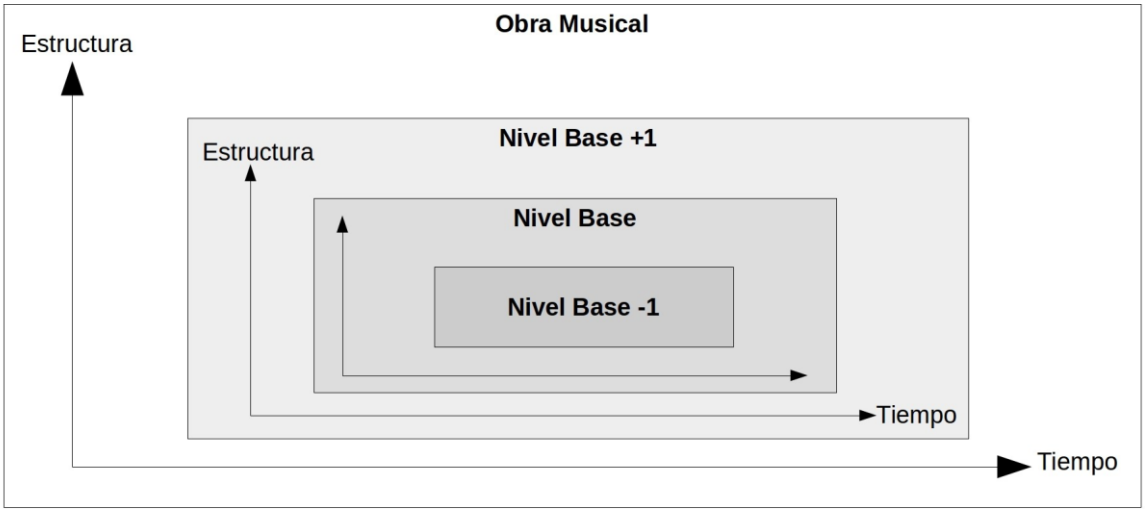


Figura 11 Nivel base y niveles relacionados.

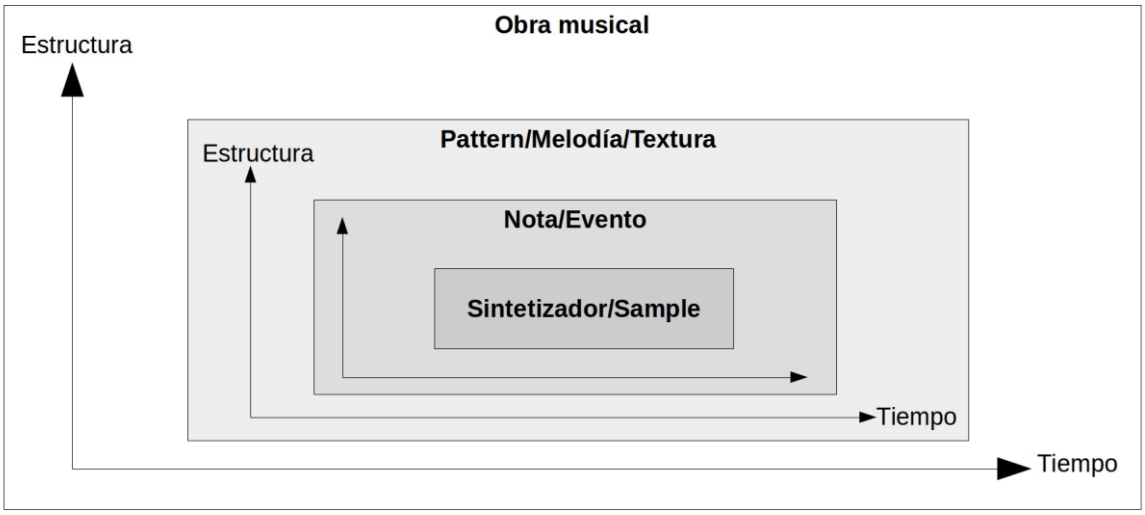


Figura 12 Posibles significaciones de los elementos estructurales.

En la Figura 12 se le asignan posibles significaciones, como estructuras compositivas, a los distintos niveles de abstracción50. El nivel base de la Figura 11 puede estar representando un evento o nota musical en la Figura 12. El nivel de abstracción inferior, “Nivel Base -1” en la Figura 11, puede estar representando un algoritmo de síntesis (“Sintetizador”) o un archivo de audio correspondiente a una nota de un banco de sonido (s*ample*) en la Figura 12. El nivel de abstracción superior, “Nivel Base +1”, al **nivel base** de la Figura 11 podría estar representando, opcionalmente, tanto un “*Pattern*” (relación abstracta de eventos) como una “Melodía” (sucesión de notas) o una “Textura” (entendida como material manipulable resultado de la combinación arbitraria de varios eventos-nota) en la Figura 12.

Lo que manifiesta este ejemplo es que el **nivel base** no solo es variable sino que, además, **puede adquirir distintos significados con respecto a los materiales musicales representados** y sus posibles manipulaciones como entidades compuestas. En el ejemplo se optó por especificar estructuras convencionales para facilitar el entendimiento del principio de estructuración jerárquica. Pero el modelo es lo suficientemente general como para representar cualquier tipo de abstracción o concepción compositiva. Incluso los agrupamientos expuestos en estos gráficos son arbitrarios y las mismas estructuras musicalmente significativas podrían estar agrupadas de distinta manera (como se explicará más adelante).

En la Figura 13 se expone otro ejemplo específico de una realización musical simple que consiste en una secuencia de cuatro notas. El gráfico se divide en la representación de tres niveles estructurales (A, B y C) del mismo fragmento musical. A cada nivel se definió un tipo de estructura musical y su organización de manera arbitraria con fines

expositivos.

La relación contextual de las estructuras resulta en los siguientes casos:

1. La “Obra Musical” actúa como contexto de la “Melodía”.
2. La “Melodía” actúa como contexto de los “Eventos-Nota”.
3. El “Evento-Nota” actúa como contexto de los parámetros “Frecuencia” y “Amplitud” y la abstracción “Sintetizador”.
4. El “Sintetizador” actúa como contexto de tres “Unidades Generadoras”.

En los casos 1, 3 y 4, el contexto representa verticalmente la estructura de los elementos contenidos de igual manera que en los ejemplos anteriores. El caso 2 difiere y representa el ordenamiento de un parámetro interno: la altura.

Como ya se mencionó, las abstracciones pueden actuar como **contexto** o como **parámetro**. Al actuar como **parámetro**, las abstracciones son entendidas como **dato de entrada de otra abstracción a un mismo nivel de jerarquía o perteneciente a algún nivel inferior**. Un parámetro puede ser, entre muchas cosas, el valor de una nota MIDI, una envolvente dinámica e incluso una estructura compuesta.

En el nivel estructural C, la “Frecuencia” y la “Amplitud” son **abstracciones paramétricas** que actúan como dato de entrada de la **abstracción instrumental** “Sintetizador” y que están contenidas dentro del contexto “Evento-Nota”.

Cuando las abstracciones internas actúan como parámetros es posible ordenarlas en el eje vertical de algún contexto que las contenga. En este ejemplo, el contexto “Melodía” le da la importancia tradicional al parámetro altura y, en consecuencia, se ordenan verticalmente los “Eventos-Nota”.

Sin embargo, en los niveles estructurales A y B no se le dio importancia gráfica a la abstracción paramétrica “Frecuencia”, representada en el nivel estructural C, pero se la empleó implícitamente como criterio de ordenamiento de los “Eventos-Nota”. Esto es importante puesto que no es posible representar visualmente todos los componentes de cada nivel de abstracción si el punto de vista (a la manera de un *zoom*) es demasiado lejano.

A cada **nivel de abstracción** se pueden representar los **contexto** y **parámetros** contenidos que se consideren de mayor utilidad, ya sea gráficamente o por otros medios51. Incluso los ordenamientos paramétricos pueden ser variables, por ejemplo, el nivel estructural B podría haberse ordenado mediante el parámetro “Amplitud” de los “Eventos-Nota” contenidos.

La representación estructural o paramétrica también se puede definir arbitrariamente. En este ejemplo se optó por pasar de la representación estructural en el contexto “Obra Musical” a la representación paramétrica en el contexto “Melodía” y se vuelve a la representación estructrual en los contextos “Evento-Nota”.

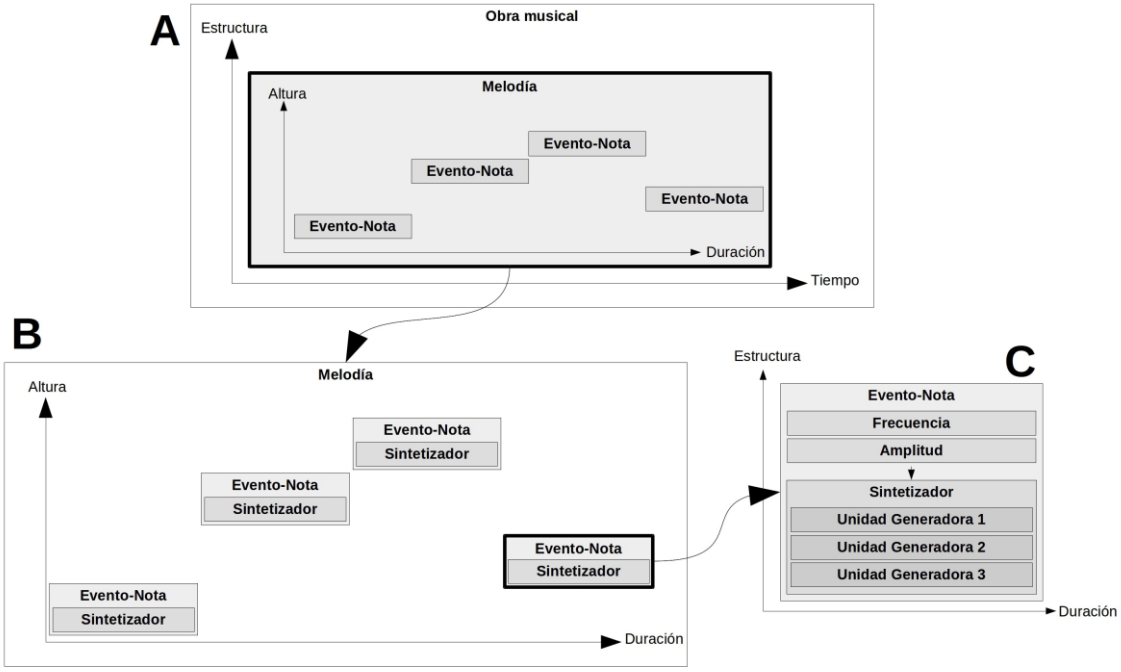


Figura 13 Realización musical simple descompuesta en tres niveles, gráficos A, B y C.

Es necesario aclara una vez más que, por razones de simplicidad expositiva, estos ejemplos fueron dados con referencias a entidades musicales tradicionales conocidas: obra musical, melodía, textura, nota, sintetizador, etc, pero los elementos que conforman el conjunto de primitivas de este modelo se ciñen a los cinco tipos ya mencionados: **evento**, **lista**, **vector**, **conjunto** y **función**.

Lingüísticamente, **las estructuras tradicionales son referencias significadas de los tipos básicos y sus agrupamientos (estructuras/abstracciones significantes)**. Por ejemplo, una melodía es un significado la primitiva **lista***,* de la misma manera, una nota lo es de un **evento**, una textura o una obra pueden ser distintos significados de un **conjunto**, una unidad generadora puede ser el significado de una **función**, la abstracción paramétrica de la amplitud puede ser un **vector**, etc.

A continuación se estudiarán los factores que llevan a la definición de estas estructuras básicas y sus principios relacionales.

Estructuras de Datos y Estructuras Musicales

Partiendo de las micro estructuras, para generar, almacenar o representar sonido se necesita una sucesión de muestras a período constante que pueda luego ser convertida a energía eléctrica y sonora respectivamente. Es aquí donde se encuentra la primera **abstracción material** básica: la *sucesión en tiempo constante* (representada por la primitiva **vector**). Esta sucesión ya puede estar representando un **material musical**, un sonido simple o complejo de duración variable. Esto sería una **abstracción material** básica puesto que puede ser convertida en sonido y, lo más importante, puede ser transformada o compuesta junto con otras mediante diversos procesos.

Etimológicamente la palabra composición significa “adición”, sin embargo, a esta primera acepción podrían agregarse las de “transformación”, “agrupamiento” y “relación” en referencia a la composición musical.

Una sucesión en tiempo constante, además de representar el sonido de forma directa puede ser relacionada o transformada (procesada) mediante algoritmos que generan variaciones o, dependiendo del grado de variación, materiales diferentes. Según los procesos aplicados, el resultado puede ser entendido como un **material compuesto** o reestructurado. Como material compuesto, la entidad resultante es el conjunto que contiene el material original y los procesos aplicados sobre este (Figura 14).

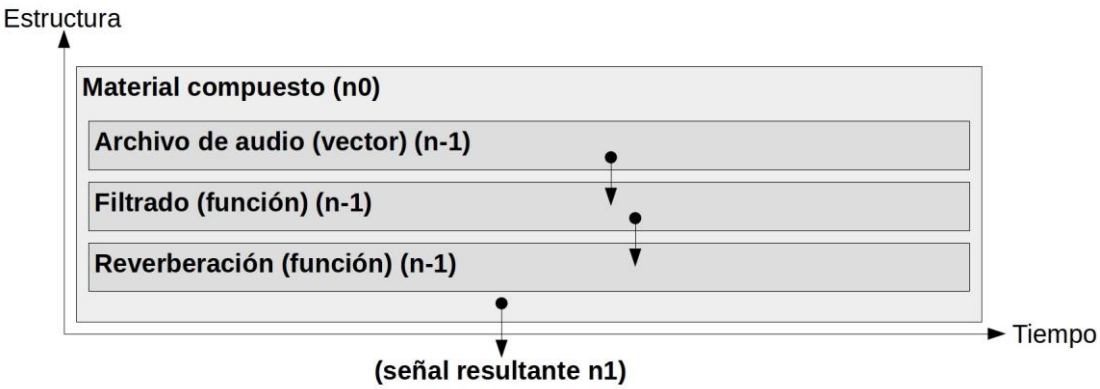


*Figura 14 Ejemplo de material compuesto por una sucesión en tiempo constante (**vector**), un algoritmo de filtrado (**función**) y un algoritmo de reverberación (**función**). La estructura representada en el eje vertical corresponde a la cadena de procesamiento interna del material compuesto el cual engloba la señal resultante (**vector**).*

El desarrollo de este material compuesto puede ser expandido mediante el **principio de agrupamiento**, pudiendo formar parte de una sucesión o un conjunto de eventos (ver elemento “Conjunto de eventos” en Figura 15). El agrupamiento (**sucesivo**, **simultáneo** o **mixto**, como se verá más adelante) es otro de los principios que genera materiales

compuestos a mayor escala estructural. A su vez, sobre este agrupamiento resultante pueden actuar nuevas sucesiones en tiempo constante como funciones de control paramétrico, por ejemplo, una variación dinámica de frase (elemento “Envolvente” en Figura 15). Incluso, en la elaboración de una textura musical, el desarrollo de los eventos internos puede estar definido por procedimientos que afecten múltiples parámetros de sus componentes (aumentando la complejidad dinámica del material compuesto).

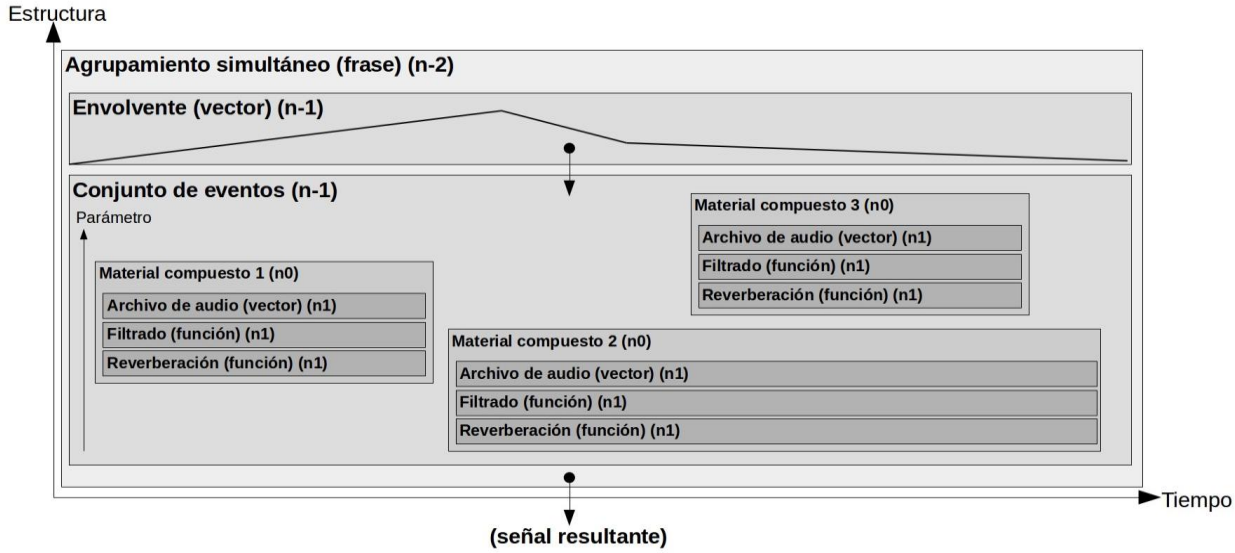


Figura 15 Principio de agrupamiento de estructuras básicas. Los materiales compuestos derivan de la definición en la Figura 14 a los que se les asigna el nivel base (n_0) y en relación a este se componen las estructuras de más alto nivel. En el eje vertical, el agrupamiento n_{-2} representa la estructura de los elementos contenidos mientras que los agrupamientos n_{-1} representan el ordenamiento paramétrico.

Lo que define un agrupamiento y lo diferencia de otros es la relación entre sus componentes internos. En sus formas más elementales, los agrupamientos pueden contener *sucesiones* de eventos (de duración fija o variable), *superposiciones* de elementos de igual duración o pueden ser *conjuntos* de elementos que se superponen, imbrican, yuxtaponen o separan52. Estos elementos básicos pueden actuar como contenedores o como contenido a medida que la escala temporal se aleja del período de discretización y las estructuras musicales se vuelven más complejas. Esta es una cualidad muy importante puesto que permite elaborar infinidad de materiales complejos empleando *recursivamente* las mismas abstracciones.

Es aquí donde entra en juego el concepto de *nivel simbólico variable*. El *nivel simbólico* es entendido entonces como el **nivel base** de las abstracciones según la definición de esta **Tesis** (véase el apartado teórico correspondiente en el **Capítulo 4**). Las abstracciones a **nivel base** son determinadas arbitrariamente por el compositor y son

empleadas para elaborar materiales musicales más complejos. Estas abstracciones actúan como los objetos combinables en la construcción de un determinado discurso musical.

Es posible cambiar de **nivel base** según se quieran elaborar micro o macro estructuras, las cuales dependen de la elaboración de materiales a un nivel base inferior. También es posible emplear abstracciones base predefinidas, ya sea en una etapa anterior de un mismo trabajo o adoptando las abstracciones de trabajos previos e incluso de otros compositores o de un entorno en particular. Por ejemplo, al emplear una librería específica, ya sea de síntesis granular, modelado físico o simplemente una librería de sonidos, se está elaborando un discurso propio a partir del nivel de abstracción desarrollado por un tercero.

Estructuras temporales y atemporales

La gran mayoría de los entornos para síntesis de sonido y composición algorítmica/asistida diferencian dos tipos de estructuras, las *definiciones de instrumentos* y los *secuenciadores*. Las *definiciones de instrumentos* (o *patch* según la denominación del entorno), se representan generalmente de manera atemporal a diferencia de los materiales dispuestos de forma concreta. Describen el comportamiento de un sistema instrumental que necesita de parámetros de control externos para ser puestas en el tiempo, para lo cual usan distintos mecanismos de secuenciación ya sea en tiempo real o en tiempo diferido.

En lenguajes de programación como *SuperCollider* (McCartney 2002) o *ChucK* (Wang 2003), entre muchos otros, es posible implementar *definiciones de instrumentos* dinámicamente, mediante los recursos del lenguaje de programación, empleando abstracciones de más alto nivel. Sin embargo, esto no cambia el hecho de que las **abstracciones instrumentales** de bajo nivel sean en esencia atemporales. Téngase en cuenta que el eje de este trabajo es demostrar como las mismas abstracciones materiales pueden ser empleadas a distintas escalas temporales y estructurales y, para ello, es necesario tener en consideración el factor temporal en todos los niveles. En relación a esto, Eckel y González-Arroyo (1994) desarrollan la noción de *concepto sonoro*:

> "[...] *un **concepto sonoro** puede ser considerado como una estructura dinámica compuesta, en la que las leyes de comportamiento y las configuraciones de procesamiento de señales se combinan para definir un objeto capaz de ser visto tanto como una entidad de producción de sonido a la vez que como un objeto lógico musicalmente significativo.*" (traducción y negritas por el autor de esta **Tesis**).

La noción de concepto sonoro afecta directamente las concepciones que se tienen sobre el sonido y los materiales musicales, las cuales, a su vez, afectan las posibilidades de manipularlos:

> "[...] *la composición de la síntesis sonora será afectada por, y afectará, niveles*

lógicos más altos del proceso compositivo. De hecho, estos pueden estar entremezclados de tan diversas maneras como para que no haya forma de establecer a priori un límite entre el nivel de definición de un objeto sonoro y el de su manipulación musical. Esto tiene una consecuencia importante: no podemos asumir un único paradigma de conceptualización, tal como la distinción conceptual más popular entre instrumento y partitura. [...]" (Ídem).

De lo cual concluyen:

"[...] *que queremos ver integrado en un todo a todas las acciones [que van] desde el micro-control de los módulos de procesamiento de señales hasta la composición de la partitura.*" (Ídem).

Es discutible, aún en la actualidad, qué grado de integración pretenden lograr diversos entornos respecto de este tema. Esto se debe a las distintas nociones que cada uno implementa, las cuales son entendidas como los recursos instrumentales provistos para la realización de las abstracciones informáticas que actúan como nivel base.

Los lenguajes de programación de propósito general son la herramienta que posibilita la "*integración de todas las acciones*", puesto que con los mismos elementos representan las abstracciones temporales y atemporales, como **objetos** y como **procesos**. Por otra parte, el diseño específico de cada entorno o librería de programación, manifiesta algún tipo de diferencia entre las **abstracciones instrumentales** y las **abstracciones compositivas** y sus capacidades de interacción. Para poder visualizar todas las acciones de manera integrada es necesario recurrir a un meta-modelo que permita analizar los componentes de manera generalizada, que contemple las cualidades de las **abstracciones instrumentales** y las **abstracciones compositivas**, y que sea independiente de diseños de *software* específicos.

A continuación se presenta un ejemplo práctico que manifiesta la diversidad de representaciones simbólicas y abstracciones implicadas en la composición sonora y musical. En la Fórmula 1, se presenta una expresión matemática de lo que podría ser una *definición de instrumento* que genera la superposición de tres movimientos armónicos simples.

$$frec = \{440, 880, 1760\}\ fase = \{0\pi, 0.1\pi, 0.9\pi\}\ pamp = \{0.5, 0.25, 0.175\}$$
$$amp = 0.1$$
$$f(t) = amp \cdot \sum_{n=1}^{3} pamp_n \cdot \sin\left(2\pi \cdot frec_n \cdot (t/R) + fase_n\right)$$

Fórmula 1 Expresión matemática de una definición de instrumento.

Aunque la variable independiente *t* sea entendida como la representación del tiempo,

el planteo matemático de la Fórmula 1 es atemporal porque se define de manera universal, no está relacionado a un contexto específico y por lo tanto no se puede saber cuando empieza ni cuanto dura. Si pasamos a una definición de síntesis como en el ejemplo Código 13, podemos apreciar que se mantienen un cierto paralelismo entre la representación matemática y la representación algorítmica. Si bien se definen parámetros de control, osciladores y buses de salida (**abstracciones instrumentales** del entorno *SuperCollider*) la definición sigue sin tener una delimitación temporal. Es simplemente una abstracción que necesita de otra que actúe como secuenciador.

```
(SynthDef(\aditi, {
  arg frec = #[440,
  880, 1760], fase
  = #[0pi, 0.1pi,
  0.9pi], pamp =
  #[0.5, 0.25,
  0.175], amp = 0.1;
  Out.ar(0, SinOsc.ar(frec, fase, pamp).sum * amp);
}).add;
)
```

Código 13

```
(
fork {
  3.do {
        x = Synth(\aditi);
        1.wait;
        x.free;
        1.wait;
  }
};
)
```

Código 14

En el ejemplo Código 14 se expone una rutina de control. Un programa que cuando se ejecuta se encarga de generar tres eventos sonoros relacionados temporalmente. Recién en este nivel de abstracción se tiene en cuenta el factor temporal a nivel musical. Estos elementos generados pueden ser representados temporalmente mediante su forma de onda, como se muestra de manera esquemática en la Figura 16. Sin embargo, la forma de onda no da cuenta de las abstracciones materiales que fueron utilizadas para su creación. En la Figura 17 se muestra, en comparación, una representación tanto de las abstracciones materiales generadoras como de la delimitación temporal resultante.

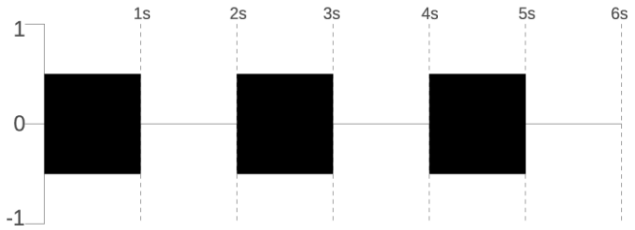


Figura 16 Eventos sonoros del ejemplo Código 14 representado como forma de onda (amplitud en función del tiempo).

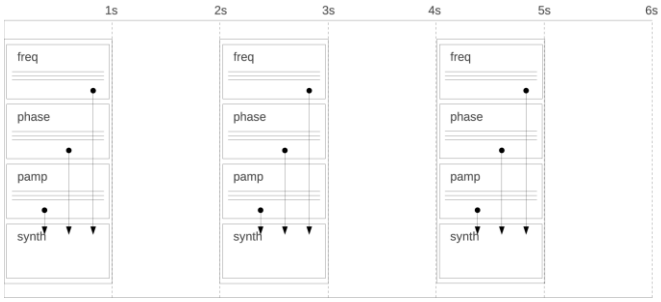


Figura 17 Eventos sonoros del ejemplo Código 14 representados estructuralmente en función del tiempo.

Es evidente que, por motivos prácticos, no es posible visualizar toda la información interna del **proceso de generación** del material sonoro ni tampoco es necesario representar todas las **abstracciones instrumentales**. Sin embargo, con respecto a esto se tienen en cuenta las necesidades de poder visualizar los elementos significativos al nivel temporal y de estructura que se quiere trabajar, y la posibilidad de cambiar dinámicamente entre la visualización de distintos niveles. Por ejemplo en la Figura 18,

se ve como el material compuesto podría representarse de una manera muy simplificada si estuviera, por ejemplo, formando parte de una textura compleja, generada dentro de un algoritmo de síntesis y lo que se quisiera visualizar fuera la distribución general de dicha textura en el tiempo.

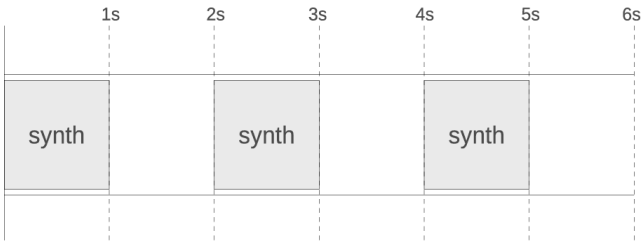


Figura 18 Representación abstracta de la distribución temporal de un material complejo generado por un algorítmo de síntesis (denominado synth en la figura).

La representación gráfica es solo una forma de representación que resulta más práctica para la visualización y manipulación de cierto tipo de información. El concepto de integración de estructuras materiales puede ser implementado mediante código de programación, el cual resulta ser un recurso mucho más versátil a la hora de aplicar procedimientos complejos. Sin embargo, estos pueden ser representados gráficamente de manera abstracta (como se muestra en la Figura 18) si se quisiera visualizar la resultante de dicho proceso o la relación temporal entre este y otros materiales.

El **material lógico**, considerado como estructura atemporal y expresado como *definición de instrumento* o **interfaz instrumental**, necesita de un secuenciador o de la interacción en tiempo real para poder concretarse como se explicó anteriormente. Sin embargo, su lógica interna implica relaciones de orden temporal y, en la práctica, un período de procesamiento. Estos factores pueden ser considerados fuera del tiempo musical o como unidades temporales dependiendo del **nivel base** al que se trabaje compositivamente.

