Tool for musical harmonization through Anser Set Programming

Rodrigo Martín Prieto
Pedro Cabalar Fernández

Universidade da Coruña

r.martin@udc.es

January 28, 2016

Overview

Motivation

- Musical teaching is still very traditional nowadays.
- Self-teaching of music theory is hard.
- There aren't many tools to aid and guide students and self-taught students.
- Composition tools seek results assuming that the user knows musical theory.

Motivation

During the course of a previous year's subject about representation of knowledge, Professor Cabalar proposed his students the implementation of a *Canon* composer through *Answer Set Programming* as a practical exercise.
The idea of combining Computer Science with music wasn't new, but it still was veary appealing.

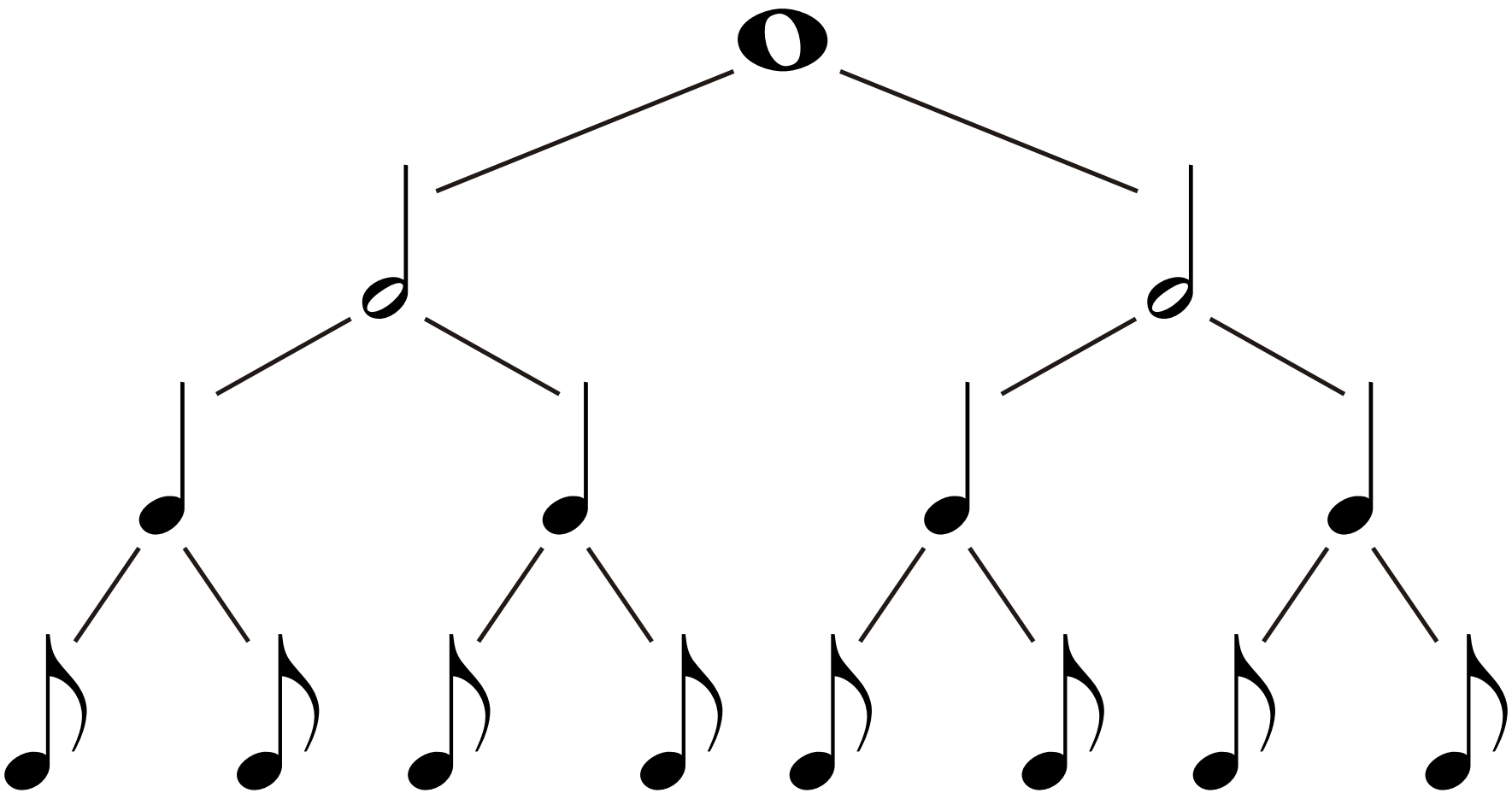
ANTON

ANTON, a composing tool by Martin Brain previously proved the succesfulness of applying *Answer Set Programming* to the musical field, creating a duet composing tool following the style of Renaissance's composer Giovanni Pierluigi da Palestrina by using his well defined musical rules.

