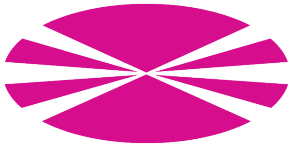


UNIVERSIDADE DA CORUÑA

Tool for musical harmonization through Answer Set Programming

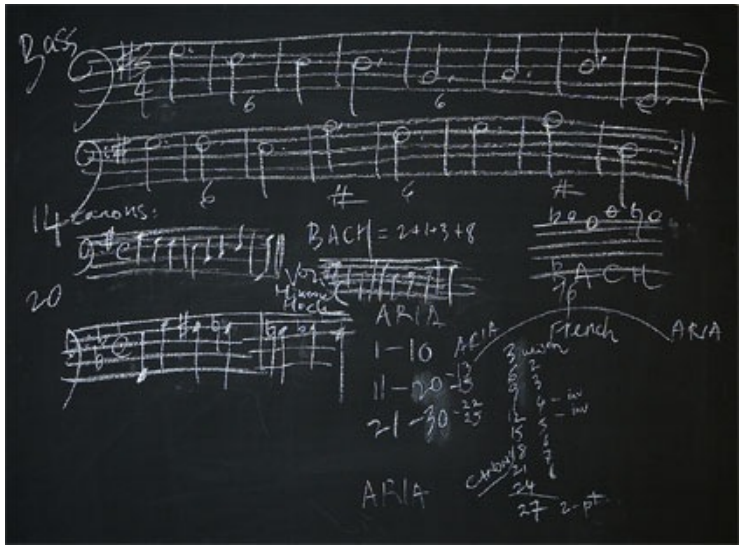
Student:
Rodrigo Martín Prieto

Director:
José Pedro Cabalar Fernández

February 7, 2016

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.



ANTON

- ANTON [Bra10] is a full-fledged composition tool written in ASP
- Limited to choral pieces for two voices
- Only Giovanni Perluigi da Palestrina's style

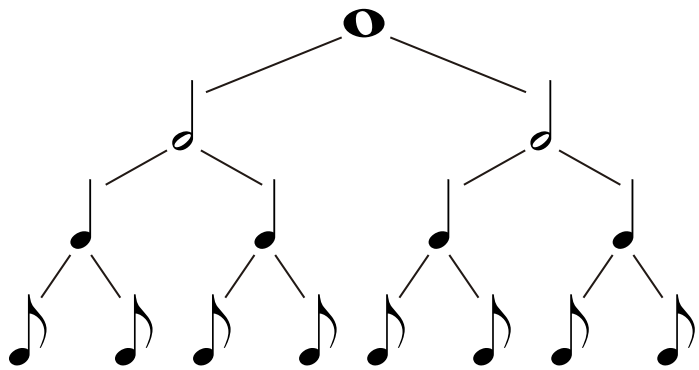
Goals

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

Overview

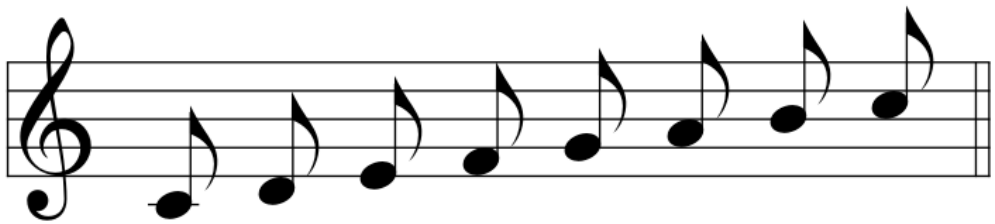
Figures and Rhythm

- Every note is represented by a figure that determines it's length
- Each figure can be subdivided in two
- Rhythm is created by combining figures of different lengths with special symbols called silences



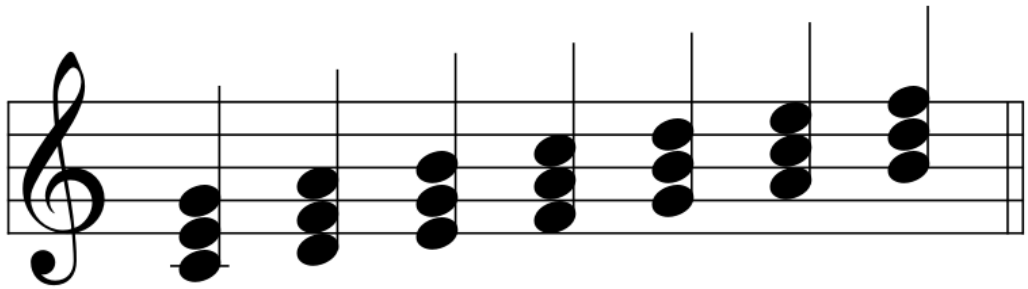
Melody

- Horizontal dimension of music
- 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
- The pitch of a note matters in relation to the notes above and below in the other voices
- Two notes of different voices that play at the same time form a chord