Por ejemplo, el siguiente código de programación en C (Código 15)53 genera una secuencia sonora mediante la manipulación de bits:

```
main(t){for(t=0;;t++)putchar(t*(((t>>12)|(t>>8))&(63&(t>>4))));}
```

Código 15

El elemento que actúa como secuenciador está integrado con el algoritmo de procesamiento de forma computacionalmente minimalista (véase Heikkilä 2010 y

Schmidhuber 1997). El ciclo iterativo *for* actúa como generador de sonido a nivel de muestra musical y secuenciador a nivel de nota musical mediante la combinación de tres operaciones de corrimiento de bits sobre un contador (la variable *t*). De manera similar al ejemplo de Nathaniel Virgo (ver **Capítulo 3**), el código de programación define procesos isorrítmicos cuya combinación polirrítmica genera la forma de onda y define los patrones de variación de altura. Este ejemplo es interesante puesto que no expresa **abstracciones instrumentales**, solo la lógica de los **procesos** a un único nivel de abstracción, y su realización depende del ordenador simplemente como herramienta de propósito general. Su lógica matemática y arquitectónica genera sonido a partir de *bits* que se direccionan a un transductor de salida mediante las abstracciones del sistema operativo que manipulan los periféricos de salida. El tiempo cronológico está implícito y depende del reloj al que se ajuste el *stream* de los datos de salida.

En un procedimiento de síntesis granular, el archivo de audio "fuente" no está representado temporalmente al mismo nivel que los materiales concretos resultantes sino como parte de la **abstracción instrumental**. Es decir que, siendo la representación de un **material concreto** no "ocupa" tiempo por sí mismo. El *buffer* actúa simplemente como dato de entrada del algoritmo de síntesis y, por lo tanto, podría ser entendido como componente del **material lógico**, puesto que no define relaciones temporales lineales al material resultante, es decir, que el tiempo representado por el archivo de audio no es empleado directamente en relación al tiempo de la obra sino mediante transformaciones algorítmicas. Es importante notar que el *buffer*, en este caso, estaría siendo considerado como **material lógico** *en relación* al **material generado** mediante un proceso algorítmico. Este ejemplo demuestra un caso específico que puede resultar particularmente ambiguo. Sin embargo, el archivo de audio fuente contiene información temporal concreta que se emplea para generar los granos. Aunque sea procesado de diversas maneras, un grano es una sucesión temporal que se recorta y extrapola del archivo fuente. La síntesis granular es, en este sentido, un método de síntesis que emplea procesos simples de edición de audio de manera automática y, por lo tanto, el *buffer*, como dato de entrada, es un material concreto. Aunque forme parte de la **abstracción instrumental**, se puede decir que es empleado como **abstracción compositiva** dentro del proceso automático de síntesis.

Un caso extremo, que se desprende del ejemplo anterior, sería la definición de una unidad generadora que procese un *buffer* como conjunto de datos de entrada y genere una señal seleccionando aleatoriamente muestras individuales. El procedimiento es extremadamente similar. Pero en este caso la información temporal de *buffer* es irrelevante, las muestras podrían estar ordenadas de cualquier manera dentro del mismo archivo fuente. En este caso, en oposición al ejemplo anterior, el *buffer* es simplemente una manera de disponer los datos al implementar un proceso algorítmico.

Para poder contemplar la "*integración de todas las acciones*" (Eckel y González-Arroyo 1994) es necesario entender que conviven **distintos planos**. Según desde qué plano se esté trabajando, ya sea desde las **abstracciones instrumentales** o las **abstracciones compositivas**, un mismo elemento material, representado mediante una única abstracción informática, puede ser entendido como **lógico** o **concreto**. Para poder determinar a qué categoría pertenece es necesario analizar sus cualidades temporales y su significado dentro de los procesos algorítmicos.

Estos ejemplos demuestran que el **nivel base** que se elige para componer una pieza electroacústica, varía con respecto a las **abstracciones instrumentales** y la **escala temporal** en la cual los materiales se desarrollan. El contexto de la realización puede variar considerablemente y define los rasgos de la representación y la manipulación sonora, los cuales, a su vez, pueden afectar el resultado estético. El empleo de **abstracciones instrumentales** y de **abstracciones compositivas**por separado es posible, pero no estrictamente necesario, y se suele adoptar implícitamente como convención. El análisis de una pieza musical electroacústica debe tener en cuenta principios más generales y analizar el contexto en el cual se desarrollan.

Diferentes agrupamientos de estructuras simples

Otro rasgo que define distintas concepciones materiales es la capacidad de agrupar parámetros (que pueden ser tanto **objetos** como **procesos**) de diferentes maneras. Esto también es importante porque según como se agrupen los datos, ya sea en estructuras visuales o estructuras de datos dentro de un lenguaje de programación, las posibilidades de manipular las abstracciones de más alto nivel se ven afectadas. En la Figura 19 se muestran dos abstracciones que agrupan parámetros de control y osciladores. Tanto los osciladores como los parámetros de control son entendidos como funciones delimitadas temporalmente. En la abstracción A del gráfico se ve como los parámetros de control *frec* están agrupados por un lado y las funciones generadoras *osc* están agrupadas por otro. Los índices (1, 2 y 3) son los que asignan internamente los parámetros de control con los osciladores y la salida debe ser entendida como la suma de las señales de audio resultantes. En la abstracción B se pueden apreciar los mismo elementos pero agrupados de a pares como parámetro de control y oscilador, la salida debe ser entendida también como la suma de las tres señales resultantes.

Ambas abstracciones en la Figura 19 puede estar generado tanto un movimiento armónico complejo, si los valores de los parámetros de control están en relación armónica, como tres movimientos melódicos independientes, si los valores de los parámetros de control adquieren otro tipo de desarrollo (ver **estructura subjetiva** en **Capítulo 3**). En el

primer caso, la resultante sonora percibida refiere una sola entidad, mientras que en el segundo estaría involucrando a tres. Es evidente que resulta más conveniente la abstracción A para manipular un movimiento armónico complejo y la abstracción B para controlar tres movimientos armónicos simples, sin embargo, ambos agrupamientos son capaces de generar las mismas resultantes.

Respecto de la adecuación de los agrupamientos según la resultante sonora (como un solo objeto en el grupo A o como tres en el grupo B de la Figura 19) podría quererse establecer una convención, pero el segundo caso podría ser considerado también como un solo objeto material agrupado. Es posible concebir tres movimientos melódicos simples como componentes de un solo objeto sonoro resultante si a estos se les aplica algún tipo de restricción registral como, por ejemplo, que se muevan dentro de una banda crítica y que esta banda crítica a su vez vaya variando registralmente. Más aún, la resultante sonora percibida podría concebirse como la transformación entre una única resultante sonora compleja y varias simples en su desarrollo temporal.

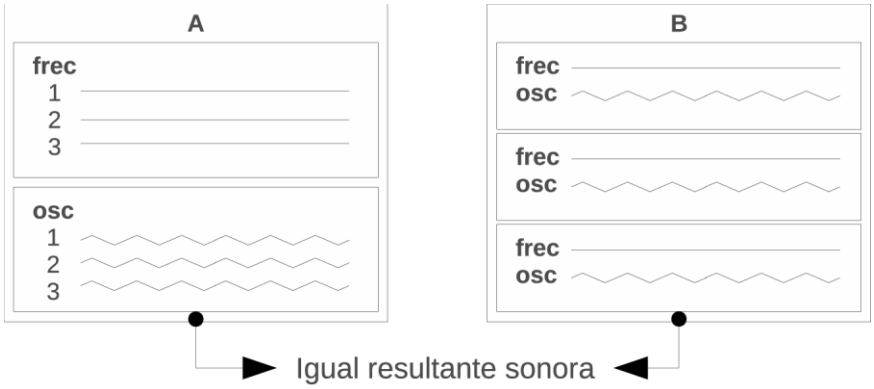


Figura 19 Agrupamientos equivalentes de estructuras simples.

Es bien sabido por los compositores que los procedimientos compositivos no necesariamente se correlacionan directamente con la resultante percibida. El ejemplo anterior resalta el hecho de que lo mismo sucede tanto a escala de composición sonora como formal. Un ejemplo de más alto nivel podría ser la orquestación de uno o varios materiales que varíen el timbre y la densidad de la resultante sonora de manera tal que la delimitación de los materiales “internos” se vuelva confusa.

Lo importante de este análisis es hacer notar que **los agrupamientos de estructuras simples pueden adquirir significados múltiples**. Que los elementos básicos del sistema de representación sean acotados no quiere decir que los posibles materiales musicales resultantes también lo sean. Por otra parte, se puede comprobar que ciertos agrupamientos son más adecuados para determinadas estructuras pero también que si se cambia la concepción compositiva es necesario volver a agrupar los parámetros

de manera diferente. La representación misma de entidades sonoras es constantemente dinámica.

Este dinamismo surge del pensamiento compositivo y se puede ver afectado si las estructuras de un entorno de composición restringen los posibles materiales a comportamientos o agrupamientos estandarizados.

En resumen:

- Los mismos agrupamientos estructurales pueden generar resultantes distintas.
- Distintos agrupamientos estructurales pueden generar resultantes equivalente.
- El agrupamiento estructural elegido puede adecuarse mejor a la manipulación de un tipo de resultante.
- Los procedimientos compositivos no necesariamente se correlacionan con la resultante percibida.
- Los agrupamientos estructurales pueden adquirir significados múltiples.

Teniendo en cuenta estas propiedades de las estructuras musicales y el marco teórico desarrollado en capítulos anteriores, en el apartado siguiente se desarrollarán, primero las propiedades temporales de las primitivas y luego las primitivas en sí mismas (**evento**, **lista**, **vector**, **conjunto** y **función**) del sistema de representación propuesto por esta **Tesis**. Es importante notar que el **principio de agrupamiento** y sus propiedades lógicas, estructurales y perceptivas es lo que define la cualidad de un **material musical** expresado informáticamente y no una definición estática como estructura de datos. Las primitivas de representación serán enfocadas como **abstracciones compositivas** que también pueden ser empleadas como **abstracciones instrumentales**.

Materiales musicales como abstracciones compositivas

Según el marco teórico de esta **Tesis**, los materiales musicales existen dentro de un rango temporal y pueden ser agrupados o compuestos a distintas escalas. Empleando la clasificación de Roads (2002) este rango va desde la *escala de muestreo* a las escalas *macro* y *supra* temporal. Los materiales pueden volverse entidades más complejas a medida que se aumenta la escala temporal, pero a medida que se acercan al límite inferior estos se vuelven inevitablemente más simples hasta llegar al nivel de la muestra de audio. Por lo tanto, no es posible definir de manera concisa los materiales musicales a escala *macro temporal* pero sí es posible definir los elementos combinables básicos cercanos a la *escala de muestreo*.

Como se vio en el **Capítulo 1**, restringir los recursos combinables (los elementos

empleados para la composición sonora) es una manera de abarcar el dominio del problema. En este caso, el dominio del problema es la generalidad de las estructuras empleadas para la composición musical empleando medio digitales. Las **restricciones compositivas** pueden ser **impuestas** (por el sistema) o **voluntarias** (elegidas por el compositor). Las **restricciones voluntarias** son un sub-conjunto de las **restricciones impuestas** que se definen en relación a estas últimas. Por lo tanto, si lo que se pretende es elaborar un meta-sistema de **restricciones voluntarias**, es necesario acotar los elementos constitutivos de manera tal que permita el mayor grado de generalidad posible en cuanto a la capacidad de relacionar y generar nuevas estructuras.

Por lo tanto, para definir sistemáticamente las entidades que componen el modelo propuesto por esta **Tesis**, es necesario analizar sus propiedades generales y sus posibles comportamientos como estructuras combinables en relación al **principio de agrupamiento**. Luego de realizada esta tarea se procederá a analizar los rasgos que ameritan la definición de las estructuras particulares: **evento**, **lista**, **vector**, **conjunto** y **función**.

Propiedades, características y tipos de relaciones temporales

Los elementos que representan entidades materiales básicas, como **materiales lógicos** o **concretos**, definen dos ***propiedades temporales fundamentales***: los valores de **inicio** y de **duración**. Para poder definir estas propiedades, los materiales deben estar contenidos dentro de un ***contexto temporal*** (Bresson 2007). **El contexto actúa como la e**

Relativa al **contexto**, la propiedad de inicio define el comienzo o la posición de un elemento mientras que la duración define el lapso temporal, relativo al comienzo, que el elemento ocupa. Esto es simplemente una manera de medir un hecho temporal con la menor cantidad de parámetros posibles de manera determinista y flexible.

El protocolo MIDI (Penfold 1992), por ejemplo, define un evento de inicio y un evento de finalización de las notas musicales (*note on* y *note off* respectivamente) en relación al número de nota MIDI. Esta representación no explicita la duración del objeto-nota, la cual se deriva de la diferencia entre el tiempo de inicio y el de finalización. Esto se debe a que el protocolo MIDI fue ideado para modelizar las **acciones instrumentales** tradicionales, las cuales, a su vez, pueden ser empleadas para representar los materiales musicales. Sin embargo, las acciones MIDI dependen de los programas (instrumentos MIDI), los cuales pueden o no realizar sonidos prolongados indefinidamente en el tiempo y, en caso negativo, ignorar el evento *note off*.

Definir las propiedades temporales como tiempo de inicio y duración posibilita la

representación de **materiales concretos**. En caso de ser necesaria la representación de acciones instrumentales, e.g. del tipo MIDI, se puede recurrir a lo que en este modelo se denomina ***característica temporal***.

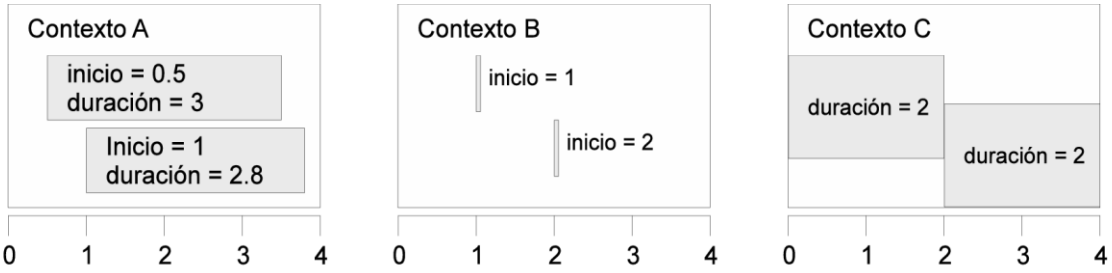


Figura 20 Características temporales (el eje de las abscisas representa la duración). El contexto A contiene eventos con inicio y duración, el contexto B contiene elementos solo con inicio y el contexto C contiene elementos solo con duración.

La **característica temporal** de una entidad material **es la relación entre las propiedades de inicio y duración**. El conjunto de características temporales concretas (Figura 20) se define según las posibles combinaciones de presencia y ausencia de las propiedades temporales. Los materiales pueden tener inicio y/o duración, o ninguno de los dos, lo cual da un total de cuatro características diferentes enumeradas a continuación y expresadas lógicamente en la Tabla 2.

1. Si define inicio y duración está representado un segmento temporal relativo al contexto.
2. Si define solo inicio puede estar representado una acción o valor puntual relativo al contexto.
3. Si define solo duración también está representado un segmento temporal pero dependiente de otros factores del contexto al que pertenece, e.g. el tiempo de inicio se manifiesta de manera implícita mediante la suma total de las duraciones previas en una sucesión de eventos (véase más adelante).
4. Si no define inicio ni duración es considerado como un evento aislado de contexto.

Si denominamos ***evento54*** al tipo de dato o **material básico**, este puede pertenecer o no a un **contexto**. Si pertenece a un contexto es una entidad temporal concreta, de lo contrario es una entidad no predecible temporalmente. Un evento temporalmente indeterminado sirve para representar **acciones externas** (del tipo de las convenciones instrumentales) o **relaciones abstractas** como recurso del pensamiento compositivo.

Inicio	Duración	Característica
V	V	Segmento temporal
V	F	Evento puntual
F	V	Segmento relativo
F	F	Relación abstracta/Evento indeterminado

*Tabla 2 Posibles **características** según las relaciones lógicas de las propiedades de inicio y duración.*

Además de las ***propiedades temporales*** y las ***características temporales***, el tercer y último factor temporal que afecta la definición de los materiales abstractos son *los **tipos de relaciones temporales básicas***. Las relaciones temporales se producen entre los **materiales básicos** al ser agrupados dentro de un mismo **contexto**. Estas pueden ser catalogadas de distintas maneras según los factores que se tengan en consideración (Allen 1983) (Keramane 1995) (Hanappe 1999) (Bresson 2007). En este trabajo se rescata la concepción descriptiva de las entidades temporales (Allen 1983) y se descartan, como pertenecientes a las estructuras de datos, las relaciones de carácter lógico/explícito como, por ejemplo, las relaciones temporales explícitas (Honing 1993), la causalidad (Keramane 1995) y otros factores relacionados con los procesos generativos (Hanappe 1999) (Bresson 2007). Esto se debe a la concepción del **material concreto** como entidad independiente de los procesos generadores o modificadores (**materiales lógicos**). Sin embargo no se sigue la clasificación propuesta por Allen (1983) sino que se simplifica a tres tipos de relaciones fundamentales definidas como propiedades de las estructuras de datos básicas.

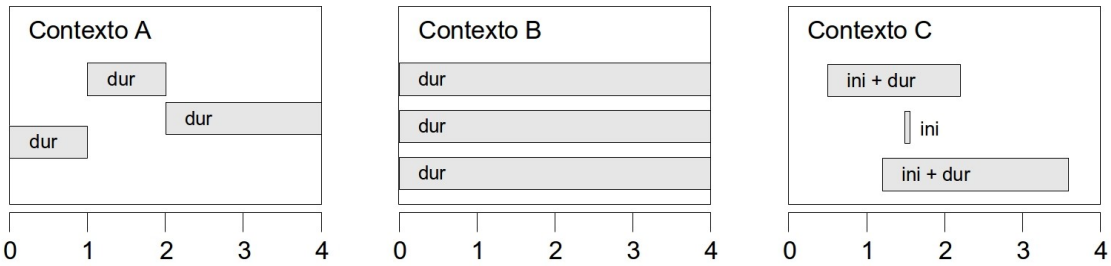


Figura 21 Tipos de relaciones temporales (el eje de las abscisas representa la duración). Los elementos del contexto A están en relación sucesiva, los elementos del contexto B están en relación simultánea y los elementos del contexto C están en relación mixta.

La relación temporal posible entre los materiales agrupados puede ser de tres tipos: ***sucesiva***, ***simultánea*** o ***mixta*** (ver Figura 21):

1. La **relación sucesiva** es solo sucesiva, los elementos relacionados solo definen su duración, no se superponen y comienzan al finalizar el evento anterior.
2. La **relación simultánea** es solo simultánea, todos los elementos relacionados comienzan y terminan al mismo tiempo.
3. La **relación mixta** es la que representa cualquier combinación posible de las dos relaciones anteriores. Los elementos contenidos tienen necesariamente inicio y pueden tener duración.

Las relaciones de tiempo explícito (Honing 1993) que se producen dentro de un contexto de relaciones mixtas no son representadas por esta clasificación salvo en los casos de sucesión y simultaneidad.

Relación temporal	**Característica temporal** de los eventos contenidos	**Propiedades temporales** de los eventos contenidos
Sucesiva	Segmento relativo	Duración
Simultánea	Segmento temporal único / Evento indeterminado / Evento puntual	Inicio + Duración / Indeterminado / Inicio
Mixta	Segmento temporal / Evento puntual	Inicio + Duración / Inicio

Tabla 3 Características y propiedades temporales que definen la realción temporal.

Al observar la Tabla 3, es importante notar que la relación simultánea es una propiedad de los contextos indeterminados, los eventos puntuales y los segmentos temporales únicos. Esto es el punto de conexión que permite que los agrupamientos de las estructuras básicas expuestas a continuación puedan ser **recursivos**. Es decir, que la **relación simultánea puede ser reinterpretada como un contexto vacío**.

Según las **propiedades**, **características** y **tipos de relaciones temporales** de los elementos contenidos, se puede definir un conjunto finito y complementario de tipos de **materiales básicos como estructuras de datos más específicas** sin perder la generalidad pretendida por este **modelo**.

Primitivas del Sistema de Representación

Material como contexto

En base a la clasificación previa, y teniendo en cuenta consideraciones de carácter

práctico, se pueden diferenciar cuatro tipos básicos de **materiales concretos** como estructuras de datos: ***evento***, ***lista***, ***vector*** y ***conjunto***. Con respecto a la representación de los **materiales lógicos** se define el elemento ***función*** como la abstracción relacionada, según sus cualidades temporales o atemporales, con los **materiales concretos**.

A continuación se enumeran las estructuras de datos según sus cualidades al actuar como **contextos** puesto que su definición se da en relación a los **materiales concretos**. En la sección siguiente se analizarán sus cualidades al actuar como **parámetros**.

Evento

El **evento** es una estructura de datos que generalmente se emplea para agrupar parámetros dentro de una acción o segmento único. Con respecto a sus **propiedades temporales** (ver tabla Tabla 3) un evento puede ser un segmento temporal, un evento puntual, un segmento relativo o una relación abstracta/evento indeterminado.

Cuando un **evento** actúa como **contexto**, es una estructura que manifiesta un segmento temporal cuyos parámetros internos son de **relación temporal** simultánea.

Un evento puntual (sin duración) puede ser, por ejemplo, una instrucción que representa una acción en un momento determinado dentro de un contexto. Los **eventos** sin duración son equivalentes a los "*discrete events*" de Dannenberg, Desain y Honing (1997 pg. 281).

Al actuar como segmento relativo (sin inicio) un evento puede ser el componente de una **lista** (véase a continuación).

Un evento indeterminado actúa como parámetro de entrada que depende de factores externos, generalmente realizados en tiempo real, de los cuales no se puede prever su organización en relación al contexto de los materiales concretos.

Lista

Una **lista**, como contexto, define una **relación temporal** sucesiva de segmentos relativos. Las listas se emplean para representar secuencias solo sucesivas, entendidas generalmente como señales de control continuas.

En el entorno *SuperCollider,* las *unidades generadoras* equivalentes que actúan como **abstracciones instrumentales** del servidor, las *unidades generadoras de frecuencia a demanda* (*demand rate ugens* en inglés) se adecuan perfectamente a la definición este tipo de estructura básica. Sin embargo, es necesario notar que no todas las implementaciones resultan unívocas respecto de estos conceptos. Por ejemplo, la librería de *patterns* de *sclang* trabajan principalmente sobre este tipo de dato, proveyendo una

gran cantidad de recursos de generación, relación y procesamiento de **listas**. Pero los eventos contenidos en un *event stream* no necesariamente empiezan y terminan por yuxtaposición, sino que pueden superponerse o estar separados por silencio si el tiempo de *sustain* de los eventos es mayor al tiempo *delta* que los separa. Según la clasificación que implica este **modelo**, los *event stream* de *SuperCollider* serían un tipo especial de **conjunto** (ver más abajo) que se comporta como una **lista55**.

Al emplear una clasificación tan general como la propuesta por este modelo, es posible que surjan casos "intermedios". Sin embargo, los principios son generales y pueden ser empleados para clasificar cualificaciones híbridas, las cuales pueden quedar a criterio del analista o desarrollador.

Vector

El **vector** (que es en la práctica del audio digital un *buffer* de muestras de audio) contiene elementos que no tienen inicio como **propiedad temporal** y su duración es constante, sus elementos se definen temporalmente de manera sucesiva por su posición dentro de la estructura de datos (estructura **sucesiva** como contexto).

El **vector** es equivalente a una **lista** de elementos de duración constante. En la práctica, su implementación como estructura de datos y la lógica de control son diferentes a las de las listas, lo que influye en las formas de manipulación posibles por parte del compositor. Por ejemplo, las listas se suelen manipular considerando a cada elemento contenido con cierto grado de importancia individual (e.g. una nota musical), mientras que los **vectores** se suelen manipular como conjuntos de datos (e.g. una señal de audio).

Para Dannenberg, Desain y Honing (1997 pg. 281) el **vector** es la estructura de datos que representa la información continua (ver sus cualidades como parámetro más abajo). Por su importancia fue implementado, por ejemplo, en *Nyquist* (Dannenberg 1997), como estructura de *primera clase*, lo que significa que puede ser manipulada al mismo nivel de abstracción/representación que las demás estructuras de datos básicas del lenguaje.

En general, los **vectores** son empleados para representar señales de audio o de control a cualquier escala temporal. La cualidad importante que lo define como entidad independiente es justamente su modo de empleo como estructura de datos básica para representar señales.

Conjunto

El **conjunto**, como **contexto**, contiene elementos cuya relación temporal es mixta y,

por lo tanto, es el tipo de estructura que posibilita las combinaciones de las estructuras sucesivas y simultáneas.

Los elementos constitutivos de un **conjunto** son los segmentos temporales y los eventos puntuales. Es la abstracción que actúa como "línea temporal" en prácticamente la totalidad de los entornos existentes. Por ejemplo, las pistas o canales de audio y MIDI en las DAW (*digital audio workstation* denominados como editores lineales multipista en esta **Tesis**) y las pistas o canales de los secuenciadores. Dentro de las pistas de audio, los archivos de audio generalmente se pueden solapar, de manera relacionada mediante un fundido cruzado (*crossfade*) o simplemente como elementos superpuestos, en cuyo caso un archivo queda visualmente superpuesto a otro.

En la práctica, las abstracciones que representan **conjuntos** son extremadamente variadas.

Algunas de estas abstracciones proveen cierto grado de especificidad para un determinado conjunto de acciones u operaciones.

Entre las implementaciones existentes, la diferencia entre una **lista** y un **conjunto** puede ser ambigua. Como se mencionó en la descripción de las propiedades de la **lista**, los *patterns* en *SuperCollider* pueden ser entendidos como **conjuntos** que se comportan principalmente como **listas**. Otro caso sería el de los canales MIDI de los secuenciadores por bloques, como el *LMMS*56, donde los bloques de secuencias no se pueden solapar temporalmente y deben ser contiguos aunque se esté representando silencio (bloque vacío). De manera similar se comportan las estructuras de datos que representan entidades musicales en, por ejemplo, el entorno *Elody* (Letz et al. 1998). En estos casos la abstracción predominante es la de la **lista**, y los **conjuntos** se conforman mediante la superposición de listas.

Más allá de la adecuación de un entorno específico al **modelo** propuesto, el analista que emplea este **modelo** debe tener la capacidad de discernir entre los detalles de la implementación, los cuales pueden definir los recursos provistos para la manipulación, y el significado específico o predominante de las estructuras abstractas y cómo estos afectan la concepción compositiva que actúa como **restricción impuesta** por el *software*.

Función

Con respecto a los **materiales lógicos**, estos se definen como **funciones generadoras** que tienen las mismas propiedades temporales que los **eventos**, pueden estar aislados de **contexto** (fuera de cualquier estructura temporal) o pueden definir su inicio y/o duración dentro de una estructura contextual.

Las **funciones** pueden generar cualquier tipo de **material concreto** representado

por las primitivas anteriores, algoritmos de control sobre otros **materiales concretos** y **abstracciones instrumentales**.

Las **propiedades temporales** variables de los **eventos** y las **funciones** son, por un lado, el elemento que posibilita la relación entre la representación de **relaciones abstractas** y su **realización** (véase el apartado correspondiente en el **Capítulo 3**). Por otro parte, el **material concreto** generado por el **material lógico** puede ser de cualquier tipo.

Las **funciones**, son la representación de los **materiales lógicos** al mismo nivel de abstracción (al compartir las propiedades temporales de los **eventos**) que los **materiales concretos**. Si se define una **función** dentro de un **contexto**, temporalmente esta puede actuar como evento puntual, evento indeterminado o segmento temporal. Estas son las característica temporales de los materiales en estado lógico. Para poder conocer las características temporales de los **materiales concretos** generados mediante **materiales lógicos** es necesario evaluar la función produciendo el **cambio de estado** de material lógico a concreto.

Estas propiedades implican que las **funciones** se pueden secuenciar junto con los **materiales concretos** dentro del sistema de representación. Producir el **cambio de estado** solo es necesario cuando sea necesario realizar operaciones entre **materiales concretos** y **lógicos**, puesto que el resultado de las operaciones entre materiales solo se pueden producir si estos comparten el mismo estado. Las funciones deben ser evaluadas para conocer las propiedades concretas resultantes del **material generado**, principalmente, sus **propiedades temporales**.

Un ejemplo de secuenciación de funciones es el entorno *Blue* (Yi 2011) que emplea *Csound* como aplicación de síntesis. Los algoritmos generadores pueden ser ordenados en una línea temporal, a la manera de una pista de audio, y luego ser *renderizados* como el resultado concreto de la partitura en **estado lógico**.

Otro ejemplo de **materiales lógicos** expresados como funciones son las *unidades generadoras* y las *definiciones de instrumentos* (según el nivel de **abstracción instrumental** que se emplee). Una **función** que representa el **estado lógico** de una *unidad generadora* puede recibir un **material concreto** como parámetro, pero para obtener el resultado de la operación es necesario evaluar la **función** que representa la parte lógica de la composición de materiales. Las operaciones entre un **material concreto** y un **material lógico** producen como resultado un **material concreto** al ser evaluadas.

