Abcl: Abc music notation with rich chord support

José João Almeida
Departamento de Informática / Centro Algoritmi
Universidade do Minho, Campus de Gualtar, Braga, Portugal
jj@di.uminho.pt
https://orcid.org/0000-0002-0722-2031

Abstract

It is well known the relevance of accompany chords but there is a lack of tools capable of automatically generating sound from them.

In this paper we describe a domain specific language (Abcl) aimed to be a prototyping environment for new experimental music operators. Currently Abcl: (1) adds support for accompany chords (chordmode, instruments, chord-lines); (2) adds clearer support for percussion (drums, drum-machine) (3) adds a support for variables and functions.

Abcl tool is a syntactic-preprocessor that produces Abc. The DSLToolkit, used to create Abcl, is also briefly presented and discussed in the paper.

2012 ACM Subject Classification Software and its engineering → Domain specific languages

Keywords and phrases music, Abc music notation, domain specific language

Digital Object Identifier 10.4230/OASIcs.SLATE.2018.8

Category Short Paper

1 Introduction

Texts with lyrics and chords are often used as an easy way of give musicians the minimal information necessary to accompany a music. Adding (guitar) chords to lyrics is a popular way of helping in the process of learning a music and learning how to play it, how to provide the musical accompaniment for singer or a musical ensemble. Most of the times, documents with chords and lyrics don't provide all the necessary information to be possible to automatically generate sound.

Consider the example in Listing 1. It should be enough for a musician if she already knows the music, but to automatically generate sound, we need to define:

- the measure (3/4);
- the general velocity (1/4=100);
- the duration of each chord;
- what instrument should be used to play the chords and the bass;
- how should the set of nodes of the chord be played during each bar.

This word was developed in the context of the Abc music notation community and we intended to discuss a set of extensions to better support accompany chords and drums. The approach taken was to develop a notation (Abcl DSL) and build a syntactic preprocessor to extend Abc. This way we built an experimental prototype that we can use in our music projects, where syntax can easily changed and discussed. When we obtain a more stable version, we plan to submit a proposal for the next version of abc notation standard.

In this rest of this section we will briefly introduce Abc music notation, and the experimental accompany chord primitives already existent in some Abc processing tools. In

Listing 1 A 20th century *folia* by J.Berthier.

```
T: Laudate Dominum
C: Jacques Berthier

La[Am]udate do[E]minum,
La[Am]udate do[G]minum,
O[C]mnes, ge[G]ntes,
A[Am]le[F]lu[Dm]ia[E]

La[Am]udate do[E]minum,
La[Am]udate do[G]minum,
O[C]mnes, ge[G]ntes,
A[Am]le[Dm]lu[E]ia[Am]
```

Listing 2 Example of the Abc notation for the score presented in Figure 1.

```
X:101
T:Verbum caro factum est
C:Anonimous, 16th century
M:3/4
L:1/8
K:G
V:1 clef=treble-1 name="Soprano" sname="S."
G4 G2 | G4 F2 |A4 A2 | B4 z2|: B3 A GF| E2 D2 EF| G4 F2 | G6 !fine!:|
w: Ver- bum|ca- ro|fac- tum|est|Por - que *|to - dos *|hos sal -|veis
```

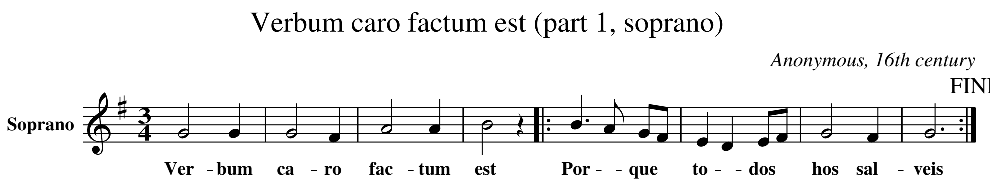

Verbum caro factum est (part 1, soprano)
Anonymous, 16th century
FINE
Soprano
Ver - bum ca - ro fac - tum est Por - - que to - - dos hos sal - veis

Figure 1 Music score for the Abc in Listing 2.