Overview

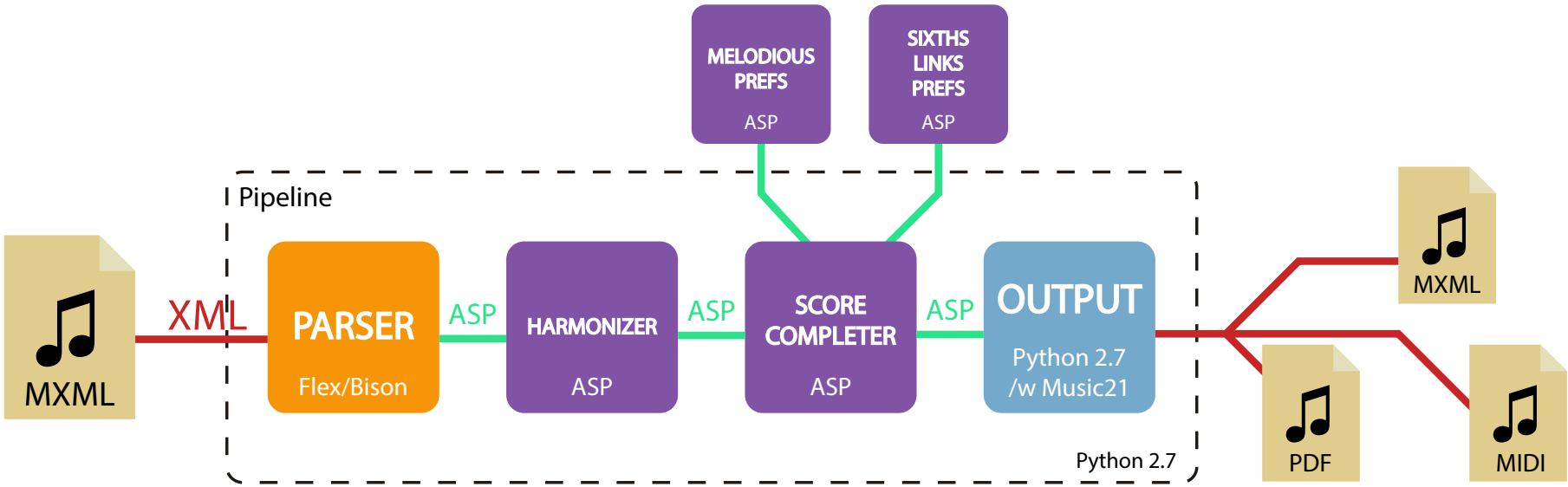
Demonstration

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

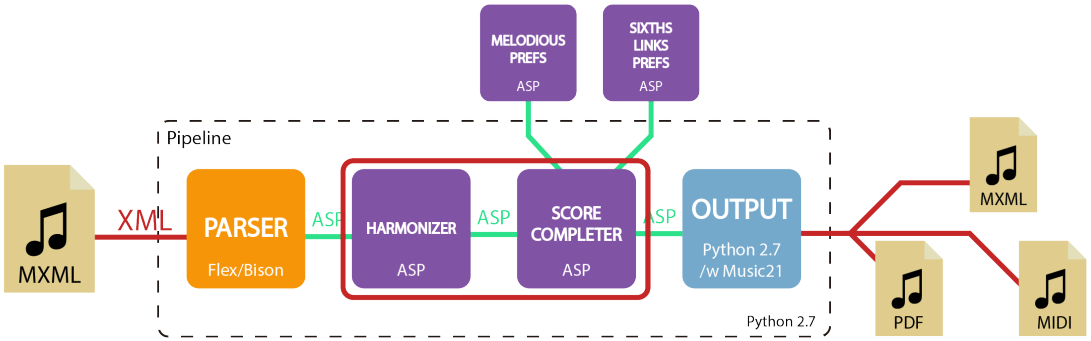
- Harmonization and chord annotation of the score
- Given the previous harmonization, the tool will complete a section of the Cello part
- Given the previous harmonization, the tool will complete a section of the Violin part

Overview

haspie's 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 (ASP Fact)

```
#const n = 8.
number(1..n).
```

