Problema 19.

¿Cuál es el problema que el artículo afirma resolver?

El artículo elegido es *Haskore music notation - An algebra of Music, Journal of Functional Programming*. Se desarrolló un enfoque algebraico a la descripción y composición de la música usando Haskore. Según el artículo, se usan operaciones como transpose y tempo-scaling para transformar “objetos musicales” como las notas musicales para formar ideas más complejas como composiciones secuenciales y concurrentes. La interpretación de las estructuras musicales está definida en términos de una noción de rendimiento, lo que permite que colectivamente formen el álgebra de la música. El diseño de las estructuras musicales en Haskore está hecho para intensificar la simetría y aplicabilidad de los métodos algebraicos.

El punto principal del artículo y ¿Qué aprendió usted de él?

El artículo describe la manera en que se crearon las estructuras musicales ya sean notas, acordes, entre otros, con datatypes y funciones en Haskell para darle paso a Haskore. También probaron muchas propiedades conmutativas, asociativas y distributivas de los operadores. y así es como nace el álgebra de la música. Las interpretaciones o actuaciones musicales era algo que tenían miedo de implementar por el estigma asociado, pero igualmente querían realizar lo que es la música computarizada, desarrollando incluso casos que solo pueden ser tocados por computadoras.

Me parece muy interesante cómo las relaciones musicales pueden ser expresadas en un lenguaje como Haskell. Siempre se ha mencionado teóricamente que la música es un algoritmo, pero no se había visto la solución o demostración de esto, entonces el desarrollo de este tema, el álgebra de la música resulta muy interesante. Aprendí de ideas como la relación entre las expresiones musicales y los lenguajes de computación, aparte de su relación con las estructuras matemáticas y

cómo los compositores deliberadamente explotaban esta relación en sus composiciones musicales. Aún más importante, me surgió la idea de cómo se pueden formar relaciones entre ideas abstractas con los lenguajes de computación que aún no han sido hechas como las composiciones literarias, deben de tener algún patrón gramatical. Además de cómo en un lenguaje como Haskell se pueden crear entidades para representar ideas del mundo real.

¿Qué contribución hizo vs cualquier trabajo relacionado que se mencione en el artículo?

El artículo se enfoca en dos cuestiones: la descripción y la interpretación de las estructuras musicales. Se demuestra la estructura algebraica de la música y abre el paso para desarrollar composiciones algebraicas más a fondo. Se han hecho varios enfoques parecidos a los del artículo como variantes en Lisp. Se menciona el lenguaje Fugue de Dannenberg, en donde los operadores son similares a los del artículo, pero se enfoca más en la síntesis de instrumentos en vez de composiciones orientadas a notas musicales. También se menciona la geometría funcional de Henderson, en el cual se utiliza un lenguaje funcional para generar gráficos computarizados.

Referencias bibliográficas

Paul Hudak, Tom Makucevich, Syam Gadde, and Bo Whong, *"Haskore Music Notation -- An Algebra of Music"*, Journal of Functional Programming, 6*(3),* Mayo 1996, pp. 465-483.

Problema 20. Responda a la pregunta: ¿Cómo es que la evaluación perezosa ayuda a escribir DSLs en haskell? de un ejemplo.

En el artículo mencionado anteriormente se mencionan muchos trabajos hechos ya sea por Paul Hudak u otros, en los cuales se usan lenguajes DSL para realizar música computarizada, ya sea Haskore o MDL. En las otras investigaciones realizadas se usó Lisp, mientras que en el caso actual fue Haskell. En un lenguaje no-estricto como Haskell, los autores mencionan que la evaluación perezosa está implementada de gratis, cuando no es el caso con Lisp. La evaluación perezosa les permite un mejor rendimiento sin sacrificar la semántica. Usar un lenguaje como Haskell ofrece beneficios como (1) incrementar el rendimiento evitando cálculos innecesarios y tratando condiciones de error al evaluar expresiones compuestas, (2) construir estructuras de datos potencialmente infinitas, y (3) definir estructuras de control como abstracciones, en lugar de operaciones primitivas. En lo personal, al ser los lenguajes DSL soluciones a un dominio específico, problema particular o un área determinada, yo diría que la ventaja principal es poder representar conceptos abstractos como entidades en el lenguaje, y de esta manera obtener los beneficios asociados a estas como lo son el rendimiento.