En síntesis, las **funciones** son los elementos que representan los **materiales lógicos** como **abstracciones compositivas**. Los **materiales lógicos** son los **materiales generadores** y el resultado generado por estos es un **material concreto**. Como los materiales en **estado lógico o concreto** pueden ser representaciones alternativas de

un mismo material musical, es posible emplear indistintamente una u otra según resulte más conveniente para el análisis o la manipulación compositiva.

Material como parámetro

Si se toman los valores representados por los materiales concretos como parámetros temporales, las señales generadas por los tipos básicos pueden ser ***continuas*** o ***discontinuas*** (Figura 22). Los objetos continuos son los que no tienen "huecos" en la señal generada durante el segmento temporal representado. Por ejemplo, si tomamos los valores de estos objetos como la evolución temporal de un parámetro, estos generan una función continua a frecuencia de audio o de control. Las estructuras básicas continuas son el **evento** (con duración), el **vector** y la **lista**. Los objetos discontinuos son los que pueden no generar señales continuas como valores de salida porque pueden no definir todos los valores de un segmento temporal (Figura 22). La estructura básica discontinua es el **conjunto**. El **evento** sin duración puede ser entendido como una estructura discontinua y la **función** puede generar tanto materiales continuos como discontinuos y estructuras jerárquicas (véase más adelante).

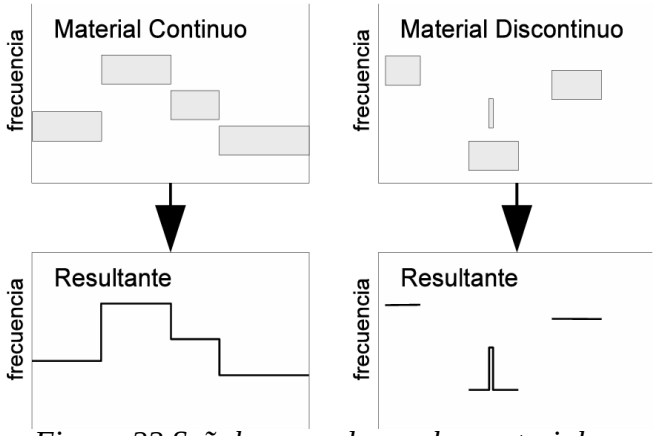


Figura 22 Señal generada por los materiales continuos y discontinuos. Los materiales discontinuos pueden expresar un mismo parámetro de manera ambigua (en la resultante del material discontinuo se tomó el valor más alto como valor de salida cuando los objetos se superponen).

Esta distinción entre estructuras continuas y discontinuas es útil para organizar la representación y relacionar parámetros musicales en estructuras jerárquicas como se describe a continuación.

Primitiva	Tipo de Material/Estado	Relación Generada	Señal Generada
Evento	Concreto	Simultánea	Continua / Discontinua
Lista	Concreto	Sucesiva	Continua
Vector	Concreto	Sucesiva	Continua
Conjunto	Concreto	Mixta	Discontinua[57]
Función	Lógico	Simultánea / Sucesiva / Mixta	Continua / Discontinua

Tabla 4 Estado del material, relación y señal generada en las distintas primitivas.

Principios de Agrupamiento

En la sección anterior se definieron las primitivas de representación que actúan como **materiales básicos**, considerados como **abstracciones compositivas e instrumentales**. En esta sección se trata sobre los **principios de agrupamiento** que generan **materiales compuestos**.

Los **principios de agrupamiento** aquí expuestos no son de carácter perceptivo sino estructural. Pero no refieren a la estructura analítica que se puede derivar a nivel semántico de las características de un discurso musical en particular como, por ejemplo, el análisis funcional/tonal o el análisis estructural schenkeriano (Forte 1998), sino, en cambio, refieren a la estructura relativamente neutra que deriva de la notación técnica compositiva empleada junto con los medios informáticos. Como fue desarrollado en el marco teórico de esta **Tesis** (**Capítulo 3**), tanto las **relaciones subjetivas** como las **relaciones objetivas** son en esencia ambiguas, por distintas razones, y dependen del análisis de factores que implican otros planos de conocimiento, musicales, acústicos, fisiológicos y culturales.

Por lo tanto, las estructuras basadas en agrupamientos definidas en esta **Tesis** y sus principios relacionales, dan cuenta de los procesos mentales que involucran la concepción técnica/musical de las abstracciones informáticas como medio de conocimiento. Pese a esto, los principios relacionales y, sobre todo, los agrupamientos resultantes tienen un cierto grado de coherencia con la teoría perceptiva.

Estudiar las concepciones compositivas, manifiestas en el desarrollo técnico, es una referencia lo suficientemente objetiva como para desarrollar análisis más profundos que involucren otros planos del conocimiento. Desde el punto de vista de la composición musical, estudiar, entender y dominar estas técnicas es un proceso que posibilita la libertad expresiva como capacidad de elección.

El **agrupamiento** de los materiales básicos es lo que genera los **materiales**

compuestos como abstracciones compositivas. Los agrupamientos se producen debido a una serie de principios relacionales derivados de las técnicas informáticas y compositivas los cuales se pueden dividir en dos categorías:

- **Relación estructural por agrupamiento compositivo**: se define como organización estructural en relación a las estructuras musicales derivadas de la concepción compositiva aplicada.
- **Relación de estructura por agrupamiento lógico**: se define por necesidades de procesamiento (e.g. abstracciones instrumentales, parámetros de control, direccionamiento de señales, propagación de parámetros a sub-estructuras, operaciones algorítmicas, etc.).

El primer tipo de **relación es estructural** genera **agrupamientos compositivos**. Los elemento agrupados jerárquicamente por medio de esta relación dan cuenta de una decisión compositiva que representa la estructura de materiales compuestos como agrupamientos organizativos. Los elementos contenidos en el agrupamiento tienen **relaciones temporales** peor no tienen **relaciones de procesamiento/lógicas**.

Un ejemplo de **agrupamiento compositivo** se puede apreciar en el elemento A de la Figura 13 (pg. 129). La relación estructural entre los elementos “Obra”, “Melodía” y “Evento-Nota” son simplemente temporales y demuestran la relación estructural que deriva de la sintaxis estructural teoría tradicional.

Los **agrupamientos compositivos** también pueden deberse a la elección de agrupamientos lógicos, pero en este caso se considera que la relación estructural es simplemente un agrupamiento lógico que deriva de una decisión o técnica compositiva. Sin embargo, un agrupamiento compositivo si afecta las posibilidades de manipulación de los materiales compuestos como entidades de mayor nivel de abstracción.

El segundo tipo de **relación es estructural** es el **agrupamiento lógico**. Difiere de la organización contextual/compositiva en que los elementos contenidos debe estar agrupados de determinada manera para poder relacionarse paramétricamente.

Un ejemplo de **agrupamiento lógico** es el que se presenta en la Figura 14 (pg. 130). Los elementos contenidos dentro del material compuesto (nivel n0) se relacionan como cadena de procesamiento de señales y por lo tanto deben estar agrupados dentro de un mismo contexto a un mismo nivel de jerarquía.

Otro ejemplo de **agrupamiento lógico** sutilmente distintos es el que se muestra en la Figura 15 (pg. 131). Los elementos a nivel n1 agrupados dentro del agrupamiento simultáneo (nivel n2), que simboliza una frase musical, están relacionados por el procesamiento dinámico de la envolvente de frase, vector n1, que afecta el conjunto de eventos n1. A su vez, los elementos denominados material compuesto a nivel n0 están

agrupados por necesidades de procesamiento dentro del conjunto n1 que recibe la envolvente n1 y propaga el parámetro la “amplitud” (implícito en el gráfico) de cada uno de los materiales compuestos a nivel n0.

En relación a los procesos de síntesis, los **agrupamientos lógicos** no expresan operaciones específicas, como la suma o la multiplicación de señales, sino la relación paramétrica. Por el momento, este **modelo** de representación no intenta expresar en detalle este tipo de relaciones específicas sino simplemente los distintos tipos de relaciones estructurales generales. Un mayor grado de especificidad con respecto a este tema puede ser desarrollado en trabajos posteriores en relación a una posible implementación como herramienta informática (ver más adelante).

Es posible que los agrupamientos lógicos coincidan con los agrupamientos compositivos pero no es una condición necesaria. La naturaleza de los agrupamientos resultantes puede producirse por factores combinados de ambos tipos de agrupamientos. Por otra parte, como se demostró anteriormente en este capítulo, tanto los agrupamientos compositivos como los agrupamientos lógicos pueden ser representados de distintas maneras y producir significados equivalentes o pueden representados de la misma manera y producir significados distintos.

Como se expresó en el **Capítulo 3**, la noción principal que se guió este análisis es la de **material musical**. En aquel momento se dio una definición operativa lo suficientemente amplia como para poder trabajar sobre los distintos aspectos que contribuyen a la elaboración de una composición musical. Es recién a estas alturas del libro que se puede dar como corolario una de **material musical** informático que deriva del análisis realizado y el modelo desarrollado.

Definición general de material musical: *Debido a que el objeto que se considere como unidad de síntesis o análisis es relativo al nivel de agrupamiento seleccionado (nivel base), los materiales musicales pueden ser definidos de manera general como entidades delimitadas y diferenciadas arbitrariamente que generan unidades de significado y que pueden ser descompuestas o combinadas para formar nuevas entidades a otro nivel de complejidad*.

Desarrollo de *Software*

En esta sección se expone de manera resumida algunos de los principios de la programación para el agrupamiento jerárquico de las estructuras de datos básicas en el entorno *SuperCollider*. Esto simplemente es una referencia a una posible implementación del sistema como *quark* (librería externa). No se especifican los detalles puesto que la implementación queda fuera del alcance de esta **Tesis**.

Estructura de datos

El principio de organización jerárquica está presente en casi la totalidad de los sistemas de representación y análisis musical, y en las técnicas de programación estructurada. Es lo que da cohesión a los elementos del sistema y posibilita distintos tipos de relaciones entre ellos. Las estructuras descritas anteriormente se emplean para representar parámetros sonoros y musicales y por lo tanto, pueden ser agrupadas y relacionadas jerárquicamente para crear entidades más complejas.

Para poder representar las estructuras musicales de manera versátil, mediante estructuras de datos propias de la computación, se recurre a listas jerárquicas. Por lo tanto se define a la estructura de datos *nodo* como el elemento contenedor (**contexto**) y organizador de los materiales básicos. Los *nodos* definen las propiedades temporales fundamentales de los **materiales concretos** y **lógicos** y por lo tanto sus **características** y tipos de r**elaciones temporales**. En cuanto a la programación, una posible jerarquía de clases lo suficientemente simple y esquemática como para organizar las estructuras de datos básicas como nodos se muestra en la Figura 23.

Los datos concretos de los materiales musicales están contenidos en los distintos tipos de nodos hoja y los nodos grupo tienen la función de organizar los materiales básicos proveyendo el contexto temporal y la lógica organizativa de los datos como parámetros de síntesis y cómo estructuras jerárquicas.

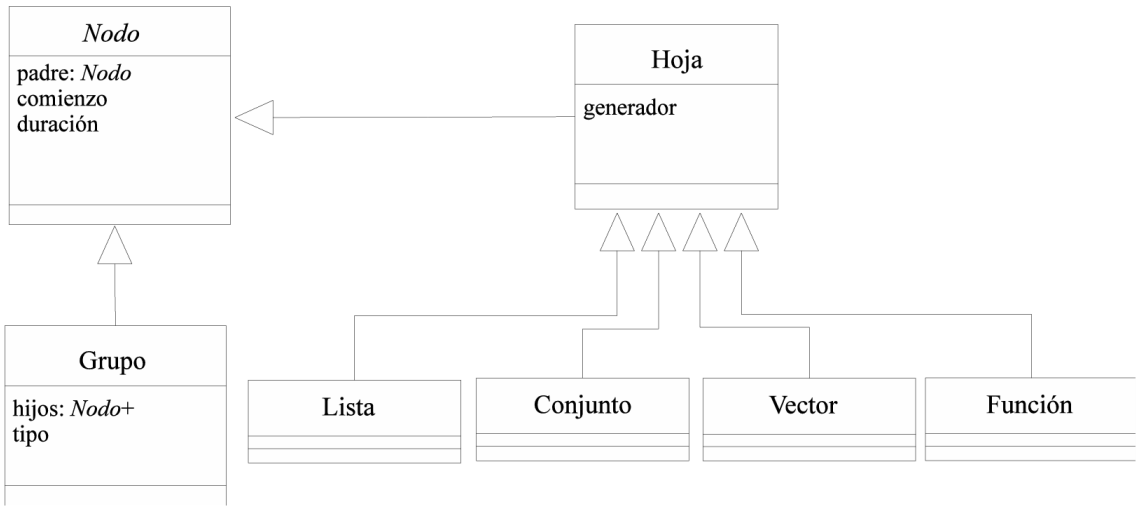


Figura 23 Diagrama de especialización de las estructuras de datos.

Relación jerárquica y comunicación

Como medio de comunicación y relación de los parámetros se emplean *buses* de la

misma manera que se emplean para realizar la síntesis de sonido en *SuperCollider*, pero con propiedades adicionales que derivan de los tipos de relación de los **materiales básicos** al ser agrupados en *nodos*. Las estructuras de más bajo nivel empleadas para generar sonido y señales de control como **abstracciones instrumentales** (*definiciones de instrumento*, *buses* globales, *nodos* de síntesis, *patterns*, etc), dependen de la arquitectura de *SuperCollider* y no son expuestas en este trabajo.

Los *nodos* pueden ser *emisores*, *receptores* o ambos según como se organicen las relaciones de parámetros de los **materiales concreto**s al ser agrupados. Debido a que la comunicación se produce mediante *buses*, se vuelven relevantes las propiedades de continuidad y discontinuidad, y los tipos de relaciones temporales de los materiales básicos como parámetros. Los nodos emisores generan señales de audio o control que son escritas en buses uno por cada parámetro. Los nodos receptores son los que reciben los parámetros de sus nodos hermanos y los heredados de un nodo superior, representan las unidades de síntesis (*definiciones de instrumento*) que son representadas como **funciones** (**material lógico**), pudiendo generar señales de audio o de control (**vectores**) como parámetros de salida para ser reutilizados dentro de la estructura jerárquica.

Por ejemplo, en la Figura 24 se puede apreciar la relación jerárquica entre *nodos emisores* y *receptores*. El grupo A contiene la lista A y los grupos B y C. La lista A (elemento emisor) representa la “amplitud” que es pasada como parámetro a los grupos B y C (nodos hermanos). Cada uno de estos grupos contienen una lista *emisora* que representa el parámetro “frecuencia” que es pasado junto con el parámetro “amplitud” a las funciones A y B que representan distintas unidades de síntesis (el algoritmo generador puede ser el mismo o no). Por último, la salida de las funciones A y B son tomadas como parámetros de salida de las grupos B y C, y se propagan como salidas del grupo A. En al Figura 24 los nodos se anidan gráficamente y se representa el tiempo como espacio horizontal para poder expresar la naturaleza temporal de la estructura. En el siguiente apartado se analizan los elementos gráficos que complementan estas estructuras de datos.

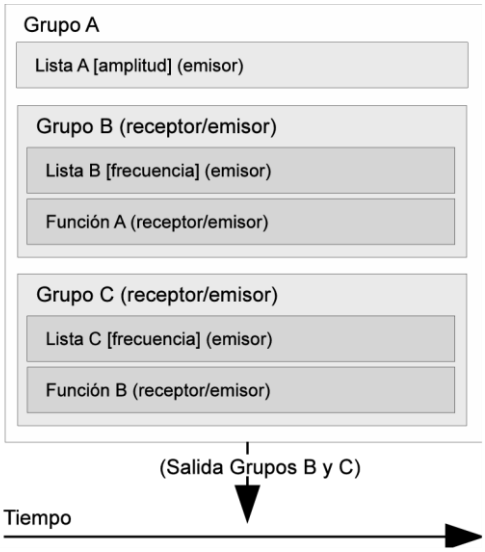


Figura 24 Agrupamiento de nodos emisores y receptores.

Recursos gráficos

Partiendo de las características y los tipos de relaciones temporales y la organización jerárquica de las estructuras de datos que se han expuesto, es fácil trazar un paralelo visual. Sin embargo, el problema es obtener la representación adecuada para cada etapa del proceso de composición sonora.

Es evidente que para cierto tipo de manipulación de datos, el código de programación es el recurso más sintético y práctico, pero en muchos casos, y sobre todo para la organización temporal, los recursos visuales resultan mucho más adecuados y facilitan la edición de las relaciones temporales. Por estas razones, la organización gráfica propuesta se encarga de representar y facilitar la edición de los **materiales generados** y deja la representación de los **materiales generadores** a los lenguajes de programación. Este enfoque permite una interacción bidireccional entre los **materiales lógicos** y los **materiales concretos** al estar ambos integrados dentro de una misma estructura de datos jerárquica.

Para correlacionar las estructuras de datos con los elementos gráficos en dos dimensiones, basta con adjudicar significado (hacer un *mapeo visual*) horizontal y vertical a los valores contenidos. A tal efecto se define como la ***unidad gráfica*** a un rectángulo de proporciones variables, su largo corresponde a la duración y su altura depende del tipo de dato y su relación con otros datos en el eje vertical. Si bien parece trivial, este concepto es importante porque todos los objetos gráficos complejos se componen por combinación de unidades gráficas, y porque a partir de estas se puede definir un conjunto acotado de propiedades visuales en relación a la representación de los

datos y estructuras involucradas.

Propiedades gráficas

Las **propiedades gráficas** detalladas a continuación son: ***relación vertical***, ***rango vertical***, ***resolución vertical*** y ***horizontal***, y ***mapeo vertical*** y ***horizontal***.

La **relación vertical** puede ser de dos tipos diferentes, ***arbitraria*** o ***escalar***, debido a que es el parámetro visual que generalmente se emplea tanto para la organización de diferentes tipos de datos en diferentes estratos, como para representar la relación entre datos del mismo tipo (Figura 25). El **rango vertical** es simplemente el conjunto de valores posibles si el tipo de **relación vertical** es escalar. La **resolución vertical** es el espacio en altura que ocupan los objetos y depende del tipo de relación y del **rango vertical**. La **resolución horizontal** es la medida visual equivalente al mínimo período temporal representable (el cual puede ser variable). El mapeo es la transposición de los valores representados por las estructuras de datos a los objetos visuales, pudiendo ser lineal o una distorsión útil para la representación gráfica.

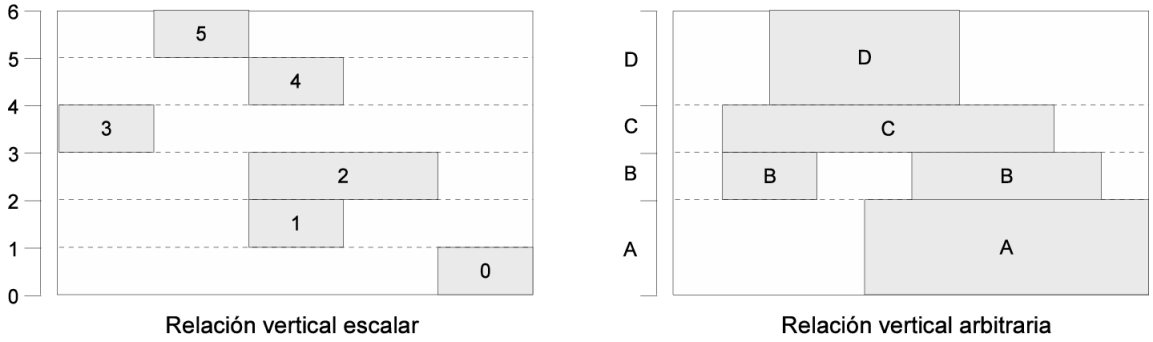


Figura 25 Tipos de relación vertical. En el objeto de relación escalar se puede apreciar el rango vertical (de 0 a 6) con una resolución vertical de 1 y un mapeo lineal entre los valores de los objetos y la escala representada visualmente mediante una regla.

La estructura de datos *nodo* se representa gráficamente como un rectángulo (*unidad gráfica*) cuya altura depende de la **resolución vertical**, y sus sub-nodos se representan de la misma manera hasta llegar al contenido de los *nodos terminales*. Si bien los *nodos terminales* (e.g. **listas** y **vectores**) que representan señales de audio o de control deben dibujar su contenido de manera eficiente (no sería práctico dibujar cada uno de sus elementos internos como nodos aislados), los recursos gráficos son teóricamente los mismo que los empleados para las estructuras (*nodos*) de más alto nivel.

La adición de las **propiedades gráficas** a los distintos tipos de estructuras de datos y sus relaciones jerárquicas, aporta coherencia estructural a la representación y posibilita

las representaciones visuales más comunes de los entornos de edición de sonido. Por ejemplo, las visualizaciones de *oscilograma*, *piano roll*, *espectrograma*, y también las combinaciones gráficas que generalmente dependen del software empleado, por ejemplo, *piano roll* y envolvente dinámica, *espectrograma* y forma de onda, entre muchas otras.

Al agrupar las estructuras de datos, la ***resultante visual*** puede resultar demasiado compleja si cada elemento representa en detalle sus componentes, por lo tanto, es posible determinar uno o varios parámetros como la resultante visual del material agrupado, pudiendo cambiar de representación gráfica según el nivel de detalle que se quiera trabajar en la edición. Por ejemplo, en la Figura 26 se pueden ver distintos materiales agrupados pero con diferentes **resultantes visuales**. En el grupo A, la **relación vertical** es arbitraria y se utiliza para organizar los materiales de los sub- grupos B, C, D y E por canales. El grupo B no representa gráficamente sus componentes internos solo emplea una etiqueta (“ruido”) para hacer referencia a su contenido. Los grupos C y D representan la envolvente dinámica como función simétrica y las alturas como *piano roll*, pero no representa gráficamente la función de síntesis o instrumento empleado. Y el grupo E representa los parámetros de altura y amplitud agrupados por **eventos**. Este último grupo es un caso especial porque agrupa la representación de dos parámetros en un solo objeto gráfico, la **escala vertical** es empleada para determinar la posición de los objetos en base a la frecuencia y la **resolución vertical** es empleada para dibujar la envolvente dinámica mediante una función simétrica. Esto recursos visuales son comunes en varios secuenciadores como *HighC*, *MetaSynth* (Thiebaut et al. 2008) y *LMMS*58 entre otros, pero el enfoque aportado aquí difiere en que no se imponen restricciones en los elementos gráficos como interfaz pre-programada sino que estos se construyen a la par de las estructuras de datos que representan.

Por último, la edición visual de los materiales concretos está sujeta a las **características** y **relaciones temporales** propias de las estructuras de datos. De la misma manera que se descartan los **procesos lógicos** de los **materiales generados**, también se hace con las propiedades visuales. Por ejemplo, la edición de **materiales concretos** sucesivos altera sus elementos constitutivos pero no cambia sus propiedades como sucesión. Sin embargo, aunque no será desarrollado en este escrito, cabe mencionar que es posible convertir un tipo de material en otro para obtener distintos recursos de manipulación, aunque tal operación puede cambiar, de manera irreversible, el tipo de material representado.

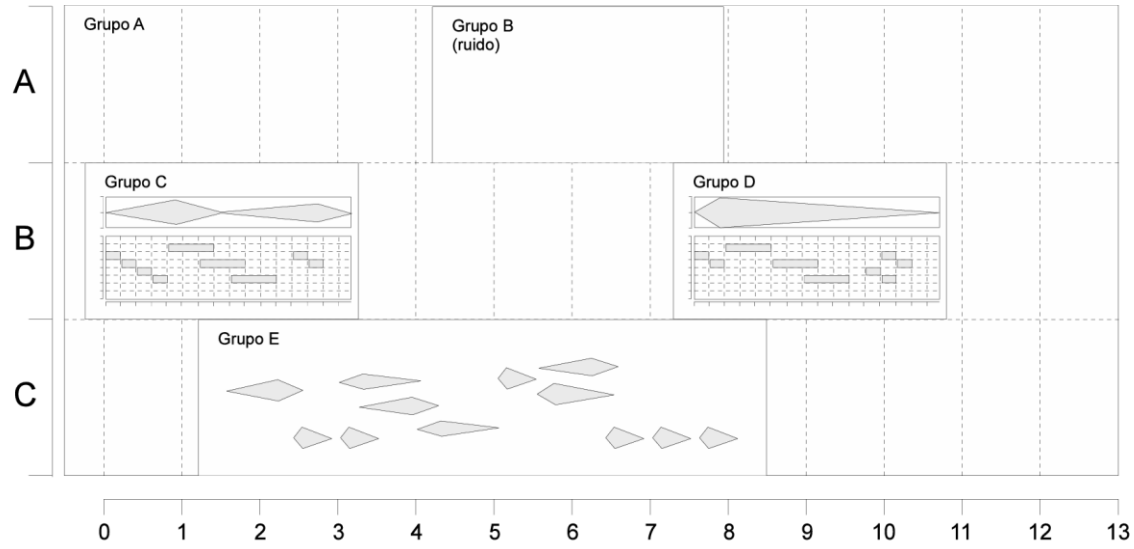


Figura 26 Resultante visual de varios agrupamientos.

CAPÍTULO 7: Ejemplos de Análisis

Introducción

La finalidad de este capítulo es la de probar el **modelo** de representación propuesto por esta **Tesis** como herramienta de análisis de obras electroacúsitcas. Se presentarán dos ejemplo, uno escrito en el entorno *SuperCollider* (McCartney 2002) y otro en el entorno *Csound* (Vercoe 1986).

Método

Prerrequisitos y delimitación

Para realizar el análisis de obras electroacústicas compuestas mediante entornos de programación aplicando el **modelo** desarrollado en esta **Tesis** es necesario que se cumplan ciertos prerrequisitos:

- Acceso al material lógico con el cual fue compuesta la obra.
- Entendimiento de los recursos informáticos empleados.

El material lógico empleado consiste principalmente en los archivos fuente desarrollados para la composición. Si la obra contuviera procesamientos realizados mediante otro tipo de *software* (tales como editores de audio, editores lineales u otros) es conveniente tener al menos las especificaciones técnicas de estos, e.g. qué tipo de procesamiento realizan, y cuales son los datos de entrada y salida que interactúan en el sistema de desarrollo de la pieza.

En los casos en que se emplean archivos de audio como **materiales concretos** predefinidos, es posible que estos estén disponibles (en el mejor de los casos) o no para analizar las transformaciones que se realizan con respecto al material original. Sin embargo, si se dispone de la lógica de programación, se pueden inferir ciertas características de los datos de entrada comparando el resultado en presencia o ausencia de los mismos con el **registro** de la obra. En estos casos, el análisis por separado de los materiales concretos no disponibles se puede de realizar mediante otros recursos (e.g. análisis espectral o algoritmos de análisis más específicos) aplicados sobre dicho **registro**.

El **registro** de la obra se refiere, en este contexto, a la grabación del resultado sonoro de la obra como producto final. Otros tipos de **registro**, como las anotaciones de un compositor o el contexto de creación de una obra se consideran relevantes, pero no

forman parte de los aspecto específicos que se pueden analizar con este **modelo**. Sin embargo, estos otros tipos de **registro** pueden contribuir a otras etapas del análisis de la obra en cuestión e incluso a la deducción de información ausente en los **materiales lógicos** disponibles. En este sentido, **el modelo de análisis propuesto es una técnica particular y no un método general**.

El entendimiento de los recursos informáticos empleados es un requisito indispensable que debe cumplir el analista, puesto que estos son considerados como el **medio de representación y producción de la obra musical**. Por ejemplo, en el caso de la música instrumental en notación tradicional, es obvio que el análisis de una pieza orquestal resultaría restrictivo si no se conociera el contexto de producción que involucra el conocimiento de las características particulares de los medios técnicos y representación. Sin embargo, en el caso de la música electroacústica, este hecho no parece estar tan claro. El análisis que se puede realizar sobre el **registro** de audio de una obra empleando otros recursos de análisis (e.g. análisis espectral o algoritmos de análisis más específicos), si bien es considera completamente válido, puede perder fácilmente de vista el grado de detalle que se presenta de manera explícita en los **materiales lógicos**. Es válido aclarar que la técnica de análisis estructural desarrollada en esta **Tesis tiene como premisa el acceso a los materiales lógicos como medio de representación, y se centra en los aspectos estructurales que se relacionan con la técnica compositiva**. El análisis conceptual, cognitivo o estético que se pueda derivar mediante estas herramientas pertenece a otras etapas o planos.

La gran variedad de entornos y herramientas de *software* específicas existentes puede dificultar notablemente la tarea de análisis y el entendimiento de los resultados para el público en general. Es finalidad de este **modelo** la de unificar la diversidad de concepciones técnicas específicas que aplican distintas herramientas informáticas, para tratar de definir un contexto de representación más general centrado en las ideas musicales. De esta manera, el conocimiento específico de los entornos será obligatorio para el analista pero no par quienes accedan a los resultados del análisis.

Factores analizables

Aplicar el **modelo** de representación al análisis musical implica distinguir aspectos técnicos de la composición con medios electroacústicos, que fueron definidos en el marco teórico desarrollado y por los elementos del sistema de representación. Estos se enumeran a continuación:

- Definición y análisis de los **recursos** técnico-tecnológicos.