Goals

- Harmonize and annotate chords over any musical score
- Given a certain harmonization, be able to complete any incomplete voice of the score
- Complete on purpose blank sections of incomplete voices of the score
- Add new voices that complement the voices already in the score

Figures and Rhythm



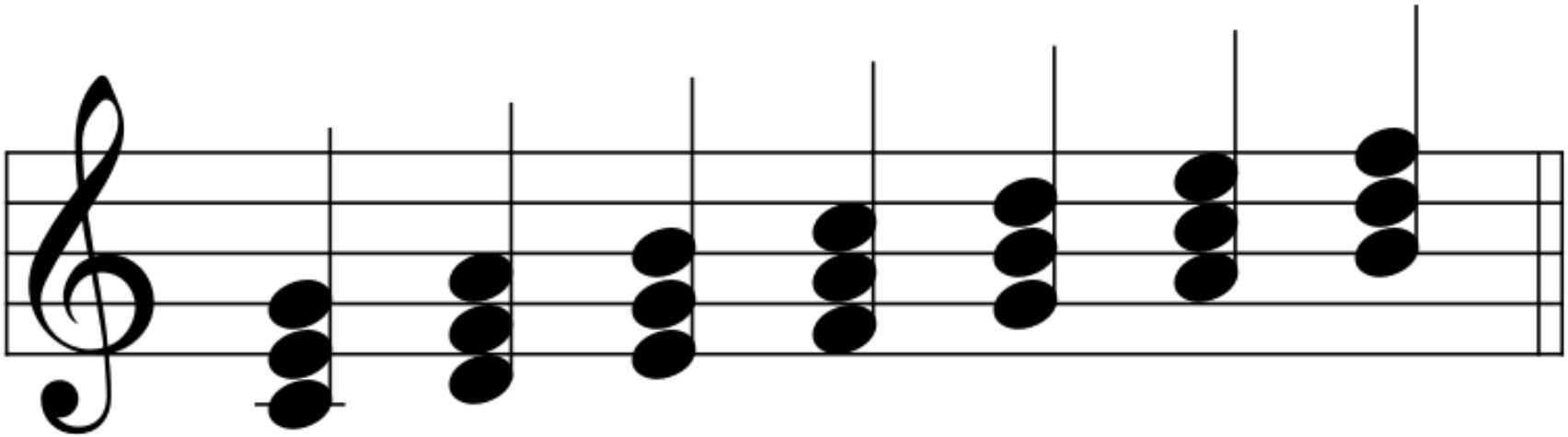
Melody

- Horizontal dimension of music
- Previous notes are previous in time, posterior notes are posterior in time
- Pitch is represented by the height at which the note is written, higher position means higher pitch
- The pitch of a note matters in relation to the adjacent notes

Harmony

- Vertical dimension of music
- Only present in polyphonic pieces or pieces with polyphonic instruments
- Two notes of different voices that play at the same time form a chord
- The pitch of a note matters in relation to the notes above and below in the other voices
- Uses grades of the scale instead of the explicit pitch values of notes to form different chords

