Proposal: ABCD - ABC Music Notation extendeD with programming language 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 proposal for our new Domain Specific Language - ABCD, the extended language that adds programming features to the classic ABC notation.

KEYWORDS

Programming Language, Music, ABC Music Notation

1 INTRODUCTION

The *proceedings* are the records of a conference.[1] ACM seeks to give these conference by-products a uniform, high-quality appearance. To do this, ACM has some rigid requirements for the format of the proceedings documents: there is a specified format (balanced double columns), a specified set of fonts (Arial or Helvetica and Times Roman) in certain specified sizes, a specified live area, centered on the page, specified size of margins, specified column width and gutter size.

2 HOW IT FULFILLS THE PROJECT TYPE REQUIREMENTS

The purpose of our project fits into the fifth project type. We will design a DSL (Domain Specific Language) called ABCD which empowers users to edit and modify music sheets written in ABC music notation more effectively and efficiently. The users of ABCD will be professionals who use ABC to write their music. ABCD could help users in many ways, and some general user cases are listed below: - A trumpet player may want a program that transposes songs for C instruments to scores for B flat instruments - A composer writing a score in ABC may wish to change the key of the last x bars of their song - An arranger may wish to add a harmony to every instance of a certain note - A producer may wish to generate part of a song or a whole song programmatically All of these can be easily accomplished as programs written in ABCD. ABC alone is not powerful enough to do these programmatically – it would have to be done manually. An alternative solution would be using another programming language, but extra efforts on parsing and compiling ABC would be required, whereas ABCD would support those natively.

3 HOW IT COULD BE A FULL 100% PROJECT

We intend to design a Domain-Specific Language extending the ABC music notation language. ABC is similar in power and purpose to LaTeX: it allows a user to hand-type a document that represents a piece of sheet music, and can be compiled to a PDF or played aloud using appropriate software. Our language, ABCD, will enhance ABC with common programming features such as identifiers and functions, empowering users to programmatically modify existing ABC documents and generate new music. A full implementation of the language would involve:

An EBNF defining concrete syntax found in ABCD that are not in ABC (and possibly including the concrete syntax for ABC) An ABCD parser An abstract syntax implementation Evaluation implementation: A compiler to produce ABC files from the ABCD abstract syntax OR An interpreter to run ABCD programs, which have ABC files as input and output [deprecated]

ABCD should have native support for using ABC files as input to ABCD programs, allowing users to easily modify existing ABC scores. There are two general ways we can imagine using ABCD: writing scripts that compile into ABC files after modification or creation, or writing programs that may accept ABC files as input and output modified or generated ABC files. By outputting ABC files, our users can leverage existing software that works with ABC, including music players and pdf compilers. For the scripting approach, an ABCD file may contain definitions of identifiers and functions as well as ABC notation. When compiled/evaluated, functions will modify, add, or remove ABC code in the script, and the script will

---

evaluate to one or more ABC documents. For the programming approach, an ABCD file may contain definitions of identifiers and functions, as well as a main program. The program can optionally take one or more ABC files (or something else) as input, create and modify ABC code during its evaluation, and optionally output one or more ABC files. We may wish to consider modifying files in-place as well. It is worth noting that this form of ABCD program is not a document containing a ?tune? like an ABC file does, and trying to compile it to an ABC document is not meaningful. We can think of the AST as the final, target form of an ABCD program, where a script would ultimately be evaluated to an ABC file. We can think of an ABCD program as a composer that creates new tunes or as an editor or remixer who transforms existing tunes. From a programmer?s lens, we can view an ABCD script as akin to a LaTeX document or HTML file, and an ABCD program more like a C program we might write for the unix command line, a pipeline that performs a transformation on data files or strings. We would likely write our parser interpreter/compiler in python to harness its string processing capabilities, as ABC has a plain-text format where characters often need to be considered individually. For a final project, we would likely also write a small ABCD program to demonstrate a possible use case for ABCD and showcase some of its capabilities. This would likely be a program that applies some transposition, addition of harmonies to certain notes, and some rhythm changes, and we would demonstrate by applying the program to a small ABC tune and playing the result or printing it as a pdf.

4 LIST OF STARTING POINT DOCUMENTS

The following is a list of sources we would consult during the project.

First of all, http://abcnotation.com is a general site for all ABC related documents. In particular, http://abcnotation.com/examples contains straightforward specifications (illustrated by examples) for how ABC works as a markup language that represents sheet music. http://abcnotation.com/software provides a list of command lines and software packages that can read or edit ABC.

Another source that allows us to understand ABC better and, more importantly, implement ABC software applications is the ABC wiki page: http://abcnotation.com/wiki/abc:standard:v2.2 In addition to the formal specifications of ABC, this site also provides information about ABC extensions, file readers, transposition operations, transposition macros, etc. It allows us to see the various features already supported by ABC, such as transposition. We would need to think about what other features we can implement and/or how to improve the existing features in our language.

http://www.mandolintab.net/abcconverter.php is an online converter that we can use for converting ABC into sheet music and export the file as MIDI, PDF, etc. It could be useful for testing whether a ABC file is valid. We may also use EasyABC (http://www.nilsliberg.se/ksp/easyabc/), an open source ABC editor, if we find the converter not sufficient.

We may also learn about the limitations of ABC at https://music.stackexchange.com/questions/23841, where many ABC users and software developers share their experiences of working with ABC, such as in parsing and in practical use.