- Definición de las **restricciones impuestas o compositivas**.
- Definición y análisis de las **relaciones abstractas**.
- Definición de la **interfaz instrumental** y la **interfaz compositiva**.
- Definición del **nivel base** y las **abstracciones compositivas**.
- Organización estructural **modelizada** de las abstracciones empleadas.

Estos factores no deben ser tenidos en cuenta como “pasos a seguir” de un método de análisis sino como variables del problema estudiado. En los siguientes ejemplos se muestra como estos factores se integran en el estudio de casos particulares.

Análisis

SuperCollider

Como demostración de análisis en el entorno *SuperCollider* se trabajará sobre el ejemplo Código 12 expuesto en el **Capítulo 5**. Este ejemplo fue elaborado *ad hoc* para el desarrollo de esta **Tesis** y resulta conveniente por razones de claridad en la exposición, puesto que los aspectos relacionados con las estructuras del lenguaje de programación ya fueron concretamente analizadas en su momento.

Este ejemplo, reproducido nuevamente en Código 16 para comodidad del lector, se ajusta al modo de empleo habitual del entorno en cuestión. Emplea únicamente las abstracciones de la librería estándar de *sclang* para demostrar, de manera clara, las propiedades de este sistema. No se emplean abstracciones de más alto nivel elaboradas por terceros, estas se construyen sobre los elementos disponibles en la distribución estándar.

```
(
// definiciones de instrumento (abstracción instrumental)
SynthDef(\playbuf, { arg out = 0, pan = 0, buf, amp = 1;
    var sig;
    sig = PlayBuf.ar(1, buf) * amp;
    sig = Pan2.ar(sig, In.kr(pan));
    Out.ar(out, sig);
}).add;

SynthDef(\mod, { arg out, freq = 5, amp = 1;
    var sig;
    sig = LFNoise2.kr(freq, amp);
    Out.kr(out, sig);
}).add;

SynthDef(\sine, { arg out = 0, freq = 440, amp = 0.2, pan = 0;
    var sig, env;
    sig = SinOsc.ar(freq.lag(0.2), 0, amp.lag(0.2));
    env = EnvGate.new;
    sig = Pan2.ar(sig * env, In.kr(pan));
    Out.ar(out, sig);
}).add;

// nodo grupo (abstracción instrumental del servidor)
~grupo1 = Group.new;
~grupo2 = Group.after(~grupo1);

// bus (abstracción instrumental del servidor)
~bus1 = Bus.control;
~bus2 = Bus.control;

// buffer (abstracción instrumental que contiene
// una abstracción material concreta)
~archivo = Buffer.read(s, "archivo.wav");
)

(
// ~funcSine organiza el código, no los materiales musicales
~funcSine = { arg synth;
    var ret = [];

    // rutina de control (material lógico)
    ret = ret ++ Routine {
        loop {
            synth.set(\freq, [60, 62, 67, 69, 72].choose.midicps);
            [0.5, 1].choose.wait;
        };
    }.play;

```

```
50      // rutina de control (material lógico)
51      ret = ret ++ Routine {
52          loop {
53              synth.set(\amp, 1.0.rand);
54              0.2.wait;
55          };
56      }.play;
57
58      ret;
59  };
60  )
61
62  (
63  Routine.new {
64      var playbuf, mod1, sine, mod2, rout;
65
66      s.bind {
67          mod1 = Synth(\mod, [out: ~bus1], ~grupo1);
68          playbuf = Synth(\playbuf,
69              [buf: ~archivo, pan: ~bus1], ~grupo2);
70      };
71
72      10.wait;
73
74      s.bind {
75          mod2 = Synth(\mod, [out: ~bus2], ~grupo1);
76          sine = Synth(\sine, [fadeTime: 10, pan: ~bus2], ~grupo2);
77          s.sync;
78          rout = ~funcSine.value(sine);
79      };
80
81      20.wait;
82
83      sine.set(\gate, 0);
84      8.wait;
85
86      rout.do(_.stop);
87      playbuf.free;
88      mod1.free;
89      mod2.free;
90
91      // necesario para la programación
92      ~bus1.free;
93      ~bus2.free;
94      ~archivo.free;
95  }.play
96  )
```

Código 16

Interfaz instrumental

Los elementos que representan la **interfaz instrumental** empleada están definidos entre las líneas 1 y 3559 y consisten principalmente en las definiciones instrumentales *playbuf*, *mod*, y *sine*.

La definición de instrumento *playbuf*, organiza la lógica de reproducción y *paneo* de un *buffer* de audio que, en este caso, está previsto que represente el **material concreto** contenido en “archivo.wav”. Sus parámetros de control son *out* (*bus* de salida de la señal de audio), *pan* (posición del *paneo* en el campo estéreo), *buf* (referencia al *buffer* a reproducir) y *amp* (amplitud de la señal de salida).

En la Figura 27, gráfico A, se expone la representación del algoritmo de síntesis como **abstracción instrumental** atemporal, donde se puede observar la organización estructural de sus componentes principales, a la manera de un *patch* en entornos como *Pure Data* (Puckette 1996). En el gráfico B de la Figura 27, se muestra la representación de la estructura **realizada** en función del tiempo como **abstracción compositiva**. Ambas representaciones (A y B) son equivalentes con respecto a la estructura algorítmica60. La estructura queda representada en función de sí misma como **lógica abstracta** (gráfico A), o en función del tiempo, como relación entre **objetos** temporales (gráfico B).

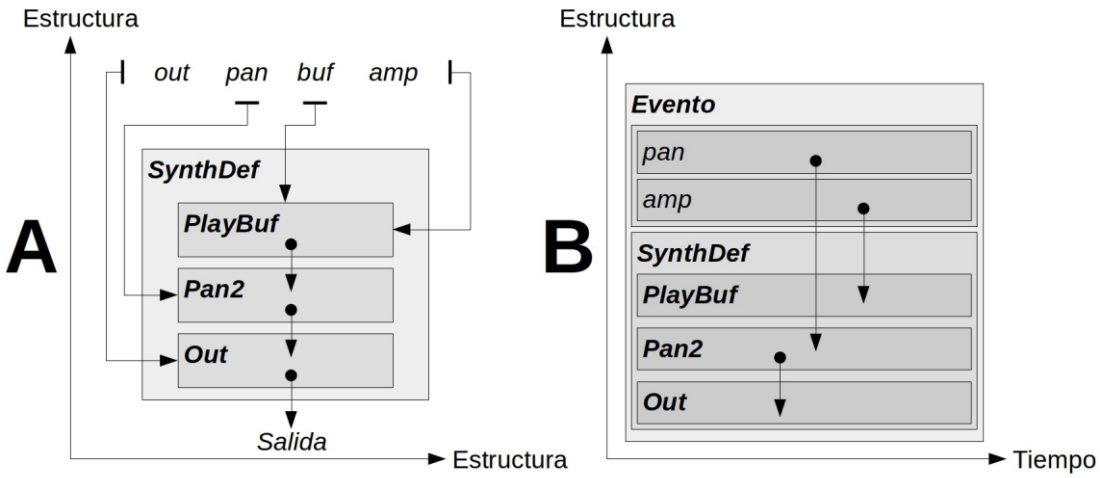


Figura 27 Representación del instrumento ***\playbuf*** *como* ***abstracción instrumental*** *atemporal (gráfico A) y como* ***abstracción material*** *realizada temporalmente (gráfico B).*

Al ser representado como **abstracción compositiva**, el instrumento *playbuf* puede ser entendido de varias maneras según el nivel de abstracción al que se quiera representar el **material musical**. En este caso se distinguen tres posibilidades, una como **material lógico** y dos como **material concreto**.

Como **material lógico**, su primitiva de representación es una **función** que se desarrolla en el tiempo (gráfico A de la Figura 28). Este tipo de representación facilita la visualización de las relaciones entre estructuras a mayor nivel de abstracción.

Una de sus representaciones como **material concreto** es la que se expuso en el gráfico B de la figura Figura 27. En este tipo de representación se destaca la descripción estructural de sus componentes concretados temporalmente, lo que facilita la visualización de las relaciones internas del material representado.

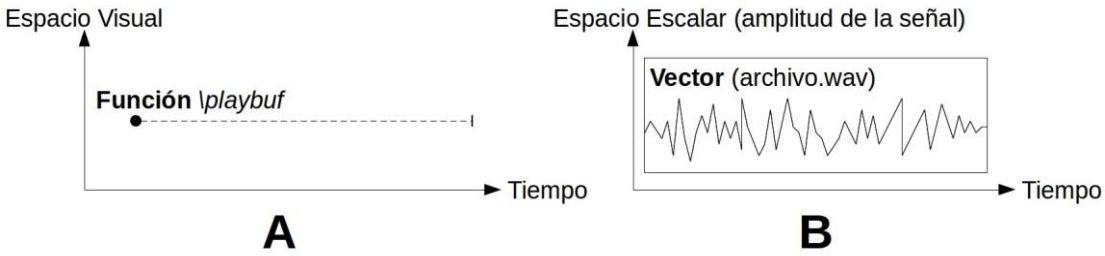


*Figura 28 Cambio de estado entre **material lógico** (A), representado como **función**, y **material concreto** (B), representado como **vector**.*

La segunda representación posible, como **material concreto**, es la que se muestra en el gráfico B de la Figura 28. La primitiva resultante de la **realización** de la **función** *playbuf* es un **vector** (señal de audio).

Según las relaciones estructurales que se quieran representar en relación a un **nivel base**, ya sea como **contexto** *contenedor* o como *contenido* dentro de un **contexto** más amplio, se puede optar alternativamente por una u otra.

En resumen, el elemento constituido por la **definición de instrumento** *playbuf*, puede ser considerado analíticamente de cuatro maneras:

1. Como **abstracción instrumental** (Figura 27, gráfico A).

2. Como **abstracción compositiva**, **material concreto** que preservar la relación estructural de la abstracción instrumental (Figura 27, gráfico B).

3. Como **abstracción compositiva**, **material lógico** simplificado por la primitiva **función** (Figura 28, gráfico A).

4. Como **abstracción compositiva**, **material concreto generado** que cambia su estado a **vector** (Figura 28, gráfico B).

La definición de instrumento *mod*, se emplea como abstracción de control. Será

considerado como **material lógico** de tipo **función** que genera una señal de ruido de baja frecuencia cuya amplitud oscila entre valores positivos y negativos. Sus parámetros de control son *out* (*bus* de salida de la señal de control), *freq* (frecuencia a la que se generan los valores aleatorios) y *amp* (amplitud del ruido generado).

Como **función**, *mod* puede representar de igual manera que *playbuf* en el gráfico A de la Figura 28. A diferencia de *playbuf* el material generado por *mod* es distinto en cada realización del algoritmo. En este caso, si que quisiera representar como **material concreto** se estaría optando por una realización en particular, lo cual es un factor que hay que tener en cuenta a la hora de analizar las intenciones compositivas.

La definición de instrumento *sine*, genera sinusoides continuas que pueden variar tanto en frecuencia (parámetro *freq*), amplitud (parámetro *amp*) y posición en el campo estéreo (parámetro *pan*). También es posible direccionar la salida de audio mediante el parámetro *out*. Estructuralmente está compuesta por una unidad generadora que produce sinusoides, una envolvente ASR (acrónimo en inglés de *Attack Sustain Release*) y una unidad de *paneo*. Su realización estructural puede ser expresada diversamente de la misma manera que *playbuf*.

Entre las líneas 25 y 30 se definen las **abstracciones instrumentales** del programa de síntesis (*scsynth*), *grupos* y *buses*, que se almacenan en las variables de entorno *~grupo1*, *~grupo2*, *~bus1* y *~bus2*. El *buffer*, en el que carga el archivo de audio “archivo.wav”, se guarda en la variable de entorno *~archivo* (línea 34). Estas abstracciones instrumentales no son directamente representadas por el **modelo** propuesto en esta **Tesis** sino que actúan como **principios relacionales** de los **materiales lógicos** y **concretos**. Por ejemplo, los *grupos* y los *buses* de audio o de control, sirven para relacionar paramétricamente distintos materiales como se verá más adelante.

Interfaz de composición

Los elementos que representan la **interfaz de composición** empleada están definidos entre las líneas 37 y 96. Estos consisten principalmente en las *rutinas* como método de secuenciación.

Como **material lógico**, una *rutina* puede ser entendida como **función**, si se omite representar la concreción de sus elementos internos, o como **conjunto**, si se toman en cuenta las propiedades temporales y las relaciones de los elementos contenidos.

Entre las líneas 63 y 95 se define la *rutina* (objeto *Routine*) principal que secuencia todos los demás elementos. Esta *rutina* será considerada como **conjunto**. Puesto que es el mayor nivel de abstracción empleado en este fragmento musical, la representación de sus elementos internos es imperativa.

Los primeros elementos secuenciados por esta *rutina* son los nodos de síntesis *mod* (línea 67), almacenado en la variable *mod1*, y *playbuf* (línea 68), almacenado en la variable *playbuf*. Al instanciar estos nodos de síntesis se crean dos procesos simultáneos, relacionados mediante el *bus* de control almacenado en la variable de entorno *~bus1*. El proceso *mod1* actúa como modulador del proceso *plalybuf* que reproduce el material concreto almacenado en el *buffer ~archivo*.

Luego de 10 unidades de espera, determinadas por la instrucción *10.wait* de la línea 72, se crean los nodos de síntesis *mod2* y *sine*. El nodo de síntesis *sine* genera una sinusoide continua. El nodo de síntesis *mod2* actúa de la misma manera que *mod1* pero en relación al nodo *sine*.

La siguiente instrucción de secuenciación importante es la evaluación de la función *~funcSine* (línea 78) que genera dos *rutinas* de control sobre los parámetros *freq* y *amp* del nodo de síntesis *sine*. Ambas “su-rutinas” son almacenadas en la variable *rout*.

La instrucción *20.wait* de la línea 81 define el tiempo de espera de las acciones siguientes. En la línea 83 se envía el valor 0 al parámetro *gate* del nodo de síntesis *sine*. Esto produce que la envolvente ASR contenida en la definición de instrumento *sine* inicie su etapa de *release* que dura 10 unidades temporales según se definió por el parámetro *fadeTime*61 en la línea 76 y coincide con el final del sonido producido por este fragmento. 8 unidades temporales después se detienen las rutinas de control de los parámetros *freq* y *amp* del nodo *sine*, y se liberan los nodo *playbuf*, *mod1* y *mod2* que dejan de actuar inmediatamente (lineas 86 a 89). Las instrucciones comprendidas entre las líneas 92 y 94 no tienen significación musical como se explicó en el **Capítulo 5**.

Representación modelizada

La descripción de este ejemplo evidencia que las *rutinas*, como **interfaz de composición**, actúan como secuenciadores de acciones instrumentales en similitud con la *notación de acciones* de la música instrumental (Karkoschka 1972). Este tipo de notación, si bien es conveniente para la composición algorítmica, no expone con la misma claridad que otras herramientas (e.g. editores lineales) las relaciones temporales y la estructura de los materiales representados.

En la Figura 29 se expone el análisis de las estructuras materiales comprendidas por este programa vistas desde la macro estructura. En este análisis, se optó por la representación en forma de **funciones** de estos materiales puesto que se simplifica la visualización de los distintos niveles estructurales.

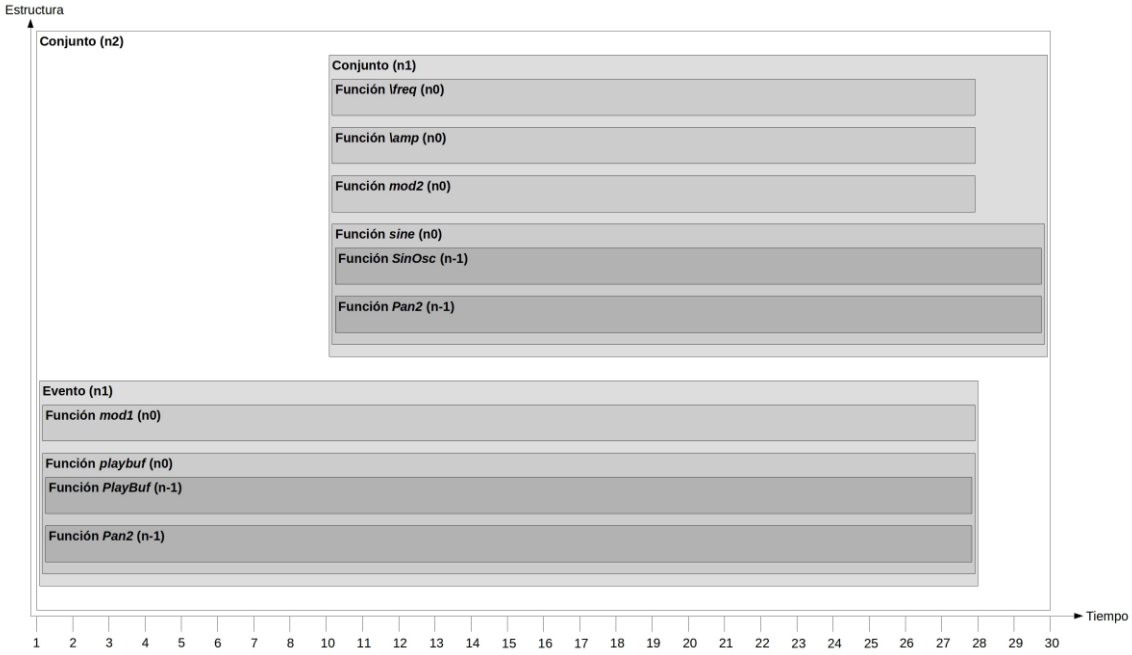


Figura 29 Estructura modelizada del ejemplo Código 16.

Nivel base

El **nivel base** (n0) se define en las estructuras de control y generación de sonido: los *nodos* de síntesis *sine* y *playbuf*, los nodos de síntesis *mod1* y *mod2*, que actúan como funciones de control, y las *rutinas* que controlan los parámetros *freq* y *amp* del nodo *sine*. En la Figura 29 se muestra, de manera simplificada, el nivel n-1 que se produce internamente en las definiciones de instrumento *sine* y *playbuf*, y los niveles de agrupamiento superior (n1), **evento n1** y **conjunto n1**, que expresan la relación paramétrica entre los elementos combinables a **nivel base**. El fragmento analizado, como estructura global, es un **conjunto** que queda a nivel n2.

Materiales compuestos

Aunque los agrupamientos resultantes se puedan considerar de distintas maneras, en este análisis se prioriza la relación paramétrica como principio de cohesión de las estructuras manipuladas a **nivel base**. Se consideran como materiales compuestos los elementos **conjunto n1** y **evento n1** (Figura 29).

En relación al resultado concreto, la percepción de los planos sonoros coincide con el nivel n1 que representa los estratos compuestos por el material concreto contenido en “archivo.wav” (**evento n1**) y la melodía sinusoidal que se genera en el **conjunto n1**. Con respecto a los diferentes tipos de análisis estructurales, se puede mencionar que el *paneo* aleatorio de ambos estratos actúa como elementos de cohesión perceptiva por similitud en el comportamiento a escala *meso temporal*, pero esto no coincide con la estructura lógica empleada para la

construcción del ejemplo.

Csound

Como ejemplo de la aplicación general de este **modelo de análisis**, en esta sección se realizará el análisis de un fragmento de la obra "*El aire y la reja*" (para clarinete bajo y electroacústica *surround*, 2014, inédita62) del compositor Oscar Pablo Di Liscia, cuya parte electroacústica fue compuesta con el entorno *Csound*.

Puesto que la lógica de *Csound* difiere sustancialmente de la de *SuperCollider*, este ejemplo es útil para demostrar la independencia que proponen los elementos representacionales del **modelo** elaborado en esta **Tesis** con respecto a las implementaciones particulares que distintos entornos realizan de las **abstracciones materiales**, y el grado de flexibilidad que la herramienta pueda ofrecer para la elaboración de las **concepciones compositivas**.

Debido a la extensión de los archivos fuente empleados para la elaboración de la sección analizada, a los fines de poder visualizar los aspectos tratados se expondrán solo los fragmentos de código correspondientes, con referencias al número de línea coincidente con los archivos originales disponibles los **Apéndices 1** y **2**.

Delimitación práctica del análisis

La obra dura aproximadamente 10 minutos y consta de varias secciones que emplean distintos recursos instrumentales y electroacústicos. La sección analizada para este trabajo es la que va, en relación a la partitura, desde el compás 116 hasta el final de la obra (compás 155). Los **materiales musicales** en esta sección se componen en tres planos, dos realizados electroacústicamente en *Csound* y la parte del instrumentos solista. Existen otros elementos que hacen a la estructura lógica de la programación en *Csound*, por ejemplo, los recursos de espacialización en Ambisonics (Malham 2009). Por razones de extensión, de todos los elementos que componen la sección, se analizará solo uno de los planos electroacústicos en relación a la parte de clarinete.

Parte de clarinete: notación convencional y notación extendida

Puesto que no es finalidad de este análisis estudiar la obra en su totalidad, sino los aspectos estructurales de la parte electroacústica, los recursos instrumentales convencionales y los materiales por estos producidos no serán analizados sino mencionados en cuanto a sus características generales.

Hecha esta salvedad, la parte de clarinete solista está representada en notación convencional, en compás de 4/4 a MM 60 (definido previamente como cambio de *tempo* en el

compás 81 a *poco meno mosso*). Los recursos instrumentales empleados en la sección analizada son: notas tenidas de variación dinámica progresiva en forma de *crescendo*/*decrescendo*, el *sforzato* como variación dinámica-tímbrica de los ataques, variaciones entre *frullato* y toque normal que se relacionan con *trinos* (mesurados y no mesurados), *acciaccaturas* y *floriturас* breves.

La parte de clarinete se relaciona con el material electroacústico no analizado en este trabajo, que consiste en notas tenidas de clarinete pre-grabadas y procesadas. Estas son representadas, en la partitura, mediante notas largas (redondas y blancas) en el monograma superior (ver **Apéndice 3**). La parte electroacústica analizada en este trabajo se representa como espectrograma, del canal *W* del sistema *Ambisonics*63, entre el monograma superior y el pentagrama del clarinete (a partir del compás 117).

La representación adoptada para la partitura es híbrida, puesto que representa la relación entre la notación instrumental convencional y dos tipos de representación diferentes para los estratos electroacústicos. Cada uno de los tres tipos de elementos visuales (pentagrama, espectrograma y monograma) representan las características relevantes de la composición instrumental y su relación con los estratos electroacústicos.

Parte electroacústica: Interfaz instrumental

El material electroacústico del estrato analizado consiste principalmente en un instrumento de síntesis granular. Las **abstracciones instrumentales** empleadas se exponen en los ejemplos Código 17 y Código 18.

Entre las líneas 353 y 467 se define el instrumento *grainer* (Código 17).

```
353 instr grainer
354
355   iaux=0
356   ;AMPLITUDE
357   $get_randval(p4'p5'iaux'iamp)
358   iatab   =p6
359   ;GAP
360   if(p7 < 0) then
361         $get_tabval(p7'p8'igg'itblg)
362   else
363         $get_randval(p7'p8'iaux'igg)
364   endif
365   ;DUR
366   $get_randval(p9'p10'iaux'igd)
```

```
367     ;START
368     if(p11 < 0) then
369           $get_tabval(p11'p12'ist'itblst)
370     else
371           $get_randval(p11'p12'iaux'ist)
372     endif
373     ;FDEV
374     if(p13 < 0) then
375           $get_tabval_d(p13'p14'idev'itbldev)
376     else
377           $get_randval(p13'p14'iaux'idev)
378     endif
379     ;AZIMUTH
380     $get_birandval(p15'p16'iaux'iaz)
381     ;ELEVATION
382     $get_birandval(p17'p18'iaux'iel)
383     ;DISTANCE
384     $get_randval(p19'p20'iaux'id)
385
386 ;AUDIO FILE
387 ifile     =p21    ;audio sound file
388
389 ;EVOLUTION TABLES
390 itgap     =p22    ;gap evolution table
391 itdur     =p23    ;duration scaling table
392 itdev     =p24    ;frequency deviation table
393 itaz      =p25    ;azimuth table
394 itds      =p26    ;distance table
395 itamp     =p27    ;amplitude table
396 itfp      =p28    ;starting file read point
397 ;FILTER PARAMETERS
398 ifc       =p29
399 ibw       =p30
400 itbw      =p31
401
402 ip2       =p2     ;p2 is needed for indexing tables but scaled by tempo
403 ip3       =p3     ;p3 is needed for indexing tables but scaled by tempo
404
405 ;gap table must be read
406 if(itgap !=0) then
407     ipoint        = (itgap < 0? .999 : 0)
408     if(frac(itgap) > 0) then
409           ipoint *= (frac(itgap)*10)
410           until ipoint < 1 do
411                 ipoint -=1
412           od
413     endif
414     ifac    tablei ipoint, abs(itgap), 1, 0, 1
415     igg     = igg * ifac
416     itacum =0
417 endif
```

```
418 reset:   timout  0, igg, contin
419          ifile  =p21
420          ;AMP
421          $get_randval(p4'p5'iaux'iamp)
422          ;GAP
423          if(p7 < 0) then
424                $get_tabval(p7'p8'igg'itblg)
425          else
426                $get_randval(p7'p8'iaux'igg)
427          endif
428          ;DUR
429          $get_randval(p9'p10'iaux'igd)
430          ;START
431          if(p11 < 0) then
432                $get_tabval(p11'p12'ist'itblst)
433          else
434                $get_randval(p11'p12'iaux'ist)
435          endif
436          ;FDEV
437          if(p13 < 0) then
438                $get_tabval_d(p13'p14'idev'itbldev)
439          else
440                $get_randval(p13'p14'iaux'idev)
```

```
441          endif
442          ;AZIMUTH
443          $get_birandval(p15'p16'iaux'iaz)
444          ;ELEVATION
445          $get_birandval(p17'p18'iaux'iel)
446          ;DISTANCE
447          $get_randval(p19'p20'iaux'id)
448          ;Gap evolution
449          if(itgap !=0) then
450                itacum = itacum + igg*ifac
451                ipoint  = (itgap > 0 ? itacum / ip3 : 1-itacum / ip3)
452                if(frac(itgap) > 0) then
453                      ipoint *= (frac(itgap)*10)
454                      until ipoint < 1 do
455                            ipoint -=1
456                      od
457                endif
458                ifac    tablei ipoint, abs(itgap), 1, 0, 1
459                igg    =igg * ifac
460          endif
461    reinit    reset
462
463  contin:  schedule "my_grain", 0, igd, iamp, iatab, ist, ifile, idev,
     iaz, id, iel, igd, itdur, itdev, itaz, itds,itamp, itfp, ip2, ip3, ifc,
     ibw, itbw
464    rireturn
465
466
467  endin
```

Código 17

En la línea 463 de Código 17, el instrumento *grainer*, mediante la instrucción *schedule*, llama al instrumento *my_grain* definido entre las líneas 469 y 575 (Código 18) y le pasa los parámetros calculados en el instrumento actual.

```
469 instr my_grain
470 iamp     =p4
471 iatab    =p5
472 iskip    =p6
473 $select_afile(p7)
474 idev     =p8
475 iaz      =p9
476 ids      =p10
477 iel      =p11
478 igdur    =p12
479
480 itdur    =p13
481 itdev    =p14
482 itaz     =p15
483 itds     =p16
484 itamp    =p17
485 itfp     =p18
486
487 ip2      =p19
488 ip3      =p20
489
490 ifc      =p21
491 ibw      =p22
492 itbw     =p23
493 ibwsc    =1
494 ilen     filelen Sfilename
495
496 ;Get table deviations
497 if(itdur !=0) then
498   $get_tab_index(itdur'indx'ip2'ip3)
499   idf    tablei indx, abs(itdur), 1
500   p3     =igdur*idf
501 endif
502 if(itdev !=0) then
503   $get_tab_index(itdev'indx'ip2'ip3)
504   iddev  tablei indx, abs(itdev), 1
505   idev   =idev*iddev
506 endif
507 if(itaz !=0) then
508   $get_tab_index(itaz'indx'ip2'ip3)
509   iazof  tablei indx, abs(itaz), 1
510   iaz    =iaz+iazof
511 endif
```

```
512 if(itds !=0) then
513   $get_tab_index(itds'indx'ip2'ip3)
514   idsof  tablei indx, abs(itds), 1
515   ids    =ids+idsof
516 endif
517 if(itamp !=0) then
518   $get_tab_index(itamp'indx'ip2'ip3)
519   iasc   tablei indx, abs(itamp), 1
520   iamp   =iamp*iasc
521 endif
522 if(itfp !=0) then
523   $get_tab_index(itfp'indx'ip2'ip3)
524   ifp    tablei indx, abs(itfp), 1, 0, 0
525   iskip  =(iskip+ifp*ilen)
526 endif
527
528 if(ifc == 0 ) igoto bwset                          ;no filtering
529   if(ibw < 0) goto rbw
530         if(itbw !=0) then
531               $get_tab_index(itbw'indx'ip2'ip3) ;bw scaler is scanned
    from itbw
532               ibwsc  tablei indx, abs(itbw), 1
533               ibw    =ibw*ibwsc*ifc
534         else
535               ibw    *=ifc
536         endif
537   goto bwset
538   rbw:                                    ;random bw scaler
539         ibwsc   unirand abs(ibwsc) ;scaler is a random value between
    itbw and itbw+abs(ibwsc)
540         ibw    =(itbw+ibwsc)*ifc
541 bwset:
542 ;amp gate envelope parameters
543 iseg     = 0.05*p3
544 isug     = p3-iseg
545   ifper  =1/p3
546 asig     diskin2      Sfilename, idev, iskip,0,0,8,32768     ;read
    audio file
547
548 if(ifc != 0) goto flta
549 abala = asig
550 goto noflta
551
552 flta:
553 ;print ifc
554 ares     resonx  asig, ifc, ibw, 4, 1
555 abala    balance ares, asig
556
557 noflta:
558 aenv     oscili 1, ifper, iatab
559 ahp      dcblock abala*aenv
```

```
560
561 ;spatialization
562 p3 = p3+gimaxdel
563 $POL2REC(iaz'ids'iel)
564 ift             =99          ;room parameters table
565 imode           =3           ;B format output
566 iucirc          =1           ;distance of the unit circle
567 agate           linseg iamp,isug,iamp,iseg,0
568 ain             =ahp*agate
569 ;BF ENCODING
570 aW,aX,aY,aZ     spat3di ain,ix,iy,iz,iucirc,ift,imode,0
571 ;UHJ DECODING
572 $UHJ
573 outs            aleft, aright
574 ;outq           aW*giamp,aX*giamp,aY*giamp,aZ*giamp
575 endin
```

Código 18

Entre las líneas 561 y 574 del ejemplo Código 18 se definen la salida del instrumento y el *engine* de espacializacion que no será tratado en este análisis.