A little sample

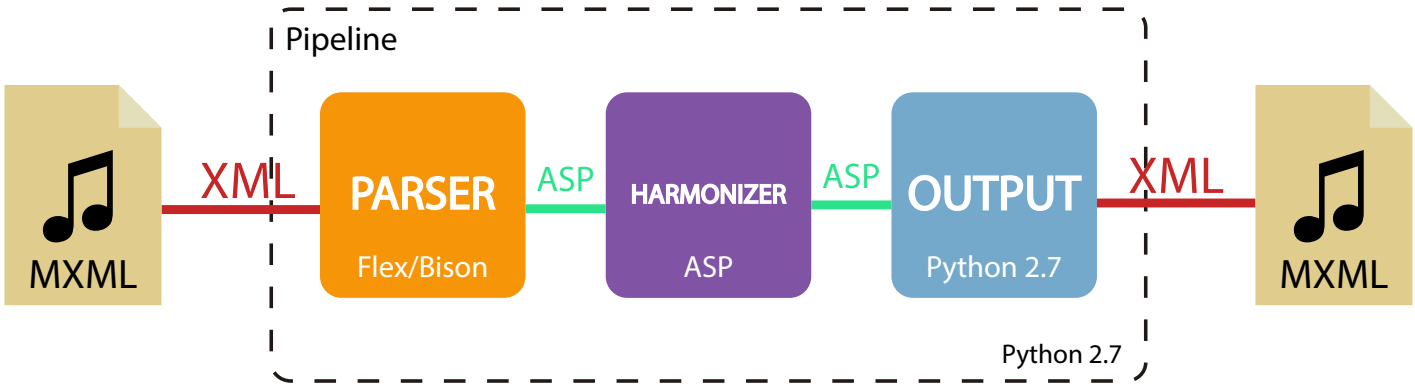


Demonstration

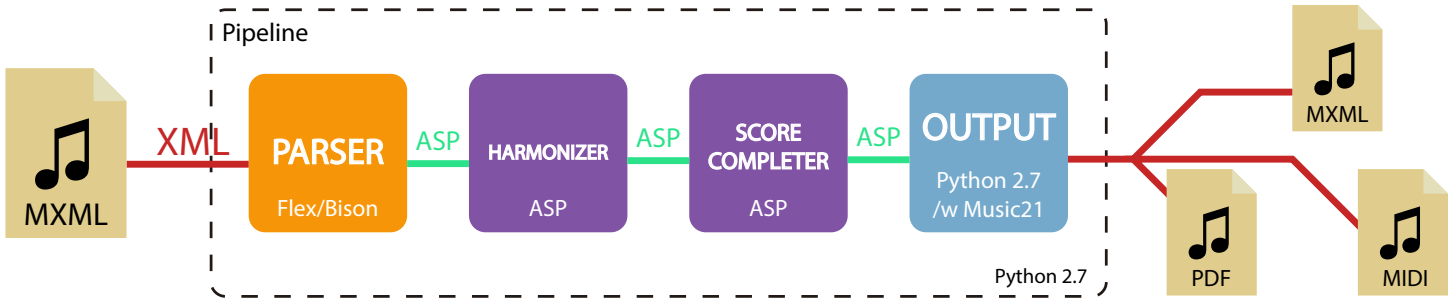
The piece selected for the Demo will be Greensleeves by Henry VII of England. We will see and hear the results of three diffrent processes performed by the tool.

- Harmonization and chord annotation of the score
- Given the previous harmonization, the tool will complete a section of the Cello part
- Given the previous harmonization, the tool will complete a section of the Violin part

Initial Architecture



Current Architecture



The ASP Core

- Independent of the solving process and its heuristics
- The power of ASP resides in this independence
- The problem only needs to be specified by rules and constraints

Rules and Constraints

Example (Note Representation)

```
figure(1,1,22).
note(1, 72, 22).
```

Example (Chord Assignation)

```
1 { chord(HT,C) : pos_chord(C) } 1 :- htime(HT).
```

Example (Score Completion Constraint)

```
:- freefigure(V,D,FB), ex_grade(V,G1,B1), ex_grade(V,G2,B2),
   B1 >= FB,B2 >= FB,B1 < FB+D,B2 < FB+D,B1 != B2,G1 != G2.
```

Harmonization

- Notes are converted to grades of the scale given the key and mode
- Chords are assignated to the harmonizable times of the score
- Errors are calculated and solver determines the fittest chords for each section.