Section 2 we will present some of the features of Abcl tool and in Section 3 we will discuss the development of the syntactic preprocessor.

1.1 Abc music notation

Abc [13, 7, 4] is used as the base notation throughout all of this paper. The extract presented in Listing 2 illustrates the use of Abc notation, and Figure 1, its corresponding score and MIDI.

Music languages are linguistically different from programming languages. In general music description languages tend to be a merge of several sublanguages like: information fields and metadata, lyrics, tunes (notes and duration), annotations, accompany chords symbols, typesetting geometry. Each sublanguage has a different linguistic flavor.

Naturally, the power of Abc is strongly connected with its processing tools [12, 1, 3, 10] and related projects [5, 2].

Although Abc standard has just a very simple support for chords, the tool Abc2midi[1] has a set of extensions that cover some relevant chord accompany properties:

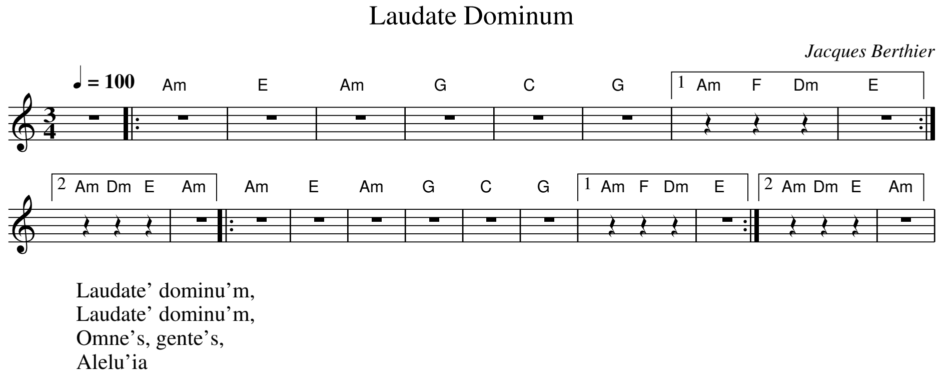


Figure 2 Music score output for Abcl code from Listing 3.

- `%%MIDI chordprog n`, to define the instrument used for the accompany;
- `%%MIDI bassprog n`, to define the instrument for the bass notes;
- `%%MIDI chorvol n`, to set the volume of the chord notes;
- `%%MIDI bassvol n`, to set the volume of the bass notes;
- `%%MIDI chordname ....`, to define e chord variant;
- `%%MIDI gchordon`, to turn on chord sounds;
- `%%MIDI gchordoff`, to turn off chord sounds;
- `%%MIDI gchord ...`, to define the arpeggiate notes for a bar;
- `%%MIDI gchordbars n`, to express that gchord arpeggiate notes are to be extended for n bars.

The use of these MIDI primitives is cryptic and hard to read and master.

1.2 Accompany Chords

Accompany chords play a very important role in partial music. They are a popular way of starting to learn music but can also be a support for musical improvisation, a way to prepare band rehearsal, to create simple karaoke, and a starting point to several relevant music activities. Several projects discuss [11], teach and *guess* [6] chords.

1.3 In this paper

In this paper we describe a domain specific language [9, 8] – Abcl – that is a syntactic preprocessor that: (1) adds clearer support for accompany chords (chordmode, instruments, chord-lines) (2) adds clearer support for percussion (drums, drum-machine) and (3) adds a reduced support for variables and functions. Abcl internally is written in Perl+DisLex.

As usual, we will call Abcl to the language and `abcl` to the compiler that converts Abcl to Abc.

2 Abcl by example

Example 1

Consider the Abcl example from Listing 3. Executing "`abcl ex.abcl > ex.abc`" we get the output presented in Figure 2.