El instrumento *grainer* es un secuenciador que calcula información necesaria para producir los granos y ajusta los parámetros que se comportan de manera aleatoria 64. Para la secuenciación, implementa un ciclo iterativo entre las líneas 418 y 463 (Código 17). Calcula los *gaps* (espacio temporal entre granos) y en cuanto se cumple la condición *timeout*, que representa la duración del *gap* del grano que se calculó previamente, vuelve a calcular los parámetros y generar el siguiente grano.

El instrumento *my_grain* obtiene los parámetros para generar los granos del instrumento *grainer*. Puede aplicar desviaciones y tablas (envolventes de control que actúan como **vectores**) a los parámetros necesarios para generar el grano (líneas 496 a 550, Código 18): *duración* de grano, desplazamiento en frecuencia, posición espacial en *azimut* y *distancia*, *amplitud*, *posición* de lectura en el archivo fuente y ancho de banda de un *filtro* resonante. Las envolventes de control pasan del instrumento *grainer* al instrumento *my_grain* y en este se aplican los cálculos. La única tabla de control que se utiliza en el instrumento *grainer* es la tabla de *gaps* puesto que afecta la secuenciación de los granos.

En resumen, se definen dos instrumentos anidados, *grainer* y *my_grain*. El primero actúa como secuenciador de los granos generados por el segundo, y ambos actúan como componentes a nivel de **abstracción instrumental**. Es un ejemplo complicado puesto que realizar este tipo de abstracciones en *Csound* es un procedimiento bastante complejo debido a las limitaciones del lenguaje. La generación de granos consiste simplemente en su disposición temporal (realizada en *grainer*) y la lectura de un fragmento de audio del

archivo fuente al cual se le aplica una envolvente dinámica y procesos de filtrado y espacialización.

En la parte correspondiente a la partitura de *Csound*, se definen las tablas que actúan como envolventes de control (líneas 44 a 128 del archivo “Part y Orq.scd”, ver en el **Apéndice 1**). Estas son **vectores** que se emplean para manipular dinámicamente los parámetros de síntesis.

Tanto en la partitura como en la orquesta y las macros necesarias para la realización de la parte electroacústica, se programan varias abstracción de alto nivel que resultan necesarias para organizar los parámetros (e.g. el cálculo de la frecuencia de los semitonos líneas 160 y 160 del archivo “Part y Orq.scd”, ver en el **Apéndice 1**). Estas **no son abstracciones compositivas sino informáticas**, necesarias para la implementación del programa, las cuales hacen notablemente compleja la tarea de análisis de las estructuras musicales. Como lenguaje de programación, *Csound*, en comparación con *SuperCollider*, carece de abstracciones básicas de más alto nivel comúnmente empleadas para la composición con medios electroacústicos, y las estructuras del lenguaje son más restringidas que el modelo de la programación orientada a objetos adoptado por *SuperCollider*.

Parte electroacústica: Interfaz de composición

Como referencia de la partitura de *Csound* empleada en el fragmento analizado, se tomarán las líneas 228 a 267 expuestas en el ejemplo Código 19. El resto de las instrucciones de la partitura no difiere en estructura y emplea sistemáticamente los mismos elementos y relaciones.

```
228 ;CONVERGENTE ------------------------------------------------------------------------
229 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
230 i "grainer" 3.9     5       .5      .1      $p_cnv  .66     0.0     .05     0.0     0.2
    0.0     $se(-4) 0.0
231 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
232             90      0       0       0       1       0       $CLASA  0       0       0
    52.0125 64      0       0       $NVFLT(11000'.175)
233
234 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
235 i "grainer" .       .       .       .       $p_cnv  .       0.0     .05     0.0     0.2
    0.0     $se(-4) 0.0
236 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
237             90      0       0       0       2       0       $CLASA  0       0       0
    51.0125 64      0       0       $NVFLT(5500'.175)
238 ;DIVERGENTE 1-----------------------------------------------------------------------
239 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
240 i "grainer" 9.05    4.64    .       .       $io4    .45     0.01    .05     0.025   0.2
    0.1     $se(-3) 0.025
241 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
```

```
242             45      10      0       0       1       0       $CLASA  0       0       0
    0       63      0       0       $NVFLT(11000'.2)
243
244 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
245 i "grainer" .       .       .       .       $io3    .45     0.01    .05     0.025   0.2
    0.1     $se(-2) 0.025
246 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
247             135     10      0       0       1       0       $CLASA  0       0       0
    0       63      0       0       $NVFLT(5500'.2)
248 ;DIVERGENTE 2-----------------------------------------------------------------------
249 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
250 i "grainer" 14.25   6.3     .       .       $io1    .5      0.05    .05     0.05    0.2
    0.1     $se(-4) 0.025
251 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
252             45      30      0       0       1       0       $CLASA  0       0
    45.05   0       61      0       0       $NVFLT(11000'.2)
253
254 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
255 i "grainer" .       .       .       .       $io2    .5      0.05    .05     0.05    0.2
    0.1     $se(-1) 0.025
256 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
257             135     30      0       0       1       0       $CLASA  0       0       48
    0       61      0       0       $NVFLT(5500'.2)
258 ;DIVERGENTE 1 hacia convergente-----------------------------------------------------
259 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
```

```
260 i "grainer" 20.6    5       .       .       $io3    .5      0.1     .05     0.05    0.2
    0.1     $se(-5) 0.025
261 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
262             45      10      0       0       1       0       $CLASA  0       0       0
    0       61      0       0       $NVFLT(12000'.15)
263
264 ;           st      dur     gamp    gard    gatab   gg      ggr     gd      gdr     gst
    gstr    gfdev   gfdr
265 i "grainer" .       .       .       .       $io2    .5      0.1     .05     0.05    0.2
    0.1     $se(0)  0.025
266 ;           gaz     gazr    gel     gelr    gdist   gdistr  gfile   tgap    tdur    tdev
    taz     tdis    tamp    tfp     fc      bw      bwt
267             135     10      0       0       1       0       $CLASA  0       0       0
    0       61      0       0       $NVFLT(4500'.15)
```

Código 19

En la línea 230 (Código 19) se produce el primer evento representado gráficamente por el espectrograma de la *particella* de clarinete. El planteo compositivo de esta parte electroacústica consisten en la superposición de dos o cuatro estrados (en cuanto a su posición espectral) de síntesis granular. Los estratos convergen y divergen en cuanto a su sincronía y su posición en un espacio acústico *perifónico* (3D) lo que genera una contradicción entre varias imágenes fuente y una sola fuente conceptual (Kendall 2009).

En cuanto a la representación de **abstracciones compositivas**, los eventos en la partitura de *Csound* actúan como secuenciadores del instrumento *grainer*, especificado su tiempo de inicio y duración. En esta sección analizada, los eventos se suceden en forma de **lista** puesto que cada estrato está seccionados en distintos eventos para facilitar el empleo modular del repertorio envolventes de control. Sin embargo, la abstracción que se deduce de la lógica propuesta por *Csound* sería el **conjunto**. La partitura de *Csound*, al ser la abstracción de más alto nivel que secuencia todas las demás, es definida como el **contexto** global que se emplea para la composición de la parte electroacústica analizada.

Representación modelizada

La estructura representada en la Figura 30 coincide con el fragmento de la partitura de *Csound* expuesto en el ejemplo Código 19. En este gráfico se puede apreciar la relación estructural de los estratos granulares en relación a la parte de clarinete y el estrato electroacústico de notas tenidas representado por el monograma superior.

Nivel Base

El nivel n0 se sitúa en las abstracciones que representan las envolventes de control y el instrumento *grainer*. Los elementos referidos como “Constantes (eventos)” a nivel n0 son valores escalares que actúan como parámetros del instrumento *grainer* de la misma manera que las envolventes de control, solo que, al ser constantes en el tiempo, se pueden representar como **eventos** que expresan el inicio y la duración de un parámetro.

Los puntos suspensivos debajo de los elementos “Envolventes (vectores)” y “Constantes (eventos)” en la Figura 30 indican que existen más de uno superpuestos, omitidos en el gráfico por razones de simplicidad.

El nivel n0 fue definido en base a los elementos mencionados por ser los que actúan como **elementos combinatorios de la composición a escala *meso temporal***. Cada uno de estos elementos (salvo las constantes) fueron pre-elaborados por el compositor para su posterior realización como elementos combinatorios de una textura evolutiva.

El nivel n-1 consiste en los granos generados por la definición de instrumento *my_grain*. Estos están contenidos dentro de la estructura de tipo **conjunto** generada por el instrumento *grainer* que actúa como secuenciador de granos. Los granos (n-1) son considerados como **funciones** puesto que no se intenta representar el material concreto generado, y están dispuesto de manera meramente ilustrativa dentro del **conjunto** para simplificar el gráfico.

En este análisis, el **nivel base** está definido en relación a la parte electroacústica analizada. Si se consideraran los tres estratos que componen la sección habría que verificar si el **nivel base** se mantiene constante o no. Aunque no se procederá con tal verificación de manera sistemática, a simple vista se puede apreciar que el comportamiento melódico del clarinete solista está compuesto por elementos a escala temporal de *objeto sonoro*, lo cual implica una divergencia con respecto a la elaboración de la textura analizada en cuanto a los tipos de elementos combinatorios manipulados compositivamente.

Materiales compuestos

A medida que se asciende en la estructura musical, la parte electroacústica analizada se compone de **eventos** (nivel n1) que seccionan el desarrollo de cada estrato de síntesis granular. El siguiente nivel estructural son las **listas** (nivel n2) que representan cada uno de esos estratos agrupando las sucesiones de eventos.

El mayor nivel estructural es un **conjunto** (nivel n3) que abarca las listas a nivel n2 y representa la textura resultante. En relación a la notación del compositor, el nivel estructural n3 es el que se representa mediante el espectrograma. Dentro de este se pueden visualizar las listas de eventos (n2) y los eventos por sus cualidades espectrales particulares. Esto sucede hasta aproximadamente el compás 132 de la *particella* de clarinete. Sin embargo, a partir del compás 133 los estratos granulares gradualmente se difuminan en la representación espectral

pese a que su lógica estructural permanece constante. A medida que la sección analizada avanza hacia el final de la pieza se adicionan otros dos estratos paralelos a nivel n2, que aumentan la densidad cronométrica, espectral y espacial de la parte electroacústica.

El nivel estructural n3 coincide con el lenguaje de partitura de *Csound*, esto no es casual puesto que el lenguaje de partitura en este entorno es la herramienta de mayor nivel estructural provista para la secuenciación de acciones. Este ejemplo demuestra como varía el **nivel base** de las abstracciones compositivas en relación a las interfaces instrumental y de composición, y la elaboración de distintos componentes dentro de una misma composición musical.

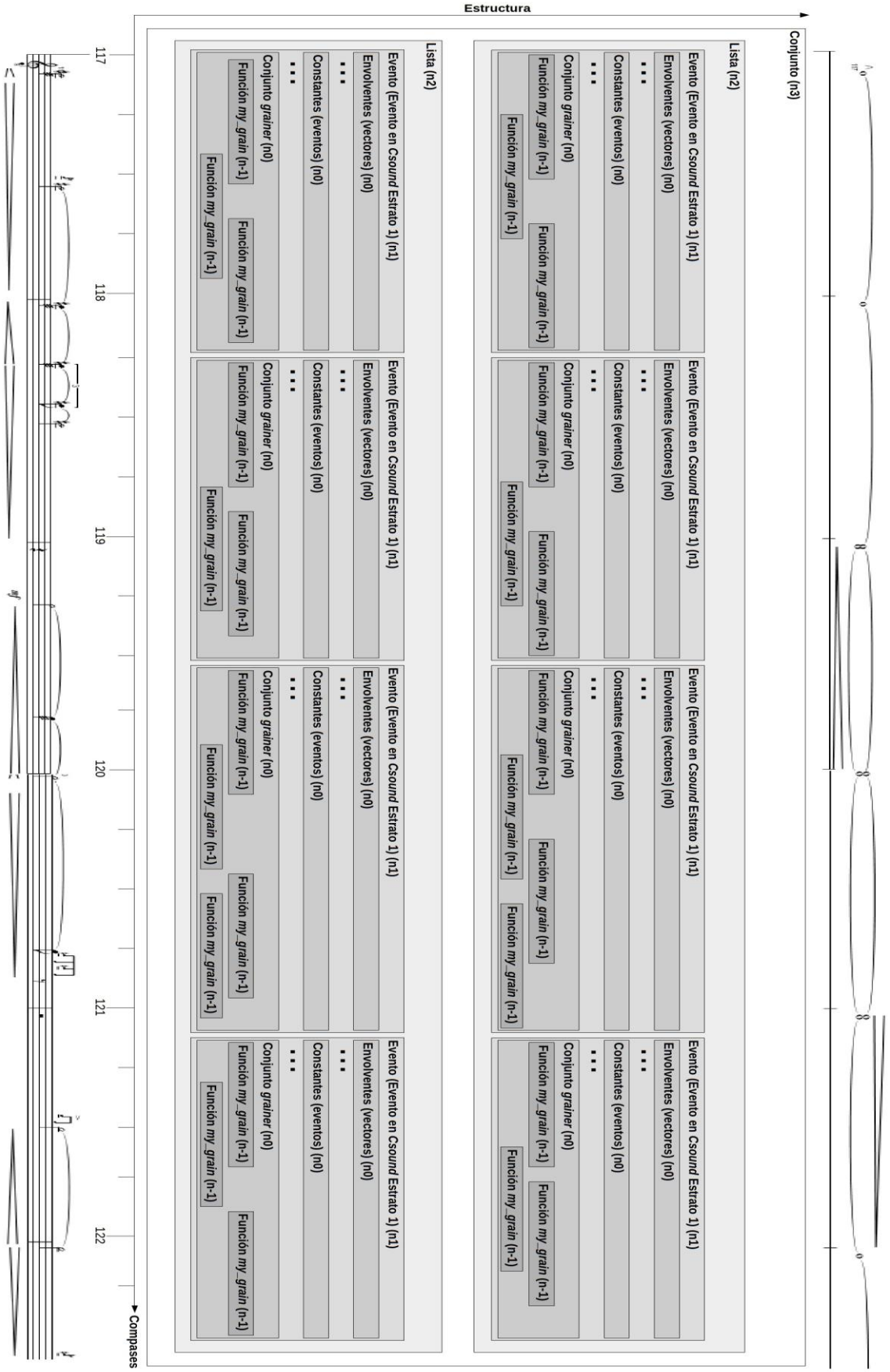


Figura 30 Representación modelizada de la sección electroacústica analizada (parte

Conclusiones Generales

De los objetivos de esta Tesis, se cumplió con la exploración del estado del arte de manera tal que posibilitó la realización de un marco teórico adecuado para el **modelo de representación** desarrollado. Los recursos empleados para la producción de música electroacústica fueron sistematizados, tanto teóricamente como prácticamente. Esta sistematización de los recursos no solo es aplicable a la producción sino también al estudio y la didáctica de la composición con medios electroacústicos. El desarrollo teórico expuesto en esta **Tesis** nombra y relaciona de manra sistemática muchos de los recursos conocidos, lo cual puede ser aplicado como sistema teórico sobre la materia de estudio.

Las ventajas del **modelo de representación**, en cuanto a la representación de los datos y la lógica de la música compuesta con lenguajes de programación fueron desarrolladas en el **Capítulo 7**, haciendo notar como se puede exponer la lógica estructural de los programas mediante una interfaz que refleja las nociones musicales temporalmente. El autor de esta **Tesis** cree que estudiar la sistematización propuestas por este **modelo** puede facilitar el desarrollo de distintas técnicas compositivas empleando diferentes entornos, lo cual puede ser desarrollado a futuro. En cuanto a la eficiencia del **modelo**, queda demostrado que, en comparación con la codificación en forma de programa, este simplifica notablemente la exposición organizada de los recursos empleados por la lógica específica de cada entorno. Sin embargo, el **modelo** es considerado como una herramienta que puede actuar sobre el empleo de cualquier entorno particular.

El diseño del modelo de representación parece ser lo suficientemente versátil como para abarcar distintos tipos de concepciones compositivas, simplificando la complejidad técnica y posibilitando que los esfuerzos (técnicos) se centren en la producción de abstracciones materiales musicales. La delimitación de estructuras materiales básicas derivadas de la programación y sus principios relacionales como formadores de estructura, son un elemento importante para el análisis de la técnica compositiva en sí misma. Sin embargo, como herramienta de trabajo para la composición, la falta de redundancia en las estructuras materiales básicas puede resultar un inconveniente. En algunos casos, en los que se requiera una organización más ambigua desde el punto de vista de la elaboración técnica instrumental, la falta de redundancia puede implicar que sea necesario reestructurar los materiales para relacionar distintos niveles de abstracción. Como herramienta de producción a desarrollar sería necesario estudiar los factores de redundancia que se podría agregar para simplificar la manipulación entre distintos niveles de abstracción.

De lo anterior se desprende que dos posibles aplicaciones a futuro pueden ser: el desarrollo de software que aplique este **modelo** como herramienta modular de producción de

alto nivel, actuando sobre entornos de programación existentes, y la sistematización del **modelo** como herramienta de análisis del repertorio de la música electroacústica más reciente.

El recorrido realizado en la elaboración de esta **Tesis** no solo resultó ser arduo, debido a los nuevos caminos que se abrían constantemente por la naturaleza dinámica de la materia de estudio, sino también apasionante, y su autor espera que el lector lo haya podido disfrutar de la misma manera. Esas nuevas posibilidades siguen abiertas y constituyen un desafío a enfrentar en desarrollos futuros, para los que este trabajo pretende haber aportado una posibilidad de acceso original.

Notas

1 No debe confundirse con las "*abstracciones de comportamientos*", concepto desarrollado por Dannenberg (1997) que será tratado luego en este libro.

2 Esta distinción no pretende generar juicio de valor alguno ni considera un técnica por sobre la otra. Ambas propician diferentes enfoques que pueden ofrecer mayor o menor grado de adecuación y flexibilidad según las intenciones compositivas.

3 Estos conceptos serán ampliamente desarrollados con posterioridad, sirva su mención como introducción al tema.

4 Este concepto será desarrollado luego en referencia al *nivel simbólico* y el ***nivel base***.

5 Aunque Dannenberg, Desin y Honing no emplean esta terminología, el concepto de modo de acción deriva de las "*action notations*" como forma de representación musical recopilada por Karkoschka. Es empleado en esta **Tesis** en referencia a las acciones del intérprete como factor que indirectamente representa el sonido resultante.

6 En informática, el alcance dinámico es una propiedad que afecta la visibilidad/disponibilidad de las variables en relación al entorno de ejecución. Se emplea el término técnico por la relevancia que el autor de confiere a las técnicas informáticas en relación a la representación de estructuras musicales.

7 Esta sentencia refiere a la especificidad musical del nivel de abstracción más bajo que un entorno elige representar restringiendo el empleo de otras abstracciones musicales posibles no consideradas.

8 En esta **Tesis**, este problema se soluciona conceptualmente mediante la definición del **nivel base** (**Capítulo 4**).

9 Aunque no se emplea de manera sistemática, este concepto proviene del antropólogo E. Cassirer (véase por ejemplo
García Amilburu 1998)

10 Esta esta distinción fue elaborada por el espectralismo francés y estudios posteriores, véase por ejemplo, Bresson et al. (2005), Liimets et al. (2008).

11 La palabra procedimiento es empleada tradicionalmente para definir procesos a una escala temporal mayor aunque en realidad, como se verá más adelante, son entidades indistintas que se diferencia en la escala temporal y el nivel de abstracción en el que se aplican.

12 El definición terminológica de **material (sonoro) concreto** etimológicamente se relaciona con las nociones de Pierre Schaeffer pero refiere a la concepción de **material musical** empleada en esta **Tesis** (véase el apartado siguiente).

13 Véase el apartado “Relaciones abstractas y realización” más adelante en este capítulo.

14 Abstracción estructural paradigmática del serialismo.

15 Véase el apartado siguiente sobre las definiciones terminologías.

16 Incluso la reexposición de cualquier material puede ser considerado como continuidad en la cual no necesariamente hay direccionalidad en el comportamiento. Efectivamente esto produce continuidad formal entendida como coherencia, es una continuidad que se puede lograr libremente, como consecuencia del empleo de distintos recursos. Existe una diferencia entre los procesos representados explícitamente al componer y los que se pueden intuir mediante el análisis. Ambos pueden ser equivalentes, aunque no hayan sido representados explícitamente en la notación se hacen explícitos en el análisis.

17 Si bien los conjunto pueden ser ordenados, se considera en principio, como demostración, que no lo son.

18 Como se vio en el **Capítulo 2**, Dannenberg, Desain y Honing (1997) definen los niveles de “*especificación*”, “*realización*” y “*procesamiento de señales*” en relación a la representación continua/discontinua de los materiales según el nivel de abstracción analizado. Aunque refiere a una cualidad específica (continuidad/discontinuidad) de las señales, el concepto es similar entre las etapas de “*especificación*” y “*realización*” puesto que se basa en principios similares. Pero no debe ser confundido con el concepto aquí expuesto.

19 En este caso, el resultado puede ser equivalente funcionalmente y muy parecido perceptivamente pero no igual.

20 Mediante la programación, obviamente es posible emplear distintos tipos de estructuras de datos para hacer énfasis en la noción de objeto, la programación orientada a objetos emplea este recurso de manera extensa. Lo que importa destacar aquí es la equivalencia entre estructuras de datos más allá de su versatilidad o adecuación a distintos propósitos.

21 https://soundcloud.com/nathaniel-virgo/supercollider-tweet-from-4614

22 http://supercollider.sourceforge.net/sc140/ y http://www.thewire.co.uk/audio/tracks/supercollider-140.1

23 La concepción de programas breves, en una sola línea de código, que generan música no es única de esta comunidad. Existen otros ejemplos en internet, algunos previos, como son: http://electro-music.com/forum/topic-13512-0.html&postdays=0&postorder=asc&highlight=,
https://www.youtube.com/watch?v=GtQdIYUtAHg&list=PLB443007A116CBC54,
http://countercomplex.blogspot.com.ar/2011/10/algorithmic- symphonies-from-one-line-of.html.

24 https://soundcloud.com/nathaniel-virgo/supercollider-tweet-from-4614

25 http://toplap.org

26 http://plork.princeton.edu

27 http://www.steinberg.net

28 http://www.steinberg.net

29 http://www.rosegardenmusic.com

30 http://debussy.music.ubc.ca/NoteAbility

31 https://www.ableton.com

32 https://lmms.io

33 http://jackaudio.org

34 https://github.com/RogueAmoeba/Soundflower

35 https://www.propellerheads.se/developer/rewire.php

36 La técnica condiciona las posibilidades de manipulación y los posibles resultados sonoros, desarrollar una técnica es explorar sus posibilidades. Las ilusiones son un ejemplo que demuestran esto en relación a la naturaleza de nuestra percepción.

37 http://www.sciss.de/scalaCollider

38 http://overtone.github.io

39 https://github.com/crucialfelix/supercolliderjs

40 https://github.com/maca/scruby

41 http://hackage.haskell.org/package/hsc3

42 El procesamiento *bufferizado* de la información transmitida por *buses* de datos es una técnica informática común puesto que la transmisión de datos suele ser más lenta que su procesamiento, e.g. entre el procesador y la memoria o un dispositivo externo. Se refiere aquí a su introducción en el ámbito de los lenguajes de programación para procesamiento de señales de audio.

43 Disponible online en http://doc.sccode.org/Classes/SCDoc.html y distribuida junto con el programa http://supercollider.github.io

44 Los detalles del **modelo** de representación musical en relación a las estructuras de datos serán explicados en el **Capítulo 6**.

45 http://doc.sccode.org/Guides/UsingQuarks.html

46 http://doc.sccode.org/Overviews/JITLib.html

47 https://github.com/supercollider-quarks/Ctk

48 Esta librería es una adaptación a *SuperCollider*, realizada por el autor de esta **Tesis**, de la librería PCSLib (Cetta y Di Liscia 2010) original para *Pure Data*.

49 http://toplap.org

50 Es importante tener en cuenta que los significados asignados son meramente ilustrativos de casos particulares y no implican una relación estática con el nivel base.

51 La representación gráfica es una ayuda visual para entender y manipular los elementos representados. Pero este modelo de representación **no depende de una representación gráfica particular** sino de su lógica estructural que es independiente del medio simbólico de representación.

52 Más adelante en este capítulo se explica como estos principios funcionan y se relacionan como estructuras de datos.

53 Ejemplo extraído de http://countercomplex.blogspot.com.ar/2011/10/algorithmic-symphonies-from-one-line- of.html

54 El término “evento” es empleado de manera extendida para referir a acciones o sucesos. Aquí es empleado para nombrar una primitiva del sistema de representación que se relaciona con estos significados. Para evitar posibles confusiones, siempre que la palabra “evento” refiera a la primitiva de representación se escribirá en negrita.

55 Hacer esta salvedad es importante puesto que son factores que deben ser tenidos en cuenta al analizar obras compuestas en entornos específicos.

56 https://lmms.io

57 Aunque en casos específicos puede generar señales continuas se considera taxonómicamente como discontinua.

58 https://lmms.io

59 En esta sección, los números de línea se refieren al ejemplo Código 16 salvo que se explicite de otra manera.

60 Difieren en el grado de detalle de las relaciones de los elementos representados solo por decisión en el análisis.

61 Los argumentos *gate* y *fadeTime* fueron creados automáticamente en la definición del instrumento *sine*, al definir la envolvente ASR mediante la clase *EnvGate* en la línea 19.

62 Estrenada el 1 de Agosto de 2014 en el Centro Mexicano para la Música y las Artes Sonoras de la Ciudad de Morelia, México (CMMAS). Clarinete Bajo: Alfredo Valdés Brito. Los materiales lógicos y el **registro** de audio de la sección analizada fueron provistos por el compositor.

63 El canal *W* contienen la información omnidireccional por lo que es útil para representar la totalidad del espectro resultante de este estrato independientemente de su espacialización.

64 Los detalles de la implementación no serán tratados debido a la extensión que esto implicaría.

Bibliografía producida y presentada por el autor en reuniones científicas dentro del marco de la presente Tesis

SAMARUGA, L. (2010) SuperCollider 3: La Concepción de un Instrumento y sus Consecuencias en los Roles del Compositor y del Intérprete de Música Electroacústica, en Actas de las I Jornadas de Música de la Escuela de Música de la UNR, Rosario.

SAMARUGA, L. (2011) Modelo de Representación de Información Sonora y Musical, en Memorias del Congreso Latinoamericano de Ingeniería de Audio de la AES, Montevideo.

SAMARUGA, L. (2012) Formas de representación en la informática musical, Jornadas de Becarios y Tesistas 2012 de la UNQ. Bernal, Argentina.

SAMARUGA, L., DI LISCIA, O. P. (2013) “Pitch-class Set design in SuperCollider”. En actas de la Linux Audio Conference, mayo de 2013, IEM, Graz, Austria.

SAMARUGA, L. (2013b) Estructuras Materiales Básicas para la Composición Sonora, en Actas de la Décima Semana de la Música y la Musicología, Buenos Aires.

SAMARUGA, L. (2014) Principios estructurales no semánticos, Jornadas de Becarios y Tesistas 2014 de la UNQ. Bernal, Argentina.