Score Completion

- Only used if there are new voices or sections that need to be completed
- Given the incomplete or new voices' *tessiturae* notes are assignated to the available positions.
- Errors are calculated and solver determines the fittest notes for each time

Melodious Preferences Module

- Althought not composing melodiously, this module smoothens the output
- Checks the tendency of the voices already on the score and makes the new voices imitate them
- Smoothens the melodic jumps between notes of a same voice
- Reduces the number of consecutive repeated sounds

Sixths Link Preferences Module

- Progressions of the second inversion of chords are very common in choral music
- Creates a per-time harmonization of the score
- Finds patterns of second inversion of chords linked in other voices
- Tries to continue the progression and creates new progressions of this kind if able

User Configuration

The style of the resulting scores produced by the tool is determined by the maximization and minimization of many preferences. These preferences are weighted to be able to specify the significance of each of the measured predicates. Users can define their own preferences by making use of configuration files.

Editors and Formats

- Scores are edited in an external editor not developed within this project
- Musescore 2 was the chosen editor for being free and Open Source
- The editor exports the score to a standard music exchange format, MusicXML
- MusicXML was chosen among other formats for its parsing simplicity and flexibility

Parser

- The project also included the development of a little parser
- Written in C with the libraries Flex and Bison
- Transforms the score in MusicXML to the ASP logic facts that the ASP module uses later
- Performs various tasks as:
  - Subdivides notes to the length of the smallest figure in the score
  - Detects most likely key from the score's clef
  - Reads measure sizes
  - Transforms chords name above the score to grades

Pipeline

- Written in Python
- Coordinates the different modules secuentially
- Gives feedback to the user throught the command line
- Allows the user to pick the desired solution for harmonization and score completion
- Calls to the internal representation library to store the results of the Harmonization and completion as Python objects.

Output Module

- Written in Python with the toolkit Music21
- Transforms the internal representation of the solution to a Music21 representation
- Exports the Music21 representation to the desired format
- Some supported formats are Lilypond, PDF, Musescore, MusicXML or MIDI
- Allows the result to be saved or directly shown/played

Measured executions results

Score	T. Harmonization	T. Measure	T. New Voice
Greensleeves	1.016s	1.926s	4m 49.032s
Menuet	0.631s	0.726s	3m 50.376s
Joy to the World	2.381s	3.813s	7m 17.115s
Twinkle Twinkle	0.685s	0.716s	2m 31.299s

Charge Tests

Test	Time
4 measures	1.481s
8 measures	2.394s
12 measures	3.978s
16 measures	3.982s
20 measures	5.966s
1 voice	2m 31.299s
2 voices	25m 17.298s

Future Work

- Improve output and correct tiny representation mistakes, althought most of them are expected to be corrected in the 3.0 version of the Music21 toolkit
- Implement a plugin interface for MuseScore 2 so the tool can be accessed and manipulated through the editor itself
- Improve detection of weak and strong beats of measure
- Be able to work with irregular figures such as duplets or triplets
- Research about modulation and implement it in the tool
- Improve execution times for the inclusion of new voices
- Include rhythmic patterning in the new generated voices
- Ask for feedback from professional harmony teachers and polish the tool so it can be used in teaching