Listing 3 Abcl example.

```
chordmode 3gui =b2c2ih guitar
chordmode 3gui1=c3 guitar
chordmode 3gui2=ccc guitar

X: 1
T: Laudate Dominum
C: Jacques Berthier
M: 3/4
L: 1/4
Q: 100
K: Am
 ch:
 |: \3gui { Am|E|Am|G|C|G |1
  \3gui2 Am F Dm | \3gui1 E :|2
  \3gui2 Am Dm E | \3gui1 Am }↦\v1
 |: \3gui1 \v1 |
W:Laudate' dominu'm,
W:Laudate' dominu'm,
W:Omne's, gente's,
W:Alelu'ia
```

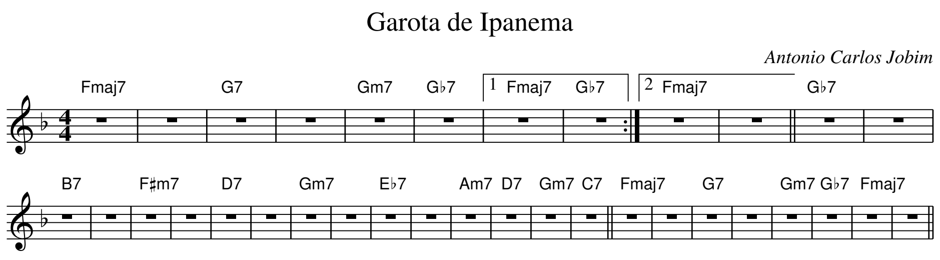


Figure 3 Music score generated for "Garota de Ipanema".

In this example, we can see the use *chordmode* instructions to define the arpeggios to be used. We can also see and ear the use of different solutions to build different spaces for improvisation.

The notation `{` *music* `}↦\id` is used to store *music* in a variable `\id`, to be reused whenever useful. Abcl also provides other ways of defining variables (see Appendix A) and functions.

Although not much relevant, the generates Abc (file `ex.abc`) is presented in Listing 4.

Example 2

In the following example we discuss some challenges related to Bossa Nova *unpredictable* rich rhythms. In this example we define a set of chords for "Garota de Ipanema". The generated accompany midi (completely unacceptable!) is using the default values of arpeggio, as presented in Figure 3.

In order to improve it a "Bossa Nova beginner's guide" was consulted. Figure 4 is an exercise for students that are learning how to accompany this type of songs. Please notice the always changing rhythm. Please note that this example is not the best choice for "Garota

Listing 4 Generated Abc file from code in Listing 3.

```
X: 1
T: Laudate Dominum
C: Jacques Berthier
M: 3/4
Q: 100
K: Am
 z3|:
%%MIDI gchordbars 1
%%MIDI gchord b2c2ih
%%MIDI chordprog 24
%%MIDI bassprog 24
  "Am"z3|
  "E"z3|
  "Am"z3|
  "G"z3|
  "C"z3|
  "G"z3|1
%%MIDI gchordbars 1
%%MIDI gchord ccc
 ... and more 70 similar lines
```

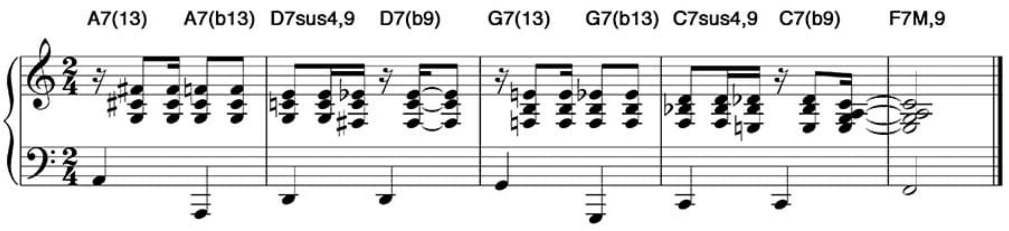


Figure 4 Example of an exercise of "Bossa Nova".

de Ipanem".

In Listing 5 we used two sub-patterns: "`bn1 = first 2 bars ; bn4 = first 4 bars`". Clearly these arpeggios introduce a relevant change in the MIDI, and can be a starting point to discuss the use of guitar/piano in Bossa Nova.

3 Abcl preprocessor tool

As we said before, our project deals with enriching a musical language with new functionality and clearer syntax. We want to be able to prototype experimental operators and syntax. Music tends to be the merge of multi sublanguage with different linguistic flavor. This raised some DSL-building challenges.

