Demo: Counterpoint by Construction

Youyou Cong
Tokyo Institute of Technology
Tokyo, Japan
cong@c.titech.ac.jp

John Leo
Halfaya Research
Bellevue, WA, USA
leo@halfaya.org

Western music of the common practice period tends to loosely follow sets of rules, which were developed over time to ensure the aethetic quality of the composition. Among various rules, those for harmony [Piston et al. 1987] and counterpoint [Fux and Mann 1965] (a technique for generating multi-voice melodies) are continue to be taught to music students, not only as a means to understand the music of that period, but also as a foundation for modern art and popular music.

To help analyze and synthesize tonal music, it is worthwhile to encode these rules into a programming language, and as shown by recent studies, functional programming languages are particularly suited to this task. In the past decade, Haskell has been extensively used to encode the rules of harmony [De Haas et al. 2011, 2013; Koops et al. 2013; Magalhães and de Haas 2011; Magalhães and Koops 2014] as well as counterpoint [Szamozvancev and Gale 2017]. Since Haskell is a statically typed language, these frameworks often come with a form of type safety property, namely "well-typed music does not sound wrong". Unfortunately, it turns out that the type system of plain Haskell is sometimes not powerful enough to guarantee this property. To avoid generating music that "sounds wrong", previous studies have incorporated various language extensions, such as GADTs [Cheney and Hinze 2002] and singleton types [Eisenberg and Weirich 2013], to enable dependently-typed programming.

In this demonstration of work in progress, we present Music Tools [Cong and Leo 2019], a library of small tools that can be combined functionally to help analyze and synthesize music. To allow simple and natural encoding of rules, we built the library in Agda [Norell 2007], which is a functional language with full dependent types. As an application of the library, we demonstrate an implementation of species counterpoint, based on the rules given by Fux and Mann [1965]. Thanks to Agda's rich type system, we can ensure by construction that well-typed counterpoint satisfies all the required rules. We show how the type-based approach both aids human composition and allows for computer-generated creation of correct counterpoint. We also contrast our work to a recent study on generating natural-sounding counterpoint by machine learning [Huang et al. 2017], which does not give us correctness guarantees. Finally, we discuss further applications of our library, including handling of functional harmony.

Conference'17, July 2017, Washington, DC, USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

Acknowledgments

The authors would like to thank the participants of the Tokyo Agda Implmentors' Meeting, especially Ulf Norell and Jesper Cockx, for many helpful suggestions that improved our Agda code.

References

James Cheney and Ralf Hinze. 2002. A lightweight implementation of generics and dynamics. In *Proceedings of the 2002 ACM SIGPLAN workshop on Haskell*. ACM, 90–104.

Youyou Cong and John Leo. 2019. Music Tools. https://github.com/halfaya/MusicTools.

W. Bas De Haas, José Pedro Magalhães, Remco C. Veltkamp, and Frans Wiering. 2011. HarmTrace: Improving Harmonic Similarity Estimation Using Functional Harmony Analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR '11)*. 67–72.

W. Bas De Haas, José Pedro Magalhães, Frans Wiering, and Remco C. Veltkamp. 2013. HarmTrace: Automatic Functional Harmonic Analysis. *Computer Music Journal* 37:4 (2013), 37–53. https://doi.org/10.1162/COMJ_a_00209

Richard A Eisenberg and Stephanie Weirich. 2013. Dependently typed programming with singletons. *ACM SIGPLAN Notices* 47, 12 (2013), 117–130.

J.J. Fux and A. Mann. 1965. *Study of Counterpoint*. W. W. Norton, Incorporated.

Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. 2017. Counterpoint by Convolution. In *Proceedings of ISMIR 2017*. https://ismir2017.smcnus.org/wp-content/uploads/2017/10/187_Paper.pdf

Hendrik Vincent Koops, José Pedro Magalhães, and W. Bas De Haas. 2013. A Functional Approach to Automatic Melody Harmonisation. In *Proceedings of the First ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design (FARM '13)*. ACM, 47–58. https://doi.org/10.1145/2505341.2505343

José Pedro Magalhães and W. Bas de Haas. 2011. Functional modelling of musical harmony: an experience report. In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP '11)*. ACM, New York, NY, USA, 156–162.

José Pedro Magalhães and Hendrik Vincent Koops. 2014. Functional Generation of Harmony and Melody. In *Proceedings of the Second ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design (FARM '14)*. ACM. https://doi.org/10.1145/2633638.2633645

Ulf Norell. 2007. *Towards a practical programming language based on dependent type theory*. Ph.D. Dissertation. Chalmers University of Technology.

W. Piston, M. DeVoto, and A. Jannery. 1987. *Harmony*. Norton.

Dmitrij Szamozvancev and Michael B Gale. 2017. Well-typed music does not sound wrong (experience report). In *ACM SIGPLAN Notices*, Vol. 52. ACM, 99–104.