Background Report: ABCD - ABC Music Notation extendeD with programming features*

Trent Jeffery
The University of British Columbia
Vancouver, B.C.
y3f0b@ugrad.cs.ubc.ca

Ziyang Jin
The University of British Columbia
Vancouver, B.C.
f4a0b@ugrad.cs.ubc.ca

Victoria Wang
The University of British Columbia
Vancouver, B.C.
y7x9a@ugrad.cs.ubc.ca

Yifan Yang
The University of British Columbia
Vancouver, B.C.
p7w9a@ugrad.cs.ubc.ca

ABSTRACT

This is our project background research report for our new Domain Specific Language - ABCD, the extended language that adds programming features to the classic ABC notation. This report gives an overview of ABCD, its value, and how it is related to ABC. We also compare similiar music programming languages, discuss their design decisions. In the end, we describe a blueprint of ABCD and offer a few examples to showcase language implementatiopn.

KEYWORDS

Programming Language, Music, ABC Music Notation, Alda, Chuck, Overtone, EBNF, Parsing, Compiler

ACM Reference Format:
Trent Jeffery, Ziyang Jin, Victoria Wang, and Yifan Yang. 2017. Background Report: ABCD - ABC Music Notation extendeD with programming features. In *Proceedings of The University of British Columbia (CPSC 311 2017W1).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 OVERVIEW

aaa

2 VALUE/IMPORTANCE/IMPACT

aaa

3 SIMILAR WORK

aaa

3.1 Alda

aaa

3.2 Chuck

3.2.1 What is it about? Chuck is a powerful programming language designed for real-time sound synthesis and music creation[9]. It has the following great features[8]:

- Strongly-timed: Chuck allows programmers to control and reason about time precisely and expressively. It is an extremely important feature in audio programming paradigm.
- Deterministic concurrency model: Chuck is able to run multiple processes (shreds) in parallel in its virtual machine. Combined with time-based model, it empowers end-user programmers to have arbitrarily fine granularity and create complex audio synthesis programs.
- Dynamic control rates: Programmers can manipulate the control rates (the manner with which a shred advances its way through time[8]) as they want and the control rates can dynamically vary with time.
- Live-coding: Chuck allows programmers to modify the code on the fly without recompiling.

3.2.2 What has it achieved?

Language Design: Chuck is first released in 2003[5]. Since then, it has evolved into a very robust and complete audio programming language. It is static typing. Besides, the designer of Chuck has includes most of the modern language features in this languages. For example, it has primitive types such as int, float, void etc, reference types, variables, arrays, functions, control structures, and even object oriented features. Furthermore, since Chuck is a programming language specialized in music, it contains many special designs such as ‘time‘ and ‘dur‘ type for users to easily manipulate time, ‘unit generator‘ and ‘unit analyzer‘ functions that are convenient to modify big chunk of music.

Applications: The Chuck has found a variety of applications in music research and music creation. First of all, it is able to turn computers into instruments. Chuck has powered two "computer music" orchestras, namely PLOrk (Princeton Laptop Orchesta) and SLOrk (Stanford Laptop Orchestra). Besides, Chuck has served as experimental platformfor on-the-fly machine learning for real-time musicinformation retrieval prototyping[5]. In addition, it has also been applied in the development of music mobile-apps like Ocarina and Smule[5].

*The authors' names are ordered by last name.

CPSC 311 2017W1, 2017, Vancouver, BC, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.2.3 How our language is different? The success of Chuck demonstrates the significance of the role of programming language plays in music and audio produciton. Designing and applying programming lanuages in generating and modifying music is definitely a valuable task to do. And Chuck is a great example we can learn from when designing our own Domain Specific Language which is customized for ABC music notation. There are many language designs in Chuck that we can follow. For instance, we should definitely include primitive types such as int, float, and music specific types like duration, note etc. Moreover, having functions and common control structures like conditional statements and loops will also be helpful. With that being said, we will not copy all the features from Chuck into ABCD. What we will do differently is that ABCD will contain ABC related data types to natively support ABC notations. (...?) Besides, we will also (...?).