Bibliografía

ABELSON, H., SUSSMAN, G. et al. (1996) Structure and Interpretation of Computer Programs, MIT Press, Massachussets.

AGON, C. (1998) OpenMusic: Un Langage Visuel pour la Composition Assistée par Ordinateur, PhD, Universidad París VI.

AGON, C., STROPPA, M. AND ASSAYAG, G. (2000) High Level Musical Control of Sound Synthesis in OpenMusic, Actas de la International Computer Music Conference, Berlin, Germany.

AGON, C., ASSAYAG, G., BRESSON, J (2006) The OM Composer's Book.1 Collection Musique/Sciences, IRCAM, Editions Delatour, Francia.

ALLEN, J. F. (1983), Maintaining knowledge about temporal intervals, Communications of the ACM, 26:11.

ANDERSON, D., KUIVILA, R. (1991) Formula: a programming language for expressive computer music. IEEE Computer 24(7): 12-21.

ARIZA, C. 2005. An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL. Ph.D. Dissertation, New York University. Available online at http://www.flexatone.net/ caac.html

ARIZA, C. (2008) “Python at the Control Rate: athenaCL Generators as Csound Signals.” Csound Journal 9. Available online at http://www.csounds.com/ journal/ issue9/ pcragcs.html

ASSAYAG, G., RUEDA, C. LAURSON, M. et al. (1999) Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic, Computer Music Journal, 23:3:59-72, Mit Press, Massachussets.

BARTHELEMY, J., BONARDI, A., CIAVARELLA, R., BOUTARD, G. (2009) Virtualization of real time audio processes: towards a musical notation of contemporary music, en Proceedings: Cultural Heritage on line. Empowering users: an active role for user communities, Florence.

BERNARDINI, N., VIDOLIN, A. (2005) Sustainable Live Electro-acoustic Music, en Proceedings of the International Sound and Music Computing Conference, Salerno, Italy.

BILBAO, S. (2009) Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics, John Wiley & Sons, Singapore.

BLECHMANN, T. (2010) supernova, a multiprocessor-aware synthesis server for SuperCollider, en Proceedings of the Linux Audio Conference - LAC 2010.

BLECHMANN, T. (2011) Supernova - A Multiprocessor Aware Real-Time Audio Synthesis Engine For SuperCollider, Master Thesis, Faculty of Informatics, Vienna University of Technology.

BOULANGER, R. C. (2000) The Csound Book, Mit Press, Massachussets. BOULEZ, P. (1957) Le Marteau sans maître, Partitura, London: Universal Edition.

BORIN, G., DE POLI, G., SARTI, A. (1997) en Musical Signal Processing, Capítulo 8, p. 271, Swets & Zeitlinger Publishers, Lisse, the Netherlands.

BREGMAN, A. (1990) Auditory scene analysis: the perceptual organization of sound, The MIT Press, Cambridge, MA.

BRESSON, J., STROPPA, M., AGON, C. (2005) Symbolic control of Sound Synthesis in Computer Assisted Composition, en actas de la International Computer Music Conference, Barcelona, Spain.

BRESSON, J., AGON, C. (2007) Musical Representation of Sound in Computer-Aided Composition: A Visual Programming Framework, Journal of New Music Research, vol 36, no. 4.

BRESSON, J. (2007) La synthèse sonore en composition musicale assistée par ordinateur, PhD, Universidad París VI. Francia.

BRESSON, J., AGON C. (2008) Scores, Programs, and Time Representation: The Sheet Object in OpenMusic, Computer Music Journal, 32:4:31-47, Mit Press,

Massachussets.

BUKVIC, I. I., MARTIN, T., STANDLEY, E., MATTHEWS, M. (2010) Introducing L2Ork: Linux Laptop Orchestra, en Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME 2010), Sydney, Australia.

CÁDIZ, R. (2008) Propuestas Metodológicas para el Análisis de Música Electroacústica, Revista Resonancias, No.23, Pontificia Universidad Católica de Chile.

CÁDIZ, R. (2010) Introducción a la Música Computacional, edición on-line.

CAMCI, A. (2012) A Cognitive Approach to Electronic Music: Theoretical and Experiment- based Perspectives, en Proceedings of the International Computer Music Conference 2012.

CETTA, P., DI LISCIA, P. (2010) Elementos de Contrapunto Atonal. Buenos Aires: EDUCA, Universidad Católica Argentina.

CHOWNING, J. (1995) Prefacio, The computer Music Tutorial, Roads, C., MIT Press, Cambridge.

CODUYS, T., LEFÈVRE, A., PAPE, G., (2003) IanniX. Journées d'Informatique Musicale, École Nationale de Musique, Montbeliard.

CODUYS, T., FERRY, G., (2004) IanniX Aesthetical/Symbolic visualisations for hypermedia composition Sound & Music Computing, IRCAM, Paris.

COINTE, P. RODET, X. (1984a) Formes: An object and time oriented system for music composition and synthesis, en Proceedings of the 1984 ACM Symposium on LISP and functional programming, pages 85–95. ACM Press.

COINTE, P., RODET, X. (1984b) Formes: Composition and scheduling of processes. Capítulo del Libro: The Music Machine, pages 405–423. MIT Press.

COLLINS, N., MCLEAN, A., ROHRHUBER, J., WARD, A. (2003). Live Coding in Laptop Performance. Organised Sound, 3 (8), pp. 321 - 330.

DUIGNAN, M. NOBLE, J. & BIDDLE, R. (2005) A Taxonomy of Sequencer User-Interfaces, en Proceedings of ICMC 2005, Barcelona: Escola Superior de Música de Catalunya.

KARKOSCHKA, E. (1972) Notation in New Music, Koening, R. trad. Praeger Publishers, New York.

DANNENBERG, R. (1996) “Extending Music Notation Through Programming,” in Computer Music in Context, Craig Harris, ed., Contemporary Music Review series, Vol. 13, Part 2, Harwood Academic Publishers, pp. 63-76.

DANNENBERG, (1997) “Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis,” Computer Music Journal, 21(3), pp. 50-60.

DANNENBERG, R. B. (2007) Nyquist Reference Manual, Version 2.36, Carnegie Mellon University, http://www.cs.cmu.edu/~rbd/nyquist.html

DANNENBERG, R.B., DESAIN, P., HONING, H. (1997) en Musical Signal Processing, Capítulo 8, p. 271, Swets & Zeitlinger Publishers, Lisse, the Netherlands.

DANNENBERG, R. B., RUBINE, D., NEUENDORFFER, T. (1991) The resource-instance model of music representation, en Porceedings of the 1991 International Computer Music Conference, San Francisco, pp 428-432.

DEUTSCH, D. Ed. (2013) The psychology of music, 3nd Edition, Elsevier, San Diego. DEUTSCH, D. (1974) An illusion with musical scales. Journal of the Acoustical Society of America, 56, s25.

DI LISCIA O. P., PAMPIN J. (2003) Spectral analysis based synthesis and transfomation of digital sound: the ATSH program, IX Simposio Internacional de Computación y Música, Minas Gerais, Brasil, Agosto de 2003.

DI LISCIA, O. P. (2004) Sintesis y Procesamiento de Sonido y Musica a través del programa Csound. Colección Música y Ciencia, Editorial UNQ.

DI LISCIA, O. P. (2013) Estructura jerárquica de invariantes en la klavierstuck VII de

Karlheinz Stockhausen. Revista del Instituto de Investigación Musicológica “Carlos Vega” No27, 81-95, UCA, Bs.As., Argentina.

DODGE, CH. & JERSE, T. (1985) Computer Music Synthesis, composition and performance, Shirmer Books, New York.

DOWNIE, J. (2003) Music information retrieval, en Annual Review of Information Science and Technology 37, ed. Blaise Cronin, 295-340. Medford, NJ: Information Today. Available from http://music-ir.org/downie_mir_arist37.pdf

ECKEL, G. (1993) La maîtrise de la synthèse sonore, en Les cahiers de l'Ircam 2 – La synthèse sonore, Ircam - Centre Pompidou.

ECKEL, G., GONZÁLEZ-ARROYO, R. (1994) Musically Salient Control Abstractions for Sound Synthesis, Proceedings of the 1994 International Computer Music Conference, Aarhus, pp. 256-259.

FINEBERG, J. (2000) Spectral music: history and techniques. Overseas Publishers Association, published by license under the Harwood Academic Publishers imprint.

FOBER, D., DAUDIN, C., ORLAREY, Y., LETZ, S., (2010) Interlude - A Framework for Augmented Music Scores, Proceedings of the Sound and Music Computing conference - SMC'10 2010.

FOBER, D., DAUDIN, C., ORLAREY, Y., LETZ, S., (2010b) Time Synchronization in Graphic Domain – A new paradigm for Augmented Music Scores, Proceedings of the International Computer Music Conference ICMA 2010.

FOBER, D., DAUDIN, C., ORLAREY, Y., LETZ, S., (2010c) Partitions musicales augmentées, Actes des Journées d'Informatique Musicale JIM2010.

FOBER, D., ORLAREY, Y., LETZ, S., (2012) INScore - An Environment for the Design of Live Music Scores, Proceedings of the Linux Audio Conference - LAC 2012.

FORTE, ALLEN (1974) The Structure of atonal music, Yale University Press, N. H. FORTE, A. (1998) Introducción al Análisis Schenkeriano, Editorial: Labor.

GARCÍA AMILBURU, M. (1998) La cultura como universo simbólico en la antropología de E. Cassirer, Pensamiento n. 209, pp. 221-244.

GESLIN, Y., LEFEVRE, A., (2004) Sound and musical representation: the Acousmographe software. International Computer Music Conference, Miami, USA.

HANAPPE, P., (1999) Design and Implementation of an Integrated Environment for Music Composition and Synthesis, PhD, Universidad Paris VI, Francia.

HEIKKILÄ, V. (2010) Defining Computationally Minimal Art (Or, taking the “8” out of “8-bit”), disponible publicado online en http://pelulamu.net/countercomplex/computationally-minimal- art el 15/03/2010.

HONING, H. (1993). Issues in the representation of time and structure in music. Contemporary Music Review, 9, 221-239.

IVERSON, E. (1980) Notation as tool of thought, en Communications of the ACM, vol. 23, issue 8, 444-465, New York, NY, USA.

JACQUEMIN, G., CODUYS, T., (2014) Partitions rétroactives avec IanniX. Journées d’Informatique Musicale, MUSINFO, Bourges.

JACQUEMIN, G., CODUYS, T., RANC, M., (2012) IanniX 0.8. Journées d’Informatique Musicale Université de Mons, Mons.

JORDA PUIG, S. (1997) Audio digital y MIDI, Anaya Multimedia, Madrid.

KARPLUS, K., STRONG, A. (1983) Digital Synthesis of Plucked String and Drum Timbres, Computer Music Journal, 7 (2): 43–55, MIT Press.

KARKOSCHKA, E. (1972) Notation in New Music, Koening, R. trad. Praeger Publishers, New York.

KENDALL, G. S. (2009) La interpretación de la espacialización electroacústica: atributos espaciales y esquemas auditivos, Capítulo del Libro Música y Espacio: ciencia, tecnología y estética, Basso, G., Di Liscia, O. P. y Pampin, J. editores, pp. 241-260, UNQ Editorial, Bernal.

KERAMANE, C. (1995) Structured Temporal Composition of Multimedia Data, en Actas del International Workshop on Multi-Media.

KOSTKA, S. (1990) Materials and Techniques of Twentieth-Centruty Music, 3ra ed., Pearson, Upper Saddle River, New Jersey.

LAURSON, M. & KUUSKANKARE, M. & NORILO, V. (2009) An Overview of PWGL a Visual Programming Environment for Music, Computer Music Journal, Mit Press, Massachussets.

LIIMETS, A. & KOTTA K. (2008) Music as limes: Preliminary concepts of the process-centered music analysis, en Proceedings of the fourth Conference on Interdisciplinary Musicology (CIM08), Thessaloniki, Greece.

LINDEMANN, E. (2001) Musical synthesizer capable of expressive phrasing. US Patent 6,316,710, 2001.

LETZ, S., FOBER D., ORLAREY, Y. (1998) The Role of Lambda-Abstraction in Elody, ICMC 1998.

MALHAM, D. (2009) El espacio acústico tridimensional y su simulación por medio de Ambisonics, Capítulo del Libro Música y Espacio: ciencia, tecnología y estética, Basso, G., Di Liscia, O. P. y Pampin, J. editores, pp. 161-202, UNQ Editorial, Bernal.

MCLEAN, A., GRIFFITHS, D., COLLINS, N., WIGGINS, G. (2010) Visualisation of live code, en Proceeding EVA'10 Proceedings of the 2010 international conference on Electronic Visualisation and the Arts Pages 26-30.

MAGNUSSON, T., MENDIETA, E. H. (2007). The Acoustic, the Digital and the Body: A Survey on Musical Instruments. En actas de New Interfaces for Musical Expression 2007.

MAGNUSSON, T. (2010a) Designing Constraints: Composing and Performing with Digital Musical Systems, Computer Music Journal, 34:4:62-62, Mit Press, Massachussets.

MAGNUSSON, T. (2010b) An Epistemic Dimension Space for Musical Devices, Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 43–46.

McCARTNEY, J. (1996) SuperCollider: A new real time synthesis language, en Proceedings of the International Computer Music Conference (ICMC'96), pp. 257–258.

McCARTNEY, J. (2002) Rethinking the Computer Music Language: SuperCollider. En Computer Music Journal, 26:4:61-68, Mit Press, Massachussets.

McCARTNEY, J. (2011) en The SuperCollider Book, Prefacio pg. ix, Wilson, S., Cottle, D. and Collins, N. (eds), Cambridge, MA: MIT Press.

MOORE, F. R. (1990) Elements of Computer Music ,Prentice Hall., New Jersey.

MORRIS, R. (1987) Composition with Pitch-Classes: A Theory of Compositional Design, New York: Yale University Press.

PENFOLD, R. A. (1992) MIDI avanzado, RA-MA, Madrid.

PEREVALOV, D., (2013) Mastering openFrameworks: Creative Coding Demystified, Packt Publishing, Birmingham – Mumbai.

POPE, S.T. (1989) Modeling Musical Structures as EventGenerators. Proceedings of the 1989 International Computer Music Conference. San Francisco: Computer Music Association.

POPE, S. T. (1992) The Smoke music representation, description language, and interchange format, en Proceedings of the 1992 International Music Conference, San Francisco.

PUCKETTE, M. (1996) Pure Data: another integrated computer music environment. Proceedings, International Computer Music Conference. San Francisco: International Computer Music Association, pp. 224-227.

PUCKETTE, M. (2002) Max at seventeen. Computer Music Journal 26(4): pp. 31-43. PUCKETTE, M. (2002b) Using Pd as a score language. Actas de la ICMC, pp. 184-187.

PUCKETTE, MILLER (2007) The Theory and Technique of Electronic Music, World Scientific Publishing Co. Pte. Ltd.

REAS, C., FRY, B., (2007) Processing: a programming handbook for visual designers and artists, The MIT Press Cambridge, Massachusetts London, England.

ROADS, C. (1995) The computer Music Tutorial, MIT Press, Cambridge.

ROADS, C., POPE, S. T., PICCIALLI, A., DE POLI, G. (1997). Musical Signal Processing. Swets & Zeitlinger Publishers, Lisse, the Netherlands.

ROADS, C. (2002) Microsound, MIT Press, Cambridge.

ROEDERER, J. G. (1997) Acústica y psicoacústica de la música, Ricordi Americana, Buenos Aires.

SCALETTI, C. (1989) The Kyma/Platypus computer music workstation, Computer Music Journal 8(3): 32-50.

SCHMIDHUBER, J. (1997) Low-Complexity Art, Leonardo Journal of the International Society for the Arts, Sciences, and Technology, vol. 30:2, p. 97-103, MIT Press.

SIGAL, R. (2014) Estrategias compositivas en la música electroacústica: Generación de materiales y creación de un lenguaje musical eficaz, Universidad Nacional de Quilmes Editorial, Bernal.

SMALLEY, D., (1986) Spectro-Morphology and Structuring Processes, Capítulo, en The Language of Electroacoustic Music, pp. 17-40. ed. Simon Emmerson (New York: Harwood).

SQUIBBS, R., (1996) Images of Sound in Xenakis's Mycenae Alpha. Troisièmes journées d'informatique musicale (JIM 96), Conference Proceedings. Les cahiers du GREYC 4 (1996): 208-18.

SCHAEFFER, P. (1988) Tratado de los objetos musicales, Ed. Alianza, Madrid.

STONE, K. (1980) Music Notation in the Twentieth Century, W.W. Norton & Company, New York.

STOCKHAUSEN, K. (1956) Studie II. Partitura, Universal Edition, Wien-Zürich-London. STOCKHAUSEN, K. (1965) Klavierstuck VII. Partitura (UE 13675c) Viena: Universal Edition.

STOCKHAUSEN, K. (1989) Moment-Forming and Momente, en Stockhausen On Music, Lectures and Interviews Compiled by Maconie, R., pp. 63 a 75, Marion Boyars, New York, London.

TAUBE, H. (1991) Common Music: a music composition language in Comon Lisp and CLOS, Computer Music Journal 15(2): 21-32.

TENNEY, J. (1961) Meta + Hodos, en “Meta + Hodos and META Meta + Hodos”, Polansky L. publisher 1988, Frog Peak Music, Oakland. (datos extraídos de la introducción)

TENNEY, J., POLANSKY, L. (1980). Temporal Gestalt perception in music. Journal of Music Theory, 24, 205-241.

THIEBAUT, J., HEALEY, P., BRYAN KINNS, N., (2008) Drawing Electroacoustic Music, International Computer Music Conference (ICMC), Belfast.

TRUEMAN, D., COOK, P., SMALLWOOD, S., and WANG, G. (2006) PLOrk: Princeton Laptop Orchestra, Year 1, en Proceedings of the International Computer Music Conference, New Orleans, U.S.

VERCOE, B. 1986. CSOUND: A Manual for the Audio Processing System and Supporting Programs. MIT Media Lab.

WANG G., COOK P. R. (2003) ChucK: A Concurrent, On-the-fly, Audio Programming Language, Actas de la International Computer Music Conference. International Computer Music Association, pp. 219-226.

WANG G., COOK P. R. (2004a) ChucK: A Programming Language for On-the-fly, Real-time Audio Synthesis and Multimedia, En ACM Multimedia, New York City, U.S.A., 2004.

WANG G., COOK P. R. (2004b) The Audicle: A Context-Sensitive, On-the-fly Audio Programming Environ/mentality, Actas de la International Computer Music Conference, Miami, U.S.A., 2004.

WANG, G., COOK, P. (2004) On-the-fly Programming: Using Code as an Expressive Musical Instrument, en New Interfaces for Musical Expression (NIME), Hamamatsu, Japan.

WANG G., COOK P. R. (2005) Designing and implementing the chuck programming language, En International Computer Music Conference, Barcelona, Spain, 2005.

WANG G. (2008) The ChucK Audio Programming Language “A Strongly- Timed and On-The- Fly Environ/Mentality”, PhD, Faculty of Princeton University in Candidacy for the Degree of Doctor of Philosophy, http://chuck.cs.princeton.edu

WILSON, S., COTTLE, D. and COLLINS, N., editores (2011) The SuperCollider Book. Cambridge, MA: MIT Press.

WINDSOR, W. L. (1995). A Perceptual Approach to the Description and Analysis of Acousmatic Music. Doctoral Thesis, City University.

WINSOR, P., DELISA, G. (1991) Computer Music in C, University of North Texas Press, Denton, Texas.

WISHART, T. (1996) On Sonic Art, Routledge Tylor & Francis Group, New York.

WRIGHT M., FREED A., MOMENI A. (2003) OpenSound Control: State of the Art 2003, Actas de la Conference on New Interfaces for Musical Expression (NIME-03), Montreal, Canada.

XENAKIS, I. (1971) Formalized Music, Bloomington, Indiana University Press.

XENAKIS, I. (1997) Xenakis – Electronic Music, CD Compilado, Electronic Music

Foundation - EMF CD 003

YI, S. (2011) Blue, a music composition environment for Csound, en http://blue.kunstmusik.com, página web accedida por última vez el 10 de agosto de 2011.

Recursos en Internet

Página original de SuperCollider elaborada por James McCartney: http://audiosynth.com/index.html

Homepage de SuperCollider en Github: http://supercollider.github.io

Antigua homepage de SuperCollider en Sourceforge: http://supercollider.sourceforge.net

Sitio web de código en SuperCollider y docuemntación online: http://sccode.org

Página wiki de SuperCollider en Sourceforge: http://supercollider.sourceforge.net/wiki

Antigua página wiki de SuperCollider: http://swiki.hfbk-hamburg.de:8888/MusicTechnology/6

Edición online del álbum “sc140” (*sctweets*): http://supercollider.sourceforge.net/sc140/ y http://www.thewire.co.uk/audio/tracks/supercollider-140.1

Documentación de *SuperCollider*: http://doc.sccode.org/Classes/SCDoc.html

Ídem: http://doc.sccode.org/Guides/UsingQuarks.html Ídem: http://doc.sccode.org/Overviews/JITLib.html Ídem: https://github.com/supercollider-quarks/Ctk

Lenguaje alternativo a *sclang*: http://www.sciss.de/scalaCollider

Ídem: http://overtone.github.io

Ídem: https://github.com/crucialfelix/supercolliderjs

Ídem: https://github.com/maca/scruby

Ídem: http://hackage.haskell.org/package/hsc3

Música por algoritmos en una línea de código en C: http://countercomplex.blogspot.com.ar/2011/10/algorithmic-symphonies-from-one-line-of.html

Ídem: https://www.youtube.com/watch?v=GtQdIYUtAHg&list=PLB443007A116CBC54

Ídem: http://countercomplex.blogspot.com.ar/2011/10/algorithmic-symphonies-from-one-line-of.html

Nathaniel Virgo: https://soundcloud.com/nathaniel-virgo/supercollider-tweet-from-4614

Música en una sola línea de código en ChucK: http://electro-music.com/forum/topic-13512-0.html&postdays=0&postorder=asc&highlight=

Software

Ableton Live: https://www.ableton.com
Jack: http://jackaudio.org
LMMS: https://lmms.io
NoteAbilityPro: http://debussy.music.ubc.ca/NoteAbility
Nuendo: http://www.steinberg.net
Rewire: https://www.propellerheads.se/developer/rewire.php
Rosegarden: http://www.rosegardenmusic.com
Soundflower: https://github.com/RogueAmoeba/Soundflower
Tecnología VST: http://www.steinberg.net
Nota: Todas las páginas web fueron revisadas por última vez el 25/02/2015