In the first experiences, we tried the set of lexical preprocessor. These tools cover pre-parsing textual substitution (macro), file inclusion and conditionals. Although useful the most popular open-source preprocessor like Gpp (GNU preprocessor) or CPP (C preprocessor) were almost impossible to use without introduce deep changes in the language. Generic lexical preprocessors like m4 also prove to be difficult for the current task.

In addition to lexical-preprocessors functionality, for the current project, we needed:

Listing 5 Sub-pattern usage.

```
chordmode bn1= fczc-bzcz|bzcc-fccz guitar
chordmode bn2= c guitar
chordmode bn3= gzcz-zczc guitar
chordmode bn4= fczc-bzcz|bzcc-fccz|fczc-bzcz|bzcc-fczc piano

X: 1
T: Garota de Ipanema
C: Antonio Carlos Jobim
M: 4/4
L: 1/4
K: F
ch: \bn1 Fmaj7| |G7| |Gm7|Gb7 |1 Fmaj7|\bn2 Gb7 :|2 Fmaj7| ||
 \bn3 Gb7| |B7| |F#m7| |D7| |Gm7| |Eb7| |Am7|D7|Gm7|C7 ||
 \bn4 Fmaj7| |G7| |Gm7|Gb7|Fmaj7| ||
```

- State conditions (like the ones presented in Flex) in order to deal with sublanguage heterogeneous syntax.
- Regular-expressions tools: in order textually rewrite new syntax.
- reflexive capabilities (runtime definition of functions).

In this context we choose to build a DSL prototyping toolkit, with a Flex-inspired language processor (DisLex), aimed to support syntactic preprocessors.

DisLex: a Flex-inspired language processor

DisLex is a Flex-flavored lexical analyzer for Perl. It is part of `Parse::DSLUtils`, a Perl module aimed to help in the construction of DSL. In complement, `Parse::DSLUtils` also covers `Parse::Yapp` simplification, and templates functionality.

Following some relevant features of DisLex tool:

- By default, input is slurped to a variable (`$yyfile`).
- Regular-expression based using `\G` and `pos($yyfile)` to keep current position in a efficient way.
- `(RegExp, Perl-action)` rules are the basic building blocks. Perl's regular-expressions offers a very rich group capture functionality that proved to be very effective. In flex group capture is not available.
- Full Unicode expressions available
- Greedy and non-greedy regular-expressions operators available.
- No support for chooser-longest-match disambiguation rule.
- State conditions: to help in implementation of automata, using:
  - `BEGIN state` to change states
  - `REC state` and `DONE` to change and came back (similar to Flex `yy_push_state(state)`, `yy_pop_state()` functions)
- It includes a large set of predefined regular-expressions, covering some non-regular patterns like:
  - curly-bracket blocks, XML elements, Latex environments

When used as a lexical analyzer, DisLex typically, uses rules like

```
(\d+) return("INT",$1))
```

It provides directives (syntactic sugar) to skip white spaces and comments:

```
%white [\ \t]+
%comments #.+
```

Some less common functionality:

- Return many – sometimes is easier to return several tokens (using a queue of symbols to be returned)
  `\+= { returnmany(['=','='],["INT",1],['+','+']);}`
- a predefined `yylexdebug(func,file)` to help testing and debugging lexical analyzer's behavior.

A set of predefined functions is provided to cover some simplified cpp-like functionality (includes, defines).

4 Conclusions

Although in an initial stage, from our experience, Abcl tool proves to be useful for:

- providing support for modeling arpeggios for specific styles of music;
- practice improvisation supported by neutral chord bases;
- experimentation on accompany solutions.

The use DisLex and Parse::DSLUtils were crucial to obtain a working prototype in a very short time.

We are currently working with Abcl with multi-voice chords, and multi-voice drums and we already have interesting examples of the use of chords and percussion using Abcl DSL language.

References

1 James Allwright and Seymour Shlien. abc2midi: Abc to midi translator. `http://abc.sourceforge.net/abcMIDI/`. Tool.

2 José João Almeida, Nuno Ramos Carvalho, and José Nuno Oliveira. Wiki::Score a collaborative environment for music transcription and publishing. *Information, Services and Use (ISU)*, 31(3-4/2011):177–187, 2012. `doi:10.3233/ISU-2012-0647`.