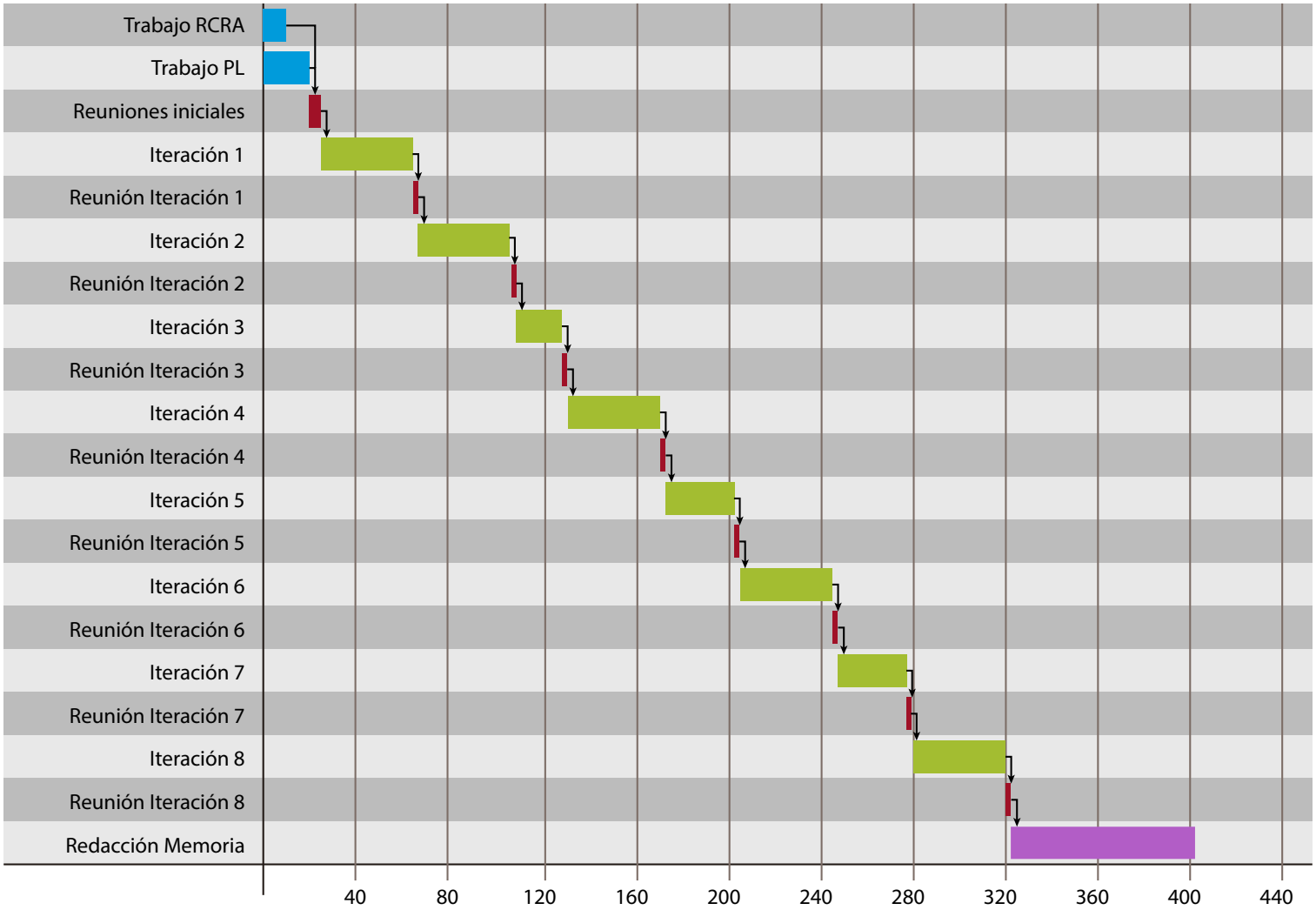
Development Cycle

- Spiral development with prototypes
- SCRUM

Development Costs

Profile	**Cost/Hour**	**Hours**	**Total**
Student	5.5€	362	1991€
Director	9€	12	108€
Total			2099€

Iteration Breakdown



Conclusions

- Achieved the main goals of the project and extended some of the functionality
- The tool produces correct scores in good times
- The interface is pretty poor
- User still needs informatic knowledge to use it

The project as it is now needs more work to achieve full usability and be able to reach the target it's intended to. Despite this, the project works really great and demonstrates once again the validity of ASP to many aspects of life, such as music.

Tool for musical harmonization through Anser Set Programming

Rodrigo Martín Prieto
Pedro Cabalar Fernández

Universidade da Coruña

r.martin@udc.es

January 28, 2016