Apéndice 1

Archivo “Part y Orq.scd”, **Autor**: Oscar Pablo Di Liscia.

```
<CsoundSynthesizer>

<CsOptions>
</CsOptions>
<CsInstruments>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
sr  =44100
ksmps          =100
nchnls         =4
0dbfs          =1

#include "../common/grainers_and_macros.csd"
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
</CsInstruments>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
<CsScore>
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;TABLE WITH ROOM PARAMETERS FOR SPAT3D IS ALWAYS 99
#define        WL         #          1          #
#define        CFL        #          .5         #
#define        FC         #          3000       #
#define        BP         #          0          #
#define        LP         #          2          #
#define        HP         #          1          #
#define RNDV   #          .1         #
#define        NOE        #0#
#define        FO         #1#
#define        SO         #2#
#define FLTL   #.5#
;                                    Er         NC         2          NC         4          5
f99 0          64         -2         10         3          -1         -1         -1         -1
;0  1          2          3          4          5          6          7
1   3.5        $RNDV      $CFL       $FC        $FLTL      .7071      $LP
1   1.5        $RNDV      $CFL       $FC        $FLTL      .7071      $LP
1   15         $RNDV      $WL        $FC        $FLTL      .7071      $LP
1   15         $RNDV      $WL        $FC        $FLTL      .7071      $LP
1   15         $RNDV      $WL        $FC        $FLTL      .7071      $LP
1   15         $RNDV      $WL        $FC        $FLTL      .7071      $LP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
f31 0          1024       -7         2          512        .1         512        2                                ;grain duration table (scaling)
f32 0          1024       -7         .1         256        1          256        .25        512        1          ;grain duration table (scaling)
f33 0          1024       -7         1          1024       .1                                                    ;grain duration table (scaling)
f34 0          1024       -7         1          512        .05        512        1
f35 0          1024       -7         .1         512        1          512        .1
f36 0          1024       -7         .5         512        1          512        .5
f37 0          1024       -7         1          1024       .33

f41 0          1024       -7         1.05       256        1          768        1.05       ;grain     freq.      dev. table (scaling)
f42 0          1024       -7         1.05       1024       1                                ;grain     freq.      dev. table (scaling)
f43 0          1024       -7         1          1024       1.05                             ;grain     freq.      dev. table (scaling)
f44 0          1024       -7         2          512        1          512        2          ;grain     freq.      dev. table (scaling)
f45 0          1024       -7         1          512        2          512        1          ;grain     freq.      dev.
```

```
    table     (scaling)
f46 0         1024       -7         1.26       1024       1                              ;grain     freq.      dev.
    table     (scaling)
f47 0         1024       -7         1          1024       .793                           ;grain     freq.      dev.
    table     (scaling)
f48 0         1024       -7         1          1024       .5                             ;grain     freq.      dev.
    table     (scaling)

f51 0         1024       -7         0          1024       360        ;azimuth   table     (offset)
f52 0         1024       -7         360        1024       0          ;azimuth   table     (offset)
f53 0         1024       -7         0          1024       -360       ;azimuth   table     (offset)

f61 0         1024       -7         5          512        0          512        5         ;distance  table
    (offset)
f62 0         1024       -7         0          512        5          512        0         ;distance  table
    (offset)
f63 0         1024       -7         0          1024       8                             ;distance  table
    (offset)
f64 0         1024       -7         8          1024       0                             ;distance  table
    (offset)

f71 0         1024       -7         1          512        10         512        1         ;res       bw         table
    (scaling)
f72 0         1024       -7         1          1024       10                            ;res       bw         table
    (scaling)
f73 0         1024       -7         10         1024       1                             ;res       bw         table
    (scaling)

f80 0         8          -2         .1         .2         .3         .5         .7        .9         ;gap duration
table

f90 0         8          -2         8.00       8.01       8.04       8.06       8.00      8.01       8.04       8.06
    ;pitch-class table for FM
f91 0         8          -2         0          1          4          6          0         1          4          6
    ;pitch-class table for ATS and SND

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;WARNING: tables from #10 to #27 are used
;envelopes tables and their macros
#define        p_cce      #8#
#define        p_cve      #9#
#define        p_cnc      #10#
#define        p_cnv      #11#
#define        f_cnv      #12#
#define        f_cnc      #13#
#define        io1        #14#
#define        io2        #15#
#define        io3        #16#
#define        io4        #17#
#define        oi1        #18#
#define        oi2        #19#
#define        oi3        #20#
#define        oi4        #21#
#define        stat1      #22#
#define        stat2      #23#
#define        stat3      #24#
#define        stat4      #25#
#define        stat5      #28#
#define        f_lin      #26#
#define tri    #27#
#define fix    #7#

f7  0         1024       7          1          1024       1
f8  0         4096       16         1          26         3          1          4070      -3         0
f9  0         4096       16         1          26         -3         1          4070      3          0
f10 0         4096       16         0          26         3          1          4070      -3         0
f11 0         4096       16         0          26         -3         1          4070      3          0
f12 0         4096       16         0          4040       -3         1          56        3          0
f13 0         4096       16         0          4040       3          1          56        -3         0
f14 0         512        16         0          256        3          1          256       -3         0
f15 0         512        16         0          256        -3         1          256       3          0
f16 0         512        16         0          256        -3         1          256       -3         0
f17 0         512        16         0          256        3          1          256       3          0
f18 0         512        16         0          16         -3         1          240       3          .1         240
    -3        1          16         3          0
f19 0         512        16         0          16         -3         1          240       -3         .1         240
    3         1          16         3          0
f20 0         512        16         0          16         -3         1          240       -3         .1         240
    -3        1          16         3          0
f21 0         512        16         0          16         -3         1          240       3          .1         240
    3         1          16         3          0
f22 0         512        16         0          8          3          1          496       0          1          8
    -3        0
f23 0         512        16         0          16         3          1          480       0          1          16
    -3        0
f24 0         512        16         0          32         3          1          448       0          1          32
```

```
    -3         0
f25 0          512        16         0          64         3          1          384        1          1          64
    -3         0
f26 0          1024       7          0          1024       1
f27 0          1024       7          0          512        1          512        0
f28 0          512        16         0          128        3          1          256        1          1          128
    -3         0

f29 0          1024       7          0          512        1          512        0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;X/Y COORDINATES TABLE
;THE SAME TABLE SHIFTED BY 90 DEGREES IS USED TO GET A COSINE WAVE
#define sine # 5 #
f5  0          32768      10         1
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;ATS MACROS
#define        CLC        #1#
#define        CLCs       #2#
#define        CLA        #3#

#define dCLC # 1.5 #
#define pCLC # 42 #
#define fCLC # 3.37 #
#define fuCLC # 261.62 #

#define dCLCs  # 1.71 #
#define pCLCs  # 51 #
#define fCLCs  # 5.56 #
#define fuCLCs # 69.3 #

#define dCLA  # 1.08 #
#define pCLA  # 38 #
#define fCLA  # 2.77 #
#define fuCLA # 220 #

#define NONOIS # 0 0 0 #

#define        se(s)      #[2^($s/12)]#
#define fracdev(integer's) #[$integer+(2^($s/12))*.1]#

;single note macro
#define ats_single_grain(start'dur'amp'tamp'from'to'fdev'az'el'dist'file'off'step'np'nlev'tnois'nof'nstep'nbands'ilowf)
#
i "ats_grain" $start $dur $amp $tamp $from $to $file $fdev $az $dist $el $dur 0 0 0 0 0 $start $dur $off $step $np
$nlev $tnois $nof $nstep $nbands $ilowf
#

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;FM macros
#define strand # .1 [~+8] .01 [~+8] .25 [~+8] #
#define norand # 0 0 0 0 0 0 #
#define raonly(de'fr) # $de $fr 0 0 0 0 #
#define rfonly(de'fr) # 0 0 $de $fr 0 0 #
#define rionly(de'fr) # 0 0 0 0 $de $fr #
#define fracdev(integer's) #[$integer+(2^($s/12))*.1]#
#define fm_single_r(st'dur'amp'atab'az'el'dist'fc'ctab'rat'ind'iof'tind)
#
i "fm_grain" $st $dur $dur $amp $atab $az $el $dist 0 0 0 0 0 $fc $ctab  [$fc*$rat] $ind $iof $tind 0 $strand $st $dur

#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define fm_single_nr(st'dur'amp'atab'az'el'dist'fc'ctab'rat'ind'iof'tind)
#
i "fm_grain" $st $dur $dur $amp $atab $az $el $dist 0 0 0 0 0 $fc $ctab  [$fc*$rat] $ind $iof $tind 0 $norand $st $dur

#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define fm_single_ar(st'dur'amp'atab'az'el'dist'fc'ctab'rat'ind'iof'tind'ad'af)
#
i "fm_grain" $st $dur $dur $amp $atab $az $el $dist 0 0 0 0 0 $fc $ctab  [$fc*$rat] $ind $iof $tind 0 $raonly($ad'$af)
$st $dur
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define fm_single_fr(st'dur'amp'atab'az'el'dist'fc'ctab'rat'ind'iof'tind'fd'ff)
#
i "fm_grain" $st $dur $dur $amp $atab $az $el $dist 0 0 0 0 0 $fc $ctab  [$fc*$rat] $ind $iof $tind 0 $rfonly($fd'$ff)
$st $dur
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define fm_single_ir(st'dur'amp'atab'az'el'dist'fc'ctab'rat'ind'iof'tind'id'imf)
#
i "fm_grain" $st $dur $dur $amp $atab $az $el $dist 0 0 0 0 0 $fc $ctab  [$fc*$rat] $ind $iof $tind 0 $rionly($id'$imf)
```

```
$st $dur
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;grainer SND MACROS
#define CLASC5  #5#
#define CLASCs  #6#
#define CLASA   #7#
#define AIRS   #8#
#define        se(s)      #[2^($s/12)]#
#define fracdev(integer's) #[$integer+(2^($s/12))*.1]#
#define  ARMC(n's) # [((2^($s/12))*261.6)*$n] #
;filter managment macros
#define  NOFLT                      # 0 0 0 #             ;no filtering
#define  NVFLT(fc'bw)               # $fc $bw 0 #                    ;time invariant filtering
#define  VFLTF(fc'bw'tf)  # $fc $bw $tf #                 ;time variant filtering forward
#define  VFLTB(fc'bw'tf)  # $fc $bw -$tf #      ;time variant filtering backward
#define  VFLTR(fc'bw'of)  # $fc -$bw $of #      ;random bw
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;
t   0          60
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;
;CONEXION CON SEC. ANT.
;Comienzo en 145 segs. (2'25") Compas 116 de la partitura.
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev                 gfdr
i "grainer"    0          3.8        .5         0.0        $p_cnv     .4         .2         .05        0.01       0.0
    0.0        $se(-4)    0.01
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               90         10         0          0          3          0          $CLASA     0          0          0
    0          0          0          26.09      $NVFLT(7000'.175)
;CONVERGENTE
-----------------------------------------------------------------------------------------------------------------
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    3.9        5          .5         .1         $p_cnv     .66        0.0        .05        0.0        0.2
    0.0        $se(-4)    0.0
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               90         0          0          0          1          0          $CLASA     0          0          0
    52.0125    64         0          0          $NVFLT(11000'.175)

;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $p_cnv     .          0.0        .05        0.0        0.2
    0.0        $se(-4)    0.0
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               90         0          0          0          2          0          $CLASA     0          0          0
    51.0125    64         0          0          $NVFLT(5500'.175)
;DIVERGENTE
1-----------------------------------------------------------------------------------------------------------------
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    9.05       4.64       .          .          $io4       .45        0.01       .05        0.025      0.2
    0.1        $se(-3)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               45         10         0          0          1          0          $CLASA     0          0          0
    0          63         0          0          $NVFLT(11000'.2)

;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io3       .45        0.01       .05        0.025      0.2
    0.1        $se(-2)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               135        10         0          0          1          0          $CLASA     0          0          0
    0          63         0          0          $NVFLT(5500'.2)
;DIVERGENTE
2-----------------------------------------------------------------------------------------------------------------
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    14.25      6.3        .          .          $io1       .5         0.05       .05        0.05       0.2
    0.1        $se(-4)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               45         30         0          0          1          0          $CLASA     0          0          45.05
    0          61         0          0          $NVFLT(11000'.2)

;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
```

```
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io2       .5         0.05       .05        0.05       0.2
    0.1        $se(-1)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               135        30         0          0          1          0          $CLASA     0          0          48
    0          61         0          0          $NVFLT(5500'.2)
;DIVERGENTE 1 hacia
convergente-----------------------------------------------------------------------------------------------------------
-----
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    20.6       5          .          .          $io3       .5         0.1        .05        0.05       0.2
    0.1        $se(-5)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               45         10         0          0          1          0          $CLASA     0          0          0
    0          61         0          0          $NVFLT(12000'.15)

;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io2       .5         0.1        .05        0.05       0.2
    0.1        $se(0)     0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               135        10         0          0          1          0          $CLASA     0          0          0
    0          61         0          0          $NVFLT(4500'.15)
;DIVERGENTE 1 hacia convergente pero atras(270
Deg)------------------------------------------------------------------------------------------
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    26         2.3        .          .          $io2       .5         0.05       .05        0.05       0.2
    0.1        $se(-4)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               45         0          0          0          1          0          $CLASA     0          0          -46
    52.0375    61         0          0          $NVFLT(12000'.2)

;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io1       .5         0.05       .05        0.05       0.2
    0.1        $se(-1)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               135        0          0          0          1          0          $CLASA     0          0          47
    51.0375    61         0          0          $NVFLT(4500'.2)
;
;CONVERGENTE 4 ESTRATOS------------------------------------------------------------------------------------------------
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    28.5       6.3        .2         .2         $io1       .45        0.0        .15        0.05       0.2
    0.1        $se(-4)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               270        0          0          0          5          0          $CLASA     0          0          0
    0          0          0          0          $NVFLT(13000'.15)
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io1       .45        0.0        .          0.05       0.2
    0.1        $se(-3)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               270        0          0          0          5          0          $CLASA     0          0          0
    0          0          $f_lin     0          $NVFLT(8000'.15)
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io1       .45        0.0        .          0.05       0.2
    0.1        $se(-1)    0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               270        0          0          0          5          0          $CLASA     0          0          0
    0          0          0          0          $NVFLT(4000'.15)
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io1       .45        0.0        .          0.05       0.2
    0.1        $se(0)     0.025
;              gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
               270        0          0          0          5          0          $CLASA     0          0          0
    0          0          $f_lin     0          $NVFLT(2000'.15)

;HACIA DIVERGENTE 4 ESTRATOS (0, 90, 180,
270)-------------------------------------------------------------------------
;              st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    34.85      9.15       .1         .1         $io1       .55        0.01       .35        0.1        0.4
```

```
    0.2         $se(-4)     0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              270         0           0           0           1           0           $CLASA      0           0           0
    0           62          0           0           $NVFLT(13000'.1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           0.01        .           0.1         0.4
    0.2         $se(-3)     0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              270         0           0           0           .           0           $CLASA      0           0           0
    51.05       .           0           0           $NVFLT(8000'.1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           0.01        .           0.1         0.4
    0.2         $se(-1)     0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              270         0           0           0           .           0           $CLASA      0           0           0
    53.025      .           0           0           $NVFLT(4000'.1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           0.01        .           0.1         0.4
    0.2         $se(0)      0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              270         0           0           0           .           0           $CLASA      0           0           0
    51.025      .           0           0           $NVFLT(2000'.1)
;DIVERGENTE 4 ESTRATOS (0, 90, 180, 270)-------------------------------------------------------------------------------
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     44.4        11.8        .1          .1          $io1        .6          0.2         .4          0.2         0.0
    0.2         $se(-1.5)   0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              270         10          0           0           1           0           $CLASA      0           0           0
    0           63          0           0           $NVFLT(13000'.1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           .           .           0.1         .
    0.2         $se(-1)     0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              90          0           0           0           .           0           $CLASA      0           0           0
    .           .           0           0           $NVFLT(8000'.1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           .           .           0.1         .
    0.2         $se(-0.5)   0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              180         0           0           0           .           0           $CLASA      0           0           0
    .           .           0           0           $NVFLT(4000'.1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           .           .           0.1         .
    0.2         $se(0)      0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              0           0           0           0           .           0           $CLASA      0           0           0
    .           .           0           0           $NVFLT(2000'.1)
;+DIVERGENTE 4 ESTRATOS (0, 90, 180, 270) ROTACION
HORARIA---------------------------------------------------------------------------
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     56          10          .1          .1          $io1        .6          0.3         .5          0.2         0.0
    0.2         $se(-1.5)   0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              180         10          0           0           2           0           $CLASCs     0           0           0
    0           61          0           0           $VFLTF(12000'.2'$io1)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           .           .           0.1         .
    0.2         $se(-1)     0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
              0           0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(8000'.2'$f_cnv)
;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
i "grainer"     .           .           .           .           .           .           .           .           0.1         .
    0.2         $se(-.5)    0.025
;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
```

```
343                 90          0           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(4000'.2'$f_cnc)
344 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
345 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(0)      0.025
346 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
347                 270         0           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(2000'.2'$io2)
348 ;+DIVERGENTE 4 ESTRATOS (0, 90, 180, 270) ROTACION
    HORARIA-----------------------------------------------------------------------------
349 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
350 i "grainer"     66.5        10.5        .1          .1          $io1        .6          0.3         .6          0.4         0.0
        0.2         $se(-1.5)   0.025
351 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
352                 90          10          0           0           1           0           $CLASCs     0           0           0
        0           61          0           0           $VFLTF(11000'.2'$io3)
353 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
354 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(-1)     0.025
355 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
356                 270         .           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(7500'.2'$p_cnv)
357 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
358 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(-.5)    0.025
359 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
360                 0           .           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(4500'.2'$p_cnc)
361 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
362 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(0)      0.025
363 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
364                 180         .           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(2500'.2'$io4)
365
366 ;+DIVERGENTE 4 ESTRATOS (0, 90, 180, 270) ROTACION
    HORARIA-----------------------------------------------------------------------------
367 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
368 i "grainer"     76.5        11.5        .1          .1          $io1        .7          0.3         .8          0.5         0.0
        0.2         $se(-1.5)   0.025
369 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
370                 0           20          0           0           2           0           $CLASCs     0           0           0
        0           61          0           0           $VFLTF(11000'.2'$f_cnv)
371 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
372 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(-1)     0.025
373 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
374                 180         0           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(7500'.2'$io4)
375 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
376 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(-.5)    0.025
377 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
378                 270         0           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(4500'.2'$io2)
379 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
380 i "grainer"     .           .           .           .           .           .           .           .           0.1         .
        0.2         $se(0)      0.025
381 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
        taz         tdis        tamp        tfp         fc          bw          bwt
382                 90          0           0           0           .           0           .           0           0           0
        .           .           0           0           $VFLTF(2500'.2'$f_cnc)
383 ;+DIVERGENTE 4 ESTRATOS (0, 90, 180, 270) ROTACION
    HORARIA-----------------------------------------------------------------------------
384 ;               st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
        gstr        gfdev       gfdr
385 i "grainer"     88          15          .1          .1          $io1        .8          0.3         1           0.5         0.0
        0.2         $se(-1)     0.025
386 ;               gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
```

```
    taz         tdis        tamp        tfp         fc          bw          bwt
387             270         45          0           0           2           0           $CLASCs     0           0           0
    0           61          0           0           $VFLTF(10000'.2'$p_cnv)
388 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev                   gfdr
389 i "grainer"     .           .           .           .           .           .           .           0.1         .
    0.2         $se(-.66)   0.025
390 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
391             90          0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(6500'.2 '$io3)
392 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
393 i "grainer"     .           .           .           .           .           .           .           0.1         .
    0.2         $se(-.33)   0.025
394 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
395             180         0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(5000'.2 '$io1)
396 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
397 i "grainer"     .           .           .           .           .           .           .           0.1         .
    0.2         $se(0)      0.025
398 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
399             0           0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(3500'.2 '$p_cnc)
400 ;+DIVERGENTE 4 ESTRATOS (0, 45, 135, 180) CONCENTRANDO SOBRE EL
    FRENTE--------------------------------------------------------------------------
401 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev                   gfdr
402 i "grainer"     102.5       20          .1          .1          $io1        1.2         0.1         2.5         0.5         0.0
    0.2         $se(-.3)    0.025
403 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
404             0           20          0           0           6           0           $CLASCs     0           0           0
    0           0           0           0           $VFLTF(10000'.25'$io4)
405 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev                   gfdr
406 i "grainer"     .           .           .           .           $io2        .           .           .           0.1         .
    0.2         $se(-.2)    0.025
407 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
408             135         0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(6500'.25 '$f_cnc) ;cont
409 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
410 i "grainer"     .           .           .           .           $io3        .           .           .           0.1         .
    0.2         $se(-.1)    0.025
411 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
412             180         0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(5000'.25 '$f_cnv) ;cont
413 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev                   gfdr
414 i "grainer"     .           .           .           .           $io4        .           .           .           0.1         .
    0.2         $se(0)                  0.025
415 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
416             45          0           0           0           .           0           .           0           0           0
    .           .           0           0           $VFLTF(3500'.25 '$io1)
417 ;+DIVERGENTE 4 ESTRATOS (45, 75, 105, 135) CONCENTRANDO SOBRE EL
    FRENTE--------------------------------------------------------------------------
418 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev                   gfdr
419 i "grainer"     102.5       30          .1          .1          $io2        1           0.1         2.5         0.5         0.0
    0.2         -91                     2.1
420 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
421             45          10          0           0           7           0           $CLASCs     0           0           48
    0           0           0           0           $VFLTF(8454'.15'$io1)
422 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev                   gfdr
423 i "grainer"     .           .           .           .           $io3        .           .           .           0.1         .
    0.2         -91                     2.1
424 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
425             105         0           0           0           .           0           .           0           0           48
    .           .           0           0           $VFLTF(7484'.15 '$p_cnc)
426 ;           st          dur         gamp        gard        gatab       gg          ggr         gd          gdr         gst
    gstr        gfdev       gfdr
427 i "grainer"     .           .           .           .           $io4        .           .           .           0.1         .
    0.2         -91                     2.1
428 ;           gaz         gazr        gel         gelr        gdist       gdistr      gfile       tgap        tdur        tdev
    taz         tdis        tamp        tfp         fc          bw          bwt
429             135         0           0           0           .           0           .           0           0           -46
```

```
    .          .          0          0          $VFLTF(5540'.15 '$p_cnv)
;             st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev                 gfdr
i "grainer"    .          .          .          .          $io1       .          .          .          0.1        .
    0.2        -91                   2.1
;             gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
                75         0          0          0          .          0          .          0          0          -46
    .          .          0          0          $VFLTF(4500'.15 '$io2)
;+DIVERGENTE 4 ESTRATOS (45, 135, 225, 315) APERTURA HACIA TODO EL
CIRCULO-------------------------------------------------------------------------
;             st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev                 gfdr
i "grainer"    132.5      30         .1         .1         $io3       1          0.1        2.5        0.5        0.0
    0.2        -91                   4.1
;             gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
                45         10         0          0          8          0          $CLASCs    31         0          0
    0          0          0          0          $VFLTF(7622'.15'$io3)
;             st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev                 gfdr
i "grainer"    .          .          .          .          $io4       .          .          .          0.1        .
    0.2        -91                   4.1
;             gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
                135        0          0          0          .          0          .          .          0          0
    51.0333    .          0          0          $VFLTF(6929'.15 '$io2)
;             st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev      gfdr
i "grainer"    .          .          .          .          $io1       .          .          .          0.1        .
    0.2        -91                   4.1
;             gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
                225        0          0          0          .          0          .          .          0          0
    0          .          0          0          $VFLTF(6237'.15 '$io1)
;             st         dur        gamp       gard       gatab      gg         ggr        gd         gdr        gst
    gstr       gfdev                 gfdr
i "grainer"    .          .          .          .          $io2       .          .          .          0.1        .
    0.2        -91                   4.1
;             gaz        gazr       gel        gelr       gdist      gdistr     gfile      tgap       tdur       tdev
    taz        tdis       tamp       tfp        fc         bw         bwt
                315        0          0          0          .          0          .          .          0          0
    52.0333    .          0          0          $VFLTF(5959.5'.15 '$io4)
e
</CsScore>
</CsoundSynthesizer>
```

Apéndice 2

Archivo: “Instr y Macros.scd”, **Autor**: Oscar Pablo Di Liscia.

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
/*spatial macros by Pablo Di Liscia, 2014*/
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
seed           5          ;arbitrary seeding to keep the same "random" choices at each call.

;the 25 critical band edges
giSbands_edges            ftgen      0, 0, 32, -2, 0, 100, 200, 300, 400, 510, 630, 770, 920, 1080, 1270, 1480, 1720, 2000, 2320, 2700, 3150, 3700, 4400, 5300, 6400, 7700, 9500, 12000, 15500, 20000
#define        NBN                   # 25 #
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;ORC MACROS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define rev1     # "../IR/rev1.wav" #
#define rev2     # "../IR/rev2.wav" #
#define rev3     # "../IR/rev3.wav" #
#define rev4     # "../IR/rev4.wav" #

gisend                    init       .15
garev                     init       0
gipi                      init       4.*taninv(1.)
gipiS2                    init       gipi/2.
gimaxdel       init       1;(sqrt(15*15 + 15*15 + 5*5)*4)/333
giamp                     init       .53

instr 99
;this instrument makes octophonic reverb
ivol = p4
ipartitionsize =    4096             ; size of each convolution partition

aw1, aw2       pconvolve ivol*garev,$rev1, ipartitionsize, 0
aw3, aw4       pconvolve ivol*garev,$rev2, ipartitionsize, 0
aw5, aw6       pconvolve ivol*garev,$rev3, ipartitionsize, 0
aw7, aw8       pconvolve ivol*garev,$rev4, ipartitionsize, 0

aW1, aX1, aY1, aZ1 bformenc1 aw1, 90,  45
aW2, aX2, aY2, aZ2 bformenc1 aw2, 135, 45
aW3, aX3, aY3, aZ3 bformenc1 aw3, 225, 45
aW4, aX4, aY4, aZ4 bformenc1 aw4, 315, 45

aW5, aX5, aY5, aZ5 bformenc1 aw5, 90,  315
aW6, aX6, aY6, aZ6 bformenc1 aw6, 135, 315
aW7, aX7, aY7, aZ7 bformenc1 aw7, 225, 315
aW8, aX8, aY8, aZ8 bformenc1 aw8, 315, 315

aWmix= aW1+aW2+aW3+aW4+aW5+aW6+aW7+aW8
aXmix= aX1+aX2+aX3+aX4+aX5+aX6+aX7+aX8
aYmix= aY1+aY2+aY3+aY4+aY5+aY6+aY7+aY8
aZmix= aZ1+aZ2+aZ3+aZ4+aZ5+aZ6+aZ7+aZ8

;outq          aWmix, aXmix, aYmix, aZmix
    garev      =0
    endin

#define ER # 0 #

#define UHJ
```

```
#
aWre, aWim      hilbert aW
aXre, aXim      hilbert aX
aYre, aYim      hilbert aY
aWXr    =  0.0928*aXre + 0.4699*aWre
aWXiYr  =  0.2550*aXim - 0.1710*aWim + 0.3277*aYre
aleft   =  aWXr + aWXiYr
aright  =  aWXr - aWXiYr
#

/*
W = 0.5*(0.982*L + 0.982*R + j*0.164*L - j*0.164*R)
X = 0.5*(0.419*L + 0.419*R - j*0.828*L + j*0.828*R)
Y = 0.5*(0.763*L - 0.763*R + j*0.385*L + j*0.385*R)
*/
#define UHJ2BF
#
aLre, aLim      hilbert aleft
aRre, aRim      hilbert aright

aW= 0.5*(0.982*aLre + 0.982*aRre + *0.164*aLim - *0.164*aRim)
aX= 0.5*(0.419*aLre + 0.419*aRre + *0.828*aLim - *0.828*aRim)
aY= 0.5*(0.763*aLre + 0.763*aRre + *0.385*aLim - *0.385*aRim)
#

#define POL2REC(az'di'el)
#
    ia=$az
    until ia < 360 do
                ia -=360
    od
    until ia > 0 do
                ia +=360
    od
    ie                      =($el > 360 ? ($el-360) : $el)
    ie                      =($el < 0   ? (360+$el) : $el)

iazi            =(ia <= 180? (ia/180)*gipi : ((360-ia)/180)*(-gipi))
idis            = $di
iele            =(ie <= 180? (ie/180)*gipi : ((360-ie)/180)*(-gipi))
ix              =cos(iazi)*cos(iele)*idis
iy              =sin(iazi)*cos(iele)*idis
iz              =sin(iele)*idis
#


#define POL2REC_K(az'di'el)
#
checkit1:
if $az < 360 kgoto checkit2
    $az-= 360
    kgoto checkit1
checkit2:
if $az >= 0 kgoto wrapped2
    $az+= 360
    kgoto checkit2
wrapped2:

kazi=($az <= 180? ($az/180)*gipi : ((360-$az)/180)*(-gipi))
kele=($el <= 180? ($el/180)*gipi : ((360-$el)/180)*(-gipi))

kx              =cos(kazi)*cos(kele)*$di
ky              =sin(kazi)*cos(kele)*$di
kz              =sin(kele)*$di
#


#define BFDEC_Q(w'x'y'z'g)
#
;Warning: this decoder follows an ANTI-clockwise setting of loudspeakers
;   2           1
;   3           4

;front
a1=($w+$x-$y)*$g
a2=($w+$x+$y)*$g
;rear
a3=($w-$x+$y)*$g
a4=($w-$x-$y)*$g
#

#define BFDEC_CUBE(w'x'y'z'g)
#
;Warning: this decoder follows an ANTI-clockwise setting of loudspeakers
;LOWER SQUARE
;   2           1
```

```
;   3          4
;UPPER SQUARE
;   6          5
;   7          8
;;;;;;;;;;;;;;;;;;
;LOWER
;front
a1=($w+$x-$y-$z)*$g
a2=($w+$x+$y-$z)*$g
;rear
a3=($w-$x+$y-$z)*$g
a4=($w-$x-$y-$z)*$g
;;;;;;;;;;;;;;;;;;
;UPPER
;front
a5=($w+$x-$y+$z)*$g
a6=($w+$x+$y+$z)*$g
;rear
a7=($w-$x+$y+$z)*$g
a8=($w-$x-$y+$z)*$g
#

#define GA_INIT
#
ga1            init 0
ga2            init 0
ga3            init 0
ga4            init 0
ga5            init 0
ga6            init 0
ga7            init 0
ga8            init 0
garev                     init 0
#

#define GA_ASSIGN
#
ga1=ga1+a1
ga2=ga2+a2
ga3=ga3+a3
ga4=ga4+a4
ga5=ga5+a5
ga6=ga6+a6
ga7=ga7+a7
ga8=ga8+a8
#

#define GA_CLEAN
#
ga1=0
ga2=0
ga3=0
ga4=0
ga5=0
ga6=0
ga7=0
ga8=0
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
/*ATS, FM and SND grainers by Pablo Di Liscia, 2014*/
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;ATS MACROS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define        check_ATS_partials(np'op'ip'num)
#
/*here we deal with number of partials, offset and increment issues*/
$np =          (p30 < 1 ? $num : p30) ;inpars can not be <=0
$op =          (p28 < 0 ? 0 : p28) ;partial offset can not be < 0
$ip =          (p29 < 1 ? 1 : p29) ;partial increment can not be <= 0
imax           =          $op + $np*$ip ;max. partials allowed
if (imax > i_number_of_partials) then
    ;set npars to zero, so as the output will be zero and the user knows
    prints "Warning: ATS file max. number of partials (%d) reached, setting output to zero", imax      ;imax, i_number_of_partials
    $np        =          0
    $op        =          0
```

```
    $ip         =          1
endif
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define        select_atsfile(arg)
#
if $arg == 1 igoto clc
if $arg == 2 igoto clcs
if $arg == 3 igoto cla
clc:
    Sfilename  =          "../ATS-files/clarinet-C5.ats"
igoto          OK
clcs:
    Sfilename  =          "../ATS-files/clarinet-Cs2.ats"
igoto          OK
cla:
    Sfilename  =          "../ATS-files/clarinet-A3.ats"

OK:
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define        ATS_NP # 3 # ;number of Partials
#define        ATS_DU # 7 # ;duration
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define select_afile(arg)
#
if $arg == 5 then
    Sfilename = "../samples/clarinet-C5-LP.wav"
elseif $arg == 6 then
    Sfilename = "../samples/BCL-Cs2.wav"
elseif $arg == 7 then
    Sfilename = "../samples/BCL-A3.wav"
elseif $arg == 8 then
    Sfilename = "../samples/aire2.wav"
elseif $arg == 9 then
    Sfilename = "../samples/slap-Mib.wav"
elseif $arg == 10 then
    Sfilename = "../samples/keys.wav"
elseif $arg == 11 then
    Sfilename = "../samples/clC5_nois.wav"
elseif $arg == 12 then
    Sfilename = "../samples/clC5_det.wav"
elseif $arg == 13 then
    Sfilename = "../samples/single_key.wav"
elseif $arg == 14 then
    Sfilename = "../samples/keys2.wav"
endif

#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;GENERAL MACROS
#define        get_randval(val'dev'aux'var)
#
$aux           unirand    $dev
$var=$val + $aux
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define        get_birandval(val'dev'aux'var)
#
$aux           birnd      $dev
$var=$val + $aux
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define        get_tab_index(fun'indx'start'dur)
#
$indx          =( (p2-$start) / $dur)
if($fun < 0) then                   ;backwards table reading
    $indx =.999-$indx
endif
if(frac($fun)!= .1 && frac($fun)!=0) then ;fractional part (if any) times 10
    $indx *=(frac($fun)*10)         ;is read velocity
    until $indx < 1 do              ;wraparround index if out of range
    $indx -=1
od

endif
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
#define rand_sig(depth'freq'sig)
#
if($depth==0) then
    $sig=0
else
    $sig randi $depth, $freq, 2, 1, 0           ;generate a random signal
    $sig tonek $sig, 5                          ;smooth by lowpassing
endif
```

```
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;if the base value is NOT positive, then its absolute value is a table function
;number with values to be randomly selected the integral part of the next parameter
;is the number of values for the selection the mantissa is one tenth of a scaling
;value for each one of the table values
#define get_tabval(bval'rval'gpar'tbl)
#
    $tbl= abs($bval)
    itl        = int($rval)
    idv        = frac($rval)* 10
    idv        = (idv==0? 1: idv) ;avoid posssible zero value on scaler
    indx       unirand    itl-1
    $gpar      table      int(indx), $tbl
    $gpar      *=idv
#
;frequency deviation values is a special case
#define get_tabval_d(bval'rval'gpar'tbl)
#
    $tbl= abs($bval)
    itl        = int($rval)
    idv        = frac($rval)* 10
    idv        = (idv==0? 1: idv) ;avoid posssible zero value on scaler
    indx       unirand    itl-1
    $gpar      table      int(indx), $tbl
    $gpar      =(2^($gpar/12))*idv
#
;cpspch to frequency values is a special case
#define get_tabval_f(bval'rval'gpar'tbl)
#
    $tbl= abs($bval)
    itl        = int($rval)
    idv        = frac($rval)* 10
    idv        = (idv==0? 1: idv) ;avoid posssible zero value on scaler
    indx       unirand    itl-1
    $gpar      table      int(indx), $tbl
    $gpar      =cpspch($gpar)*idv
#
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;; grainer calls my_grain to make granular synthesis
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
instr grainer

    iaux=0
    ;AMPLITUDE
    $get_randval(p4'p5'iaux'iamp)
    iatab       =p6
    ;GAP
    if(p7 < 0) then
               $get_tabval(p7'p8'igg'itblg)
    else
               $get_randval(p7'p8'iaux'igg)
    endif
    ;DUR
    $get_randval(p9'p10'iaux'igd)
    ;START
    if(p11 < 0) then
               $get_tabval(p11'p12'ist'itblst)
    else
               $get_randval(p11'p12'iaux'ist)
    endif
    ;FDEV
    if(p13 < 0) then
               $get_tabval_d(p13'p14'idev'itbldev)
    else
               $get_randval(p13'p14'iaux'idev)
    endif
    ;AZIMUTH
    $get_birandval(p15'p16'iaux'iaz)
    ;ELEVATION
    $get_birandval(p17'p18'iaux'iel)
    ;DISTANCE
    $get_randval(p19'p20'iaux'id)

