Clone Detection for Max/MSP Patch Libraries
Nicolas Gold\textsuperscript{1}, Jens Krinke\textsuperscript{1}, Mark Harman\textsuperscript{1}, David Binkley\textsuperscript{2}

\textsuperscript{1}King’s College London, CREST, Dept of Computer Science (nicolas.gold, jens.krinke, mark.harman)@kcl.ac.uk
\textsuperscript{2}Computer Science Department, Loyola College, Maryland. binkley@cs.loyola.edu

1. The Problem

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 \cite{1} and Max/MSP \cite{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.

2. Illustration

To illustrate the problem, consider the simple Max patch (a). The intention is to carry out the calculation 12/4 and print the result when a “bang” message is sent from the button at the top. Patch (a) achieves the intended result but (b) does not (see the highlighted result objects). The only difference between the patches is the layout of the objects. The connections are the same and thus the two patches are graph-isomorphic. The difference in the result arises because of the order in which Max/MSP sends messages (right-left, top-bottom). Approaches that use graph isomorphism to identify clones and near-clones (e.g. Pham et al. \cite{6}) would thus identify these patches as identical when they are not.

3. Proposed Solution

We are working on a method for 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 \cite{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 \cite{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 \cite{6} cannot be applied without prior transformation of the source patch.

4. Example

\textbf{(a)}

\textbf{Number objects}

\textbf{(b)}

\textbf{Number objects}

\textbf{Create DCG w.r.t. Max/MSP execution semantics}

\textbf{Create DCG w.r.t. Max/MSP execution semantics}

\textbf{Identify Clones}

\textbf{(a)}

\textbf{(b)}

\textbf{Figures}

\textbf{References}

1. http://puredata.info
2. http://www.cycling74.com