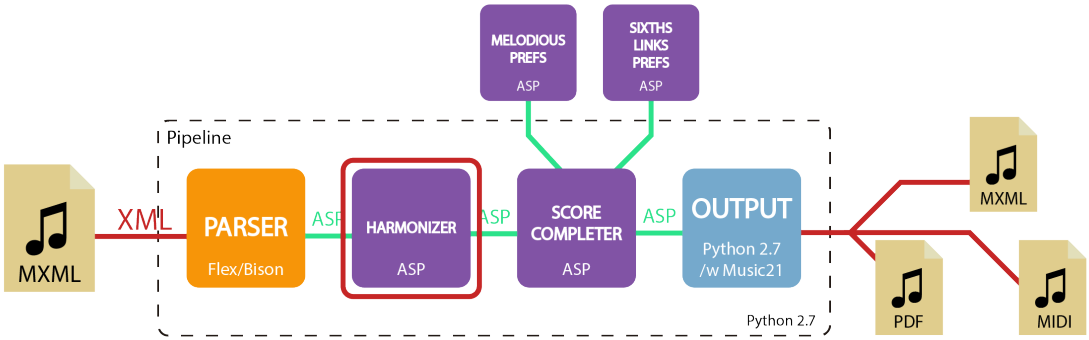
Example (Generator Rule)

```
1 { q(X,Y) : number(Y) } 1 :- number(X).
1 { q(X,Y) : number(X) } 1 :- number(Y).
```

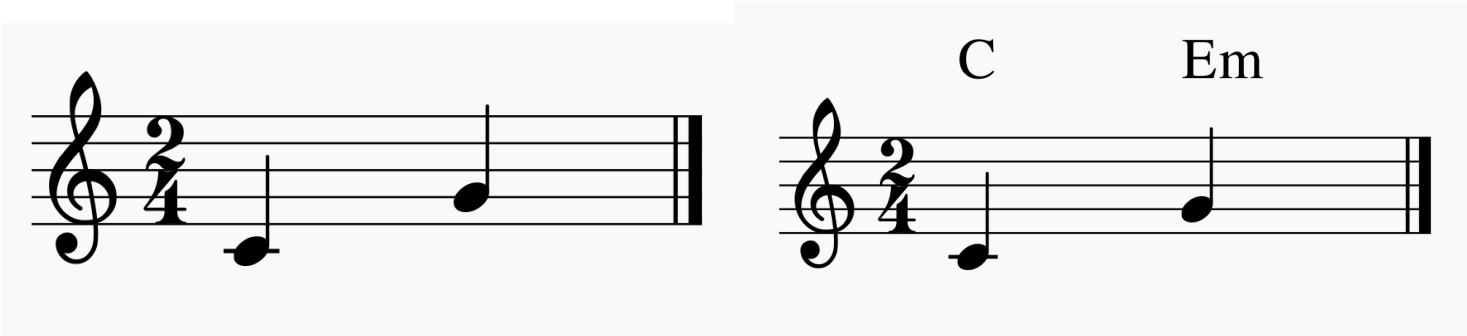
Example (Constraint)

```
:- number(X1;X2;Y1;Y2), q(X1,Y1), q(X2,Y2), X1 < X2,
   Y1 == Y2.
```

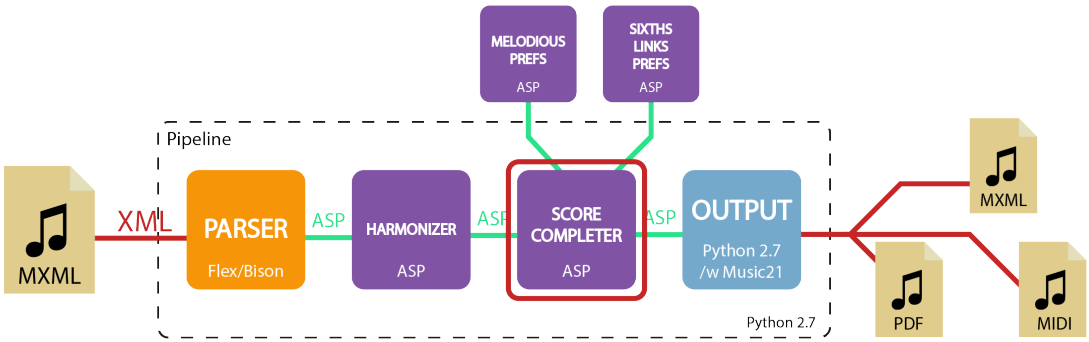
Harmonization