3 Bruno M. Azevedo and José João Almeida. Abc with a unix flavor. In *2nd Symposium on Languages, Applications and Technologies (SLATE)*, volume 29, pages 203–218, 2013. `doi:10.4230/OASIcs.SLATE.2013.203`.

4 Abc Comunity. Abc musical notation – standard version 2.2, 2013. Standard. URL: `http://abcnotation.com/wiki/abc:standard:v2.2/`.

5 Michael Scott Cuthbert and Ben Houge. Music21 - a toolkit for computer-aided musicology. `http://web.mit.edu/music21/`. Toolkit's Homepage.

6 W. Bas de Haas, José Pedro Magalhães, and Frans Wiering. Improving audio chord transcription by exploiting harmonic and metric knowledge. In *13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

7 Guido Gonzato. Making music with abc2 - a pratical guide, 2018. URL: `http://abcplus.sourceforge.net/abcplus_en.html`.

8 Tomaž Kosar, Sudev Bohra, and Marjan Mernik. Domain-specific languages: a systematic mapping study. *Information and Software Technology*, 71:77–91, 2016.

9 Tomaž Kosar, Nuno Oliveira, Marjan Mernik, Maria João Varanda Pereira, Matej Črepinšek, Daniela da Cruz, and Pedro Rangel Henriques. Comparing general-purpose and domain-specific languages: An empirical study. *Computer Science and Information Systems*, 7(2):247–264, 2010. doi:10.2298/CSIS1002247K.

10 Nils Liberg et al. EasyABC abc editor, 2016. URL: http://easyabc.sourceforge.net/.

11 José Pedro Magalhães and Hendrik Vincent Koops. Functional generation of harmony and melody. In *2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modeling and Design*, pages 11–21, 2014. doi:10.1145/2633638.2633645.

12 Jean-François Moine. abcm2ps - abc to postscript/eps/svg translator. http://moinejf.free.fr/. Tool.

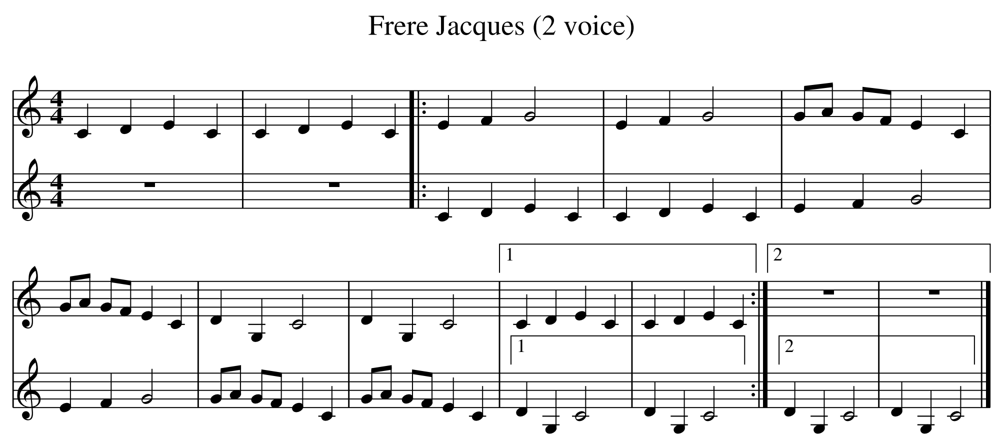
13 Chris Walshaw. Abc notation. http://abcnotation.com/. Musical Notation.

A Cannon

Example of a very simple use of tune variables: the structure of a 2 voice cannon (Frere Jacques).

```
\p1={CDEC | CDEC}
\p2={EFG2 | EFG2 | G/2A/2 G/2F/2E C | G/2A/2 G/2F/2E C}
\p3={DG,C2 | DG,C2}

X:1
T: Frere Jacques (2 voice)
M: 4/4
L: 1/4
K: C
 [V:1] \p1 |: \p2 | \p3 |1 \p1 :|2 Z2 ||
 [V:2] Z2 |: \p1 | \p2 |1 \p3 :|2 \p3 ||
```


Frere Jacques (2 voice)
1
2
1
2