

Clone Detection for Max/MSP Patch Libraries

Nicolas Gold¹, Jens Krinke¹, Mark Harman¹, David Binkley²

¹King's College London, CREST, Dept of Computer Science
{nicolas.gold, jens.krinke, mark.harman}@kcl.ac.uk

²Computer Science Department, Loyola College, Maryland
binkley@cs.loyola.edu

Poster Abstract

Determining music similarity is a well-known problem in the music information retrieval community. Of the music representations previously addressed by such work, patches written in visual languages such as Pure Data [1] and Max/MSP [2] are not usually considered, yet they represent a class of music representation where the generative elements can have a close and often deterministic relationship with the produced sound. As such they are potentially good candidates for discovering similarity in the final audio through analysis of the source language (similar discovery techniques may also apply to packages like Supercollider and CSound although these are textual). Such a technique could have applications not only in music information retrieval but also in live coding and education.

This poster presents work in progress on detecting similarities in patches and sub-patches written in Max/MSP and grouping these, allowing collections to be formed and navigated. The technique we propose is based on clone detection, a well-known approach in software engineering to finding similar and identical pieces of source code within large software systems (see [3-5]). Dataflow languages such as Max/MSP present particular challenges to existing clone detection technology because of the absence of explicit control flow in the source language. Recent work has addressed this problem for Simulink models [6, 7] but these algorithms are unsuitable for application directly to Max/MSP. Control flow in Max/MSP is dependent on the spatial relationships of the objects used in a patch, thus graph isomorphic approaches such as [6] cannot be applied without prior transformation of the source patch.

Our work proposes initially to adapt the Operation Control Graph approach of Karam et al. [8] to make control flow explicit with respect to the semantics of the execution engine and subsequently to apply the graph isomorphism algorithms of Pham et al. [6]. In addition to the need to address implicit control flow, the nature of patch libraries (a large number of relatively small programs) is likely to be different to large software systems (a few large, but highly interconnected, pieces of source code) and this will also need to be addressed in the future. We also aim to extend this work to more general forms of clone detection for non-traditional programming languages and paradigms.

References

- 1 <http://puredata.info>
- 2 <http://www.cycling74.com>
- 3 Krinke, J.: 'Identifying Similar Code with Program Dependence Graphs'. Proc. Eighth Working Conference on Reverse Engineering, Stuttgart, Germany, 2001
- 4 Jia, Y., Binkley, D., Harman, M., Krinke, J., and Matsushita, M.: 'KClone: A Proposed Approach to Fast Precise Clone Detection'. Proc. 3rd International Workshop on Software Clones, Kaiserslautern, Germany, 24th March 2009.
- 5 Stefan, B., Koschke, R., Antoniol, G., Krinke, J., and Merlo, E.: 'Comparison and Evaluation of Clone Detection Tools', IEEE Trans. Softw. Eng., 2007, 33, (9), pp. 577-591.
- 6 Pham, N.M., Nguyen, H.A., Nguyen, T.T., Al-Kofahi, J.M., and Nguyen, T.N.: 'Complete and Accurate Clone Detection in Graph-based Models'. Proc. IEEE International Conference on Software Engineering, Vancouver, Canada, May 2009.
- 7 Deissenboeck, F., Hummel, B., Juergens, E., Schatz, B., Wagner, S., Girard, J.-F., and Teuchert, S.: 'Clone detection in automotive model-based development'. Proc. Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, 2008.
- 8 Karam, M.R., Smedley, T.J., and Dascalu, S.M.: 'Unit-Level Test Adequacy Criteria for Visual Dataflow Languages and a Testing Methodology', ACM Transactions on Software Engineering and Methodology, 2008, 18, (1), pp. 1-40