3.3 Overtone

Overtone is an Open Source toolkit for designing synthesizers and collaborating with music. It provides[3]:

- A Clojure API to the SuperCollider synthesis engine
- A growing library of musical functions (scales, chords, rhythms, arpeggiators, etc.)
- Metronome and timing system to support live-programming and sequencing
- Plug and play MIDI device I/O
- A full Open Sound Control (OSC) client and server implementation.
- Pre-cache - a system for locally caching external assets such as .wav files

Overtone's idea is about sound generation. "Let me answer from the synthesis perspective - which is one of my main interests. Learning to design new synthesisers is a pretty dark art, and most of the books/resources I found take a very theory-centric stance which I found to not be particularly useful." Said the author of Overtone, Sam Aaron[1]. The current design of ABCD is just let it compile to ABC, and let ABC deals with the sound generation part. The ultimate version of ABCD can be directly interpreted to play sound, so how Overtone hooks up with SuperCollider can be case-studied for our own implementation.

Another advantage of overtone is about collaborative programming. Overtone provides an API for querying and fetching sounds from http://freesopund.org and a global concurrent event stream[3]. Sharing music is a huge part of enable people to write music. ABC is a great source in text format to share music, and Overtone is a great source to share music in computer programs. However, to understand Overtone program is not as straight forward as reading ABC notation. ABCD will have programming feature, so we enable both sides — for people who writes complicated music programs and for people who just write plain ABC with some syntactic sugar to make their life easier.

From a language design perspective, Overtones design is based on Clojure. Two core component of Overtone are synths and ugens(unit-generator). Synths are trees of ugens. Ugens are standard Clojure functions and return data-structures which are understood by the macros demo and defsynth. You can pass arguments to the ugen functions to specify their behaviour. Synths are not ugens. Calling a ugen function returns a data structure which can be used in a synth design. Calling a synth as a function triggers (i.e. plays) that synth.[2] The story of overtone gives us some hints on the language design — we probably need to define what are the automics in the music world and make them first-class members of our language. However, for a sequence of sounds, they are probably playable/translatable objects.

4 POTENTIAL PROJECT

4.1 Features

4.2 Concrete Syntax

4.3 Parse into AST

After some research, we have found a few implmentations of parsing ABC. Sergi Mansilla has implemented an ABC parser in JavaScript, which parses the ABC notation to a JSON object[7]. Remo Dentato has developed a C library to parse the ABC notation[4], however, the source code can no longer be found on the website and the usage description is quite complicated. However, on Dentato's website, it offers some high-level design graph of the scanner and documentation on tokens. Hans Höglund has implemented a parser in Haskell[6]. Höglund's implmentation has a few limitations such as do not support volatile features, text strings, and macros. abc4j is a Java library that provides API to handle ABC music notation using Java. The source code can be downloaded on https://code.google.com/archive/p/abc4j/source. However, it only supports the v1.6 standard of ABC (the current version is v2.1). There are many other people who tried to implment parser for ABC but failed or stopped their project.

4.4 Compiler

5 CONCLUSIONS

This paragraph will end the body of this sample document.

REFERENCES

[1] Sam Aaron. 2013. What are the best resources for learning music theory that mesh with Overtone's theory-related facilities? (2013). Retrieved November 8, 2017 from https://stackoverflow.com/questions/20222445/what-are-the-best-resources-for-learning-music-theory-that-mesh-with-overtones

[2] Sam Aaron. 2014. Structuring Overtone Project. (2014). Retrieved November 8, 2017 from https://stackoverflow.com/questions/20861851/structuring-overtone-project

[3] Sam Aaron. 2016. Collaborative Programmable Music. (2016). Retrieved November 8, 2017 from https://github.com/overtone/overtone/blob/master/README.md

[4] Remo Dentato. 2009. ABCp: A parser for the ABC music notation. (2009). Retrieved November 9, 2017 from https://sites.google.com/site/abcparser/Home

[5] and Spencer Salazar Ge Wang, Perry R. Cook. 2015. ChucK: A Strongly Timed Computer Music Language. *Computer Music Journal* 39, 4, Article 5 (2015). https://doi.org/10.1162/COMJa00324

[6] Hans Hoglund. 2015. abcnotation: Haskell representation and parser for ABC notation. (2015).

[7] Sergi Mansilla. 2012. ABC notation parser for JavaScript. (2012). Retrieved November 9, 2017 from https://github.com/sergi/abcnode

[8] Ge Wang. 2008. *The ChucK Audio Programming Language "A Strongly-timed and On-the-fly Environmentality"*. Ph.D. Dissertation. Princeton University, Princeton, NJ 08544, USA. Advisor(s) Cook, Perry R.

[9] Ge Wang and Perry Cook. 2002. ChucK : Strongly-timed, Concurrent, and On-the-fly Music Programming Language. (2002).