;AUDIO FILE
ifile          =p21       ;audio sound file

;EVOLUTION TABLES
itgap          =p22       ;gap evolution table
itdur          =p23       ;duration scaling table
itdev          =p24       ;frequency deviation table
itaz=p25       ;azimuth table
itds=p26       ;distance table
itamp          =p27       ;amplitude table
itfp=p28       ;starting file read point
;FILTER PARAMETERS
```

```
ifc =p29
ibw =p30
itbw=p31
ip2 =p2        ;p2 is needed for indexing tables but scaled by tempo
ip3 =p3        ;p3 is needed for indexing tables but scaled by tempo
;gap table must be read
if(itgap !=0) then
    ipoint     = (itgap < 0? .999 : 0)
    if(frac(itgap) > 0) then
               ipoint *= (frac(itgap)*10)
               until ipoint < 1 do
                          ipoint -=1
               od
    endif
    ifac       tablei ipoint, abs(itgap), 1, 0, 1
    igg        = igg * ifac
    itacum     =0
endif
reset:         timout  0, igg, contin
               ifile      =p21
               ;AMP
               $get_randval(p4'p5'iaux'iamp)
               ;GAP
               if(p7 < 0) then
                          $get_tabval(p7'p8'igg'itblg)
               else
                          $get_randval(p7'p8'iaux'igg)
               endif
               ;DUR
               $get_randval(p9'p10'iaux'igd)
               ;START
               if(p11 < 0) then
                          $get_tabval(p11'p12'ist'itblst)
               else
                          $get_randval(p11'p12'iaux'ist)
               endif
               ;FDEV
               if(p13 < 0) then
                          $get_tabval_d(p13'p14'idev'itbldev)
               else
                          $get_randval(p13'p14'iaux'idev)
               endif
               ;AZIMUTH
               $get_birandval(p15'p16'iaux'iaz)
               ;ELEVATION
               $get_birandval(p17'p18'iaux'iel)
               ;DISTANCE
               $get_randval(p19'p20'iaux'id)
               ;Gap evolution
               if(itgap !=0) then
                          itacum     = itacum + igg*ifac
                          ipoint  = (itgap > 0 ? itacum / ip3 : 1-itacum / ip3)
                          if(frac(itgap) > 0) then
                                     ipoint *= (frac(itgap)*10)
                                     until ipoint < 1 do
                                                ipoint -=1
                                     od
                          endif
                          ifac    tablei ipoint, abs(itgap), 1, 0, 1
                          igg        =igg * ifac
               endif
    reinit    reset
contin:        schedule "my_grain", 0, igd, iamp, iatab, ist, ifile, idev, iaz, id, iel, igd, itdur, itdev, itaz, itds,itamp, itfp, ip2, ip3, ifc, ibw, itbw
    rireturn
endin
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
instr my_grain
iamp=p4
iatab          =p5
iskip          =p6
$select_afile(p7)
idev=p8
iaz =p9
ids =p10
iel =p11
igdur          =p12
itdur          =p13
itdev          =p14
itaz=p15
```

```
itds=p16
itamp          =p17
itfp=p18

ip2 =p19
ip3 =p20

ifc =p21
ibw =p22
itbw=p23
ibwsc          =1
ilen           filelen Sfilename

;Get table deviations
if(itdur !=0) then
    $get_tab_index(itdur'indx'ip2'ip3)
    idf        tablei indx, abs(itdur), 1
    p3         =igdur*idf
endif
if(itdev !=0) then
    $get_tab_index(itdev'indx'ip2'ip3)
    iddev      tablei indx, abs(itdev), 1
    idev       =idev*iddev
endif
if(itaz !=0) then
    $get_tab_index(itaz'indx'ip2'ip3)
    iazof      tablei indx, abs(itaz), 1
    iaz        =iaz+iazof
endif
if(itds !=0) then
    $get_tab_index(itds'indx'ip2'ip3)
    idsof      tablei indx, abs(itds), 1
    ids        =ids+idsof
endif
if(itamp !=0) then
    $get_tab_index(itamp'indx'ip2'ip3)
    iasc       tablei indx, abs(itamp), 1
    iamp       =iamp*iasc
endif
if(itfp !=0) then
    $get_tab_index(itfp'indx'ip2'ip3)
    ifp        tablei indx, abs(itfp), 1, 0, 0
    iskip      =(iskip+ifp*ilen)
endif

if(ifc == 0 ) igoto bwset                                  ;no filtering
    if(ibw < 0) goto rbw
                if(itbw !=0) then
                           $get_tab_index(itbw'indx'ip2'ip3)           ;bw scaler is scanned from itbw
                           ibwsc      tablei indx, abs(itbw), 1
                           ibw        =ibw*ibwsc*ifc
                else
                           ibw        *=ifc
                endif
    goto bwset
    rbw:                                                               ;random bw scaler
                ibwsc   unirand abs(ibwsc) ;scaler is a random value between itbw and itbw+abs(ibwsc)
                ibw        =(itbw+ibwsc)*ifc
bwset:
;amp gate envelope parameters
iseg= 0.05*p3
isug= p3-iseg
    ifper      =1/p3
asig           diskin2    Sfilename, idev, iskip,0,0,8,32768          ;read audio file

if(ifc != 0) goto flta
abala = asig
goto noflta

flta:
;print ifc
aresresonx  asig, ifc, ibw, 4, 1
abala          balance ares, asig

noflta:
aenvoscili 1, ifper, iatab
ahp dcblock abala*aenv

;spatialization
p3  = p3+gimaxdel
$POL2REC(iaz'ids'iel)
ift            =99                 ;room parameters table
imode                      =3                  ;B format output
iucirc                     =1                  ;distance of the unit circle
agate                      linseg iamp,isug,iamp,iseg,0
ain            =ahp*agate
```

```
;BF ENCODING
aW,aX,aY,aZ    spat3di ain,ix,iy,iz,iucirc,ift,imode,0
;UHJ DECODING
$UHJ
outs           aleft, aright
;outq                    aW*giamp,aX*giamp,aY*giamp,aZ*giamp
endin
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;; fm_grainer calls fm_grain to make granular synthesis
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    instr fm_grainer

iaux=0
;AMPLITUDE
$get_randval(p4'p5'iaux'iamp)
iatab           =p6
;GAP
if(p7 < 0) then
    $get_tabval(p7'p8'igg'itdgap)
else
    $get_randval(p7'p8'iaux'igg)
endif
;DUR
$get_randval(p9'p10'iaux'igd)
;AZIMUTH
$get_birandval(p11'p12'iaux'iaz)
;ELEVATION
$get_birandval(p13'p14'iaux'iel)
;DISTANCE
$get_randval(p15'p16'iaux'id)

;EVOLUTION TABLES
itgap          =p17       ;gap evolution table
itdur          =p18       ;duration scaling table
itdev          =p19       ;frequency deviation table
itaz=p20       ;azimuth table
itds=p21       ;distance table
itamp          =p22       ;amplitude table

;FM PARAMETERS
if(p23 < 0) then
    $get_tabval_f(p23'p24'ifc'ipctab)
else
    $get_randval(p23'p24'iaux'ifc) ;carrier frequency
endif
;Carrier Audio Table
itcar          =p25                ;carrier table
;modulator
irat=p26                ;fc/fm ratio
ifm =ifc*irat  ;fm freq
;Modulation Index Parameters
$get_randval(p27'p28'iaux'indsc)
iiof=p29                ;offset for mod. indx. table
itndx          =p30                ;table for mod. indx.
itiev          =p31                ;table for index evolution along the grain queue
irad=p32                ;random modulation parameters AM, FM, IM
iraf    =p33
irfd=p34
irff=p35
imrd=p36
imrf=p37
ip2 =p2        ;p2 is needed for indexing tables
ip3 =p3        ;p3 is needed for indexing tables

;gap table must be read
if(itgap !=0) then
    ipoint     = (itgap < 0? .999 : 0)
    ifac       tablei ipoint, abs(itgap), 1, 0, 1
    igg        = igg * ifac
    itacum     =0
endif

reset:         timout     0, igg, contin
    ip3        =p3
    iaux       =0
    ;AMPLITUDE
    $get_randval(p4'p5'iaux'iamp)
    iatab       =p6
    ;GAP
    if(p7 < 0) then
                $get_tabval(p7'p8'igg'itdgap)
    else
                $get_randval(p7'p8'iaux'igg)
    endif
    ;DUR
    $get_randval(p9'p10'iaux'igd)
```

```
    ;AZIMUTH
    $get_randval(p11'p12'iaux'iaz)
    ;ELEVATION
    $get_randval(p13'p14'iaux'iel)
    ;DISTANCE
    $get_randval(p15'p16'iaux'id)
    ;Gap evolution
    if(itgap !=0) then
               ipoint     = (itgap < 0? .999 : 0)
               ifac       tablei ipoint, abs(itgap), 1, 0, 1
               igg        = igg * ifac
               itacum     =0
    endif
    if(p23 < 0) then
               $get_tabval_f(p23'p24'ifc'ipctab)
    else
               $get_randval(p23'p24'iaux'ifc) ;carrier frequency
    endif
    ifm        =ifc*irat                     ;fm freq
    $get_randval(p27'p28'iaux'indsc)         ;IM
reinit         reset
;first time call                          2    3    4    5     6     7   8    9     10     11     12    13    14    15    16    17     18     19    20   21     22     23    24    25     26   27    28   29
contin: schedule          "fm_grain", 0, igd, igd, iamp,iatab, iaz, iel, id, itdur, itdev, itaz, itds, itamp, ifc, itcar,  ifm, indsc, iiof, itndx, itiev, irad,  iraf, irfd, irff, imrd, imrf, ip2, ip3
    rireturn

    endin
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    instr fm_grain ;makes the grain using FM Synthesis
;grain parameters
igdur          =p4
iamp=p5
iatab          =p6
iaz =p7
iel =p8
ids =p9
;evolution tables
itdur          =p10
itdev          =p11
itaz=p12
itds=p13
itamp          =p14
;FM parameters
ifc =p15
itcar          =p16
ifm =p17
indsc          =p18
iiof=p19
itndx          =p20
itiev          =p21
;random modulation parameters
irad=p22*iamp;AM
iraf    =p23
iamp-=irad
irfd=p24    ;FM
irff=p25
irid=p26       ;IM
irif=p27
;duration parameters
ip2 =p28
ip3 =p29
;Get table deviations
if(itdur !=0) then
    $get_tab_index(itdur'indx'ip2'ip3)
    idf        tablei     indx, abs(itdur), 1
    p3         =igdur*idf
endif
if(itdev !=0) then
    $get_tab_index(itdev'indx'ip2'ip3)
    iddev      tablei     indx, abs(itdev), 1
    ifc        *=iddev
    ifm        *=iddev
endif
if(itaz !=0) then
    $get_tab_index(itaz'indx'ip2'ip3)
    iazof      tablei     indx, abs(itaz), 1
    iaz        +=iazof
endif
if(itds !=0) then
    $get_tab_index(itds'indx'ip2'ip3)
    idsof      tablei     indx, abs(itds), 1
    ids        +=idsof
endif
if(itamp !=0) then
    $get_tab_index(itamp'indx'ip2'ip3)
```

```
    iasc       tablei     indx, abs(itamp), 1
    iamp       *=iasc
endif
if(itiev !=0) then
$get_tab_index(itiev'indx'ip2'ip3)
    iisc       tablei     indx, abs(itiev), 1
    indsc      *=iisc
endif
;amp gate envelope parameters
iseg           =          0.05*p3
isug           =          p3-iseg
idur           =          p3
;for the grain amplitude oscillator
ifper          =          1/p3
;random data generation
$rand_sig(irad'iraf'kar)
kar  = abs(kar)
$rand_sig(irfd'irff'kfr)
kfrc =kfr*ifc
kfrm =kfr*ifm
$rand_sig(irid'irif'kir)
;synthesis
afcenv         oscili     iamp+kar, ifper, iatab           ;carrier envelope
kindx          oscili     indsc+kir, ifper, itndx          ;modulator envelope
kindx          += iiof
agate          linseg     1,isug,1,iseg,0
asigfoscili    afcenv*agate, 1, ifc+kfrc, ifm+kfrm, kindx, itcar
ain dcblock asig

;spatialization;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;enlarge p3 by the max. eco length
p3  =          p3+gimaxdel
$POL2REC(iaz'ids'iel)
ift            =99                   ;room parameters table
imode                     =3                    ;B format output
iucirc                    =1                    ;distance of the unit circle
;BF ENCODING
aW,aX,aY,aZ    spat3di    ain,ix,iy,iz,iucirc,ift,imode,0
;$UHJ
;outs                     aleft, aright
outq           aW*giamp,aX*giamp,aY*giamp,aZ*giamp
    endin
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;; ats_grainer calls ats_grain to make spectral granular synthesis
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    instr ats_grainer

iaux=0
;AMPLITUDE
$get_randval(p4'p5'iaux'iamp)
iatab           =p6
;GAP
if(p7 < 0) then
    $get_tabval(p7'p8'igg'itblg)
else
    $get_randval(p7'p8'iaux'igg)
endif
;DUR
$get_randval(p9'p10'iaux'igd)
;FDEV
if(p13 < 0) then
    $get_tabval_d(p13'p14'idev'itbldev)
else
    $get_randval(p13'p14'iaux'idev)
endif
;AZIMUTH
$get_birandval(p15'p16'iaux'iaz)
;ELEVATION
$get_birandval(p17'p18'iaux'iel)
;DISTANCE
$get_randval(p19'p20'iaux'id)
;ATS FILE
ifile          =p21                  ;ATS file name
$select_atsfile(p21)
i_number_of_partials      ATSinfo    Sfilename, $ATS_NP
i_duration     ATSinfo    Sfilename, $ATS_DU
;TIME POINTERS ON THE ATS FILE
ifrom =(p11 > 0 && p11 < i_duration? p11 : 0)
ito =          ifrom+(i_duration-ifrom)*rnd(p12)

;EVOLUTION TABLES
itgap          =p22       ;gap evolution table
itdur          =p23       ;duration scaling table
itdev          =p24       ;frequency deviation table
itaz=p25       ;azimuth table
itds=p26       ;distance table
```

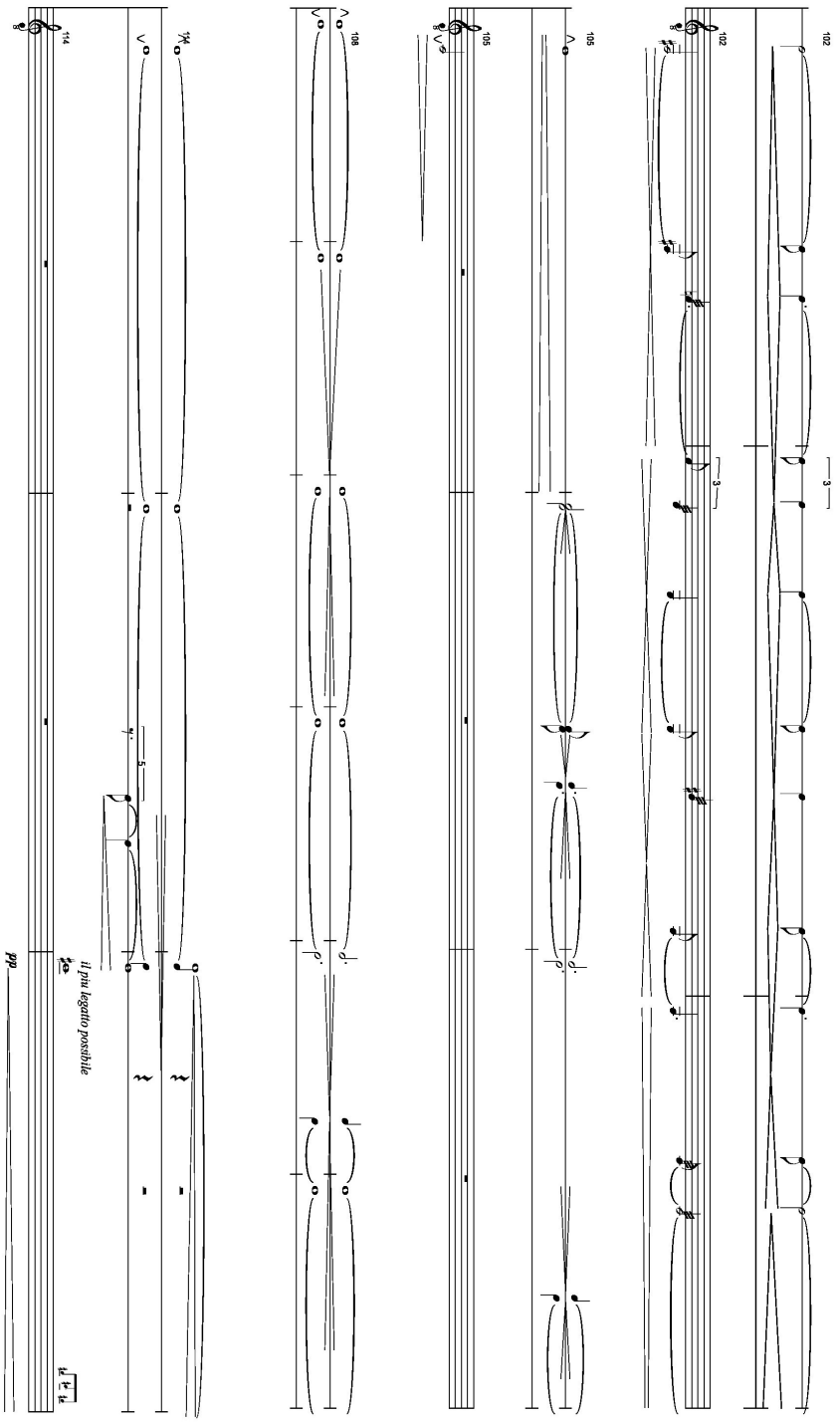
```
itamp          =p27       ;amplitude table

;ATS PARAMETERS
iop =p28       ;starting partial
isp =p29       ;step for partials
inp =p30       ;number of partials
inamp          =p31       ;noise level
intab          =p32       ;amp evolution table for noise

;here we compute the lower partial frequency
;based on the pitch of the ATS analysis
;Warning: this work only for PITCHED sounds!
;if the sound is not pitched, then put the lower
;frequency desired as a negative number.
ipitch         =p33       ;ATS sound pitch
indx    =0     ;initialize index

ilowf          =(ipitch >= 0? ipitch*iop*idev : abs(ipitch))
;here we compute the lower noise band to be synthesized
;based on the lower partial requested.
until indx == $NBN do
    ifnb1  table indx, giSbands_edges
    ifnb2  table indx+1, giSbands_edges
    if(ilowf >= ifnb1 && ilowf < ifnb2) goto band_found
    indx       +=         1 ;increment index
od
band_found:
ion =indx                 ;starting noise band
isn =1                    ;step for noise bands
inn =$NBN-indx ;number of noise bands

$check_ATS_partials(inp'iop'isp'i_number_of_partials)
ip2 =p2        ;p2 is needed for indexing tables
ip3 =p3        ;p3 is needed for indexing tables

;gap table must be read
if(itgap !=0) then
    ipoint     = (itgap < 0? .999 : 0)
    ifac       tablei ipoint, abs(itgap), 1, 0, 1
    igg        = igg * ifac
    itacum     =0
endif

reset:         timout     0, igg, contin
    ip3 =p3
    ifile      =p21
    ;AMP
    $get_randval(p4'p5'iaux'iamp)
    ;GAP
    if(p7 < 0) then
               $get_tabval(p7'p8'igg'itblg)
    else
               $get_randval(p7'p8'iaux'igg)
    endif
    ;DUR
    $get_randval(p9'p10'iaux'igd)
    ;START
    ;TIME POINTERS ON THE ATS FILE
    ifrom =(p11 > 0 && p11 < i_duration? p11 : 0)
    ito        =          ifrom+(i_duration-ifrom)*rnd(p12)
    ;FDEV
    if(p13 < 0) then
               $get_tabval_d(p13'p14'idev'itbldev)
    else
               $get_randval(p13'p14'iaux'idev)
    endif
    ;AZIMUTH
    $get_birandval(p15'p16'iaux'iaz)
    ;ELEVATION
    $get_birandval(p17'p18'iaux'iel)
    ;DISTANCE
    $get_randval(p19'p20'iaux'id)
    ;Gap evolution
    if(itgap !=0) then
                           itacum     = itacum + igg*ifac
                           ipoint  = (itgap > 0 ? itacum / ip3 : 1-itacum / ip3)
                           ifac    tablei ipoint, abs(itgap), 1, 0, 1
                           igg        =igg * ifac
    endif
reinit         reset

contin: ;schedule         "ats_grain", 0, igd, iamp, iatab, ifrom, ito, ifile, idev, iaz, id, iel, igd, itdur, itdev,
itaz, itds ,itamp, ip2, ip3, iop, isp, inp, inamp, intab, ion, isn, inn, abs(ilowf)
    event_i "i", "ats_grain", 0, igd, iamp, iatab, ifrom, ito, ifile, idev, iaz, id, iel, igd, itdur, itdev, itaz, itds
,itamp, ip2, ip3, iop, isp, inp, inamp, intab, ion, isn, inn, abs(ilowf)

```

```
    rireturn

    endin
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    instr ats_grain
asigd          init       0
asign          init       0

iamp=p4
iatab          =p5
ifrom          =p6
ito =p7
$select_atsfile(p8)
idev=p9
iaz =p10
ids =p11
iel =p12
igdur          =p13

itdur          =p14
itdev          =p15
itaz=p16
itds=p17
itamp          =p18

ip2 =p19
ip3 =p20

iof =p21
isp =p22
inp =p23
inamp          =p24
intab          =p25
ion =p26
isn =p27
inn =p28
ilowf          =p29
ifn =5         ;sine wave table
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Get table deviations
if(itdur !=0) then
    $get_tab_index(itdur'indx'ip2'ip3)
    idf        tablei     indx, abs(itdur), 1, 0, 1
    p3         =igdur*idf
endif
if(itdev !=0) then
    $get_tab_index(itdev'indx'ip2'ip3)
    iddev      tablei     indx, abs(itdev), 1, 0, 1
    idev       =idev*iddev
endif
if(itaz !=0) then
    $get_tab_index(itaz'indx'ip2'ip3)
    iazof      tablei     indx, abs(itaz), 1, 0, 1
    iaz        =iaz+iazof
endif
if(itds !=0) then
    $get_tab_index(itds'indx'ip2'ip3)
    idsof      tablei     indx, abs(itds), 1, 0, 1
    ids        =ids+idsof
endif
if(itamp !=0) then
    $get_tab_index(itamp'indx'ip2'ip3)
    iasc       tablei     indx, abs(itamp), 1, 0, 1
    iamp       =iamp*iasc
endif
if(intab !=0) then
    $get_tab_index(intab'indx'ip2'ip3)
    inoisc     tablei     indx, abs(intab), 1, 0, 1
    inamp      =inamp*inoisc
endif
;amp gate envelope parameters;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
iseg           =          0.05*p3
isug           =          p3-iseg
idur           =          p3
ifper          =          1/p3
;synthesis;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;synthesize ATS file splitting into deterministic and residual
ktime          linseg ifrom, idur, ito ;time envelope
if(inamp == 0) goto nonoi
    anoi ATSaddnz ktime, Sfilename, inn, ion, isn
    anoi butterhp anoi, ilowf
    goto noiset
nonoi:
    anoi=0
noiset:
;;;;;;;;;;;;;;;;;;;;;;;;
```

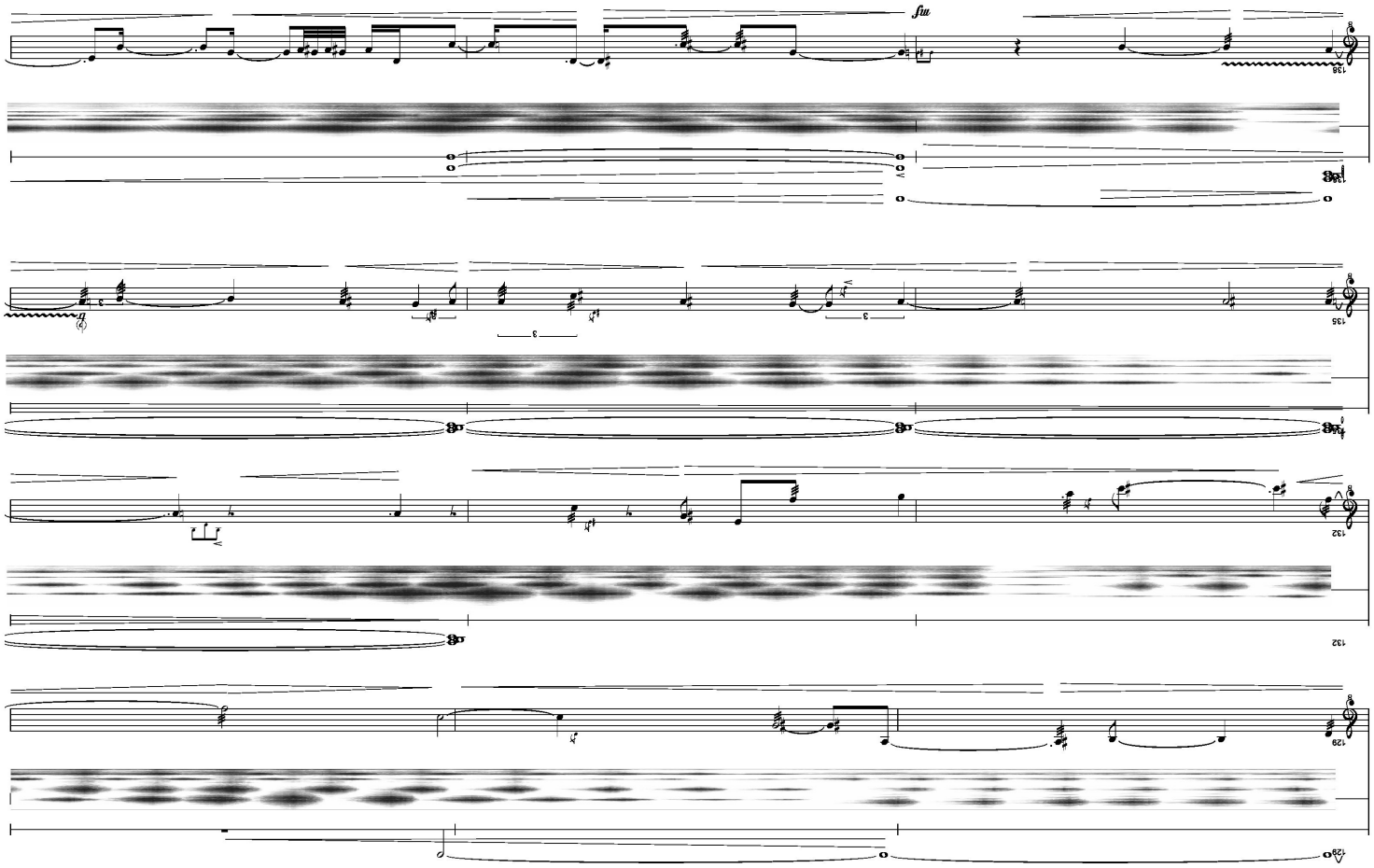
```
if(iamp == 0) goto nodet
    adet ATSadd ktime, idev, Sfilename, ifn, inp, iof, isp, 0
    goto detset
nodet:
    adet=0
detset:
;;;;;;;;;;;;;;;;;;;;;;;;
if(iatab != 0) goto env
aenv=1
goto envset
env:
    aenv oscili 1, ifper, iatab
envset:
;spatialization;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
p3  =          p3+gimaxdel
$POL2REC(iaz'ids'iel)
ift            = 99                 ;room parameters table
imode                     = 3                  ;B format output
iucirc                    = 1                  ;distance of the unit circle
agate                     linseg 1,isug,1,iseg,0
ain            = (adet*iamp+anoi*inamp)*agate*aenv
;BF ENCODING
aW,aX,aY,aZ    spat3di    ain,ix,iy,iz,iucirc,ift,imode,0
;$UHJ
;outs                     aleft, aright
outq           aW*giamp,aX*giamp,aY*giamp,aZ*giamp

endin
```

Apéndice 3

Archivo: Partitura de “el aire y la reja”, páginas 7 a 10, **Autor**: Oscar Pablo Di Liscia.



-8-
113
sfm
120
126


fm