PPP – a Framework for Algorithmic Composition

Thomas Neuhaus
Institut für Computermusic und elektronische Medien, Folkwang-Rochschule Essen
neuhaus@folkwang.uni-essen.de
Phone (40)-201-4903170

Abstract

PPP is a language designed as a framework for the evaluation and testing of compositional algorithms. It is based on a table-oriented model of abstract musical scores, the output can thus easily be converted into formats like Csound-scordilles or MIDI-standardfiles. It is easily expandable on the source-code level (C++) through the use of simple parser tables. This approach implies the need to distinguish between the implementation-language and the application-language for a given compositional algorithm. On the other hand it helps the programmer to concentrate on the design of compositional algorithms without having to bother with more complicated programming issues e.g. scanning and parsing of input text.

The Score Model

A score in PPP is regarded as a collection of sequences, each starting at a virtual time of 0. A sequence is a time-ordered collection of events of similar inner structure. Each event consists of a time field indicating the entrydelay to its successor and an arbitrary number of parameterfields describing the behaviour of these events. Events in the same sequence have the same underlying structure, i.e. the same number of parameterfields, the position of which indicating similar semantics.

A PPP input file describes

1. the structure of the events in each sequence
2. how the events of these sequences get their exact values

Several such descriptions may exist which result in different sequences. The first part of such a description consists of a declaration of parameters and their (symbolic) values which the corresponding parameterfields of the resulting events may have. The second part references a sequence of algorithms which are to be carried out in order to actually generate the events.

Parameter- vs. Eventoriented Algorithms

Algorithms supported by PPP can broadly be divided into two classes. One class regards the succession of values in a single parameter, independent of whatever happens in the other parameters as the main compositional focus. This approach resemble the polyphony of parameters in classical serialization (the algorithms themselves need not be of serialistic nature though). Examples of this class are permutations, tendency-marks, weighted distributions of values and the like.

The other class of algorithms regards the whole event (or subsequences thereof) as the main entity and thus always generates complete and selfsustant events. Examples are sequencing, markov-states, vector manipulations and the like.
Both approaches are supported in PPP, although the different nature of these classes of algorithms does not permit them to be used in parallel. In both classes it is possible, to refer to previously generated values or events and use them as arguments for subsequent algorithms.

The Parser Interface

The language PPP itself does not permit a composer to formulate algorithms of his own. It was designed to simplify the use and to evaluate the usefulness of algorithms in conjunction or combination with others. The inclusion of self-written algorithms however is supported and encouraged on the source-code level. Great care has been taken in the design of the language-parser which is completely table-driven. The table, that denotes all available algorithms has been even more simplified, so that the addition of a simple datastructure into this table (and a recompilation of course) is enough to integrate any user-defined algorithm into PPP. (Space is too short here to give a more detailed description of this interface)

As PPP is available in source code and expected to run on several architectures, the author expects the number of algorithms to grow over time. Eventually PPP will then appear as a big toolbox of compositional algorithms.

Converters

The general approach of the score-model (and that the output) of PPP makes it necessary to provide versatile and flexible converter programs that can transform such an output into various formats recognized by standard tools which in turn use them to produce sound, MIDI-data, common western music notation or other representations of the intended music.

At the time being, two such converters exist, one produces standard midifiles, the other produces Csound scores. These converter programs share a lot of code and are written in quite a modular way so that extending them or adding new backends for other output formats should not be too difficult (though not as simple as adding algorithms to PPP due to the peculiarities of the different output formats).

Educational Issues

PPP was developed at the Institut für Computermusik und elektronische Medien at the Folkwang-Hochschule in Germany. This institute offers a program for the education of composers, many of whom have no prior knowledge of computer science or programming. It is the intent of the institute to supply these students with at least some basic knowledge of these topics. But trying to make practical use of some basic programming techniques to compose music is hardly possible and often leads to frustration on the students’ side.

Within the framework of PPP however, the students may concentrate on the development of the algorithms themselves without having to bother about the environment into which they should be embedded. This enables the students to directly evaluate and incorporate their algorithms in concrete compositional projects.

Final Remarks

By the time of this ICMBE 1995 and the above mentioned converter programs will be freely available via anonymous ftp (URL: ftp://ftp.folkwang uni-muen de/pub/mac or /pub/unix) under terms and conditions of the General Public License of the Free Software Foundation.

The author like to thank everyone who made this development possible, especially Prof. Dirk Reith for his critical advice and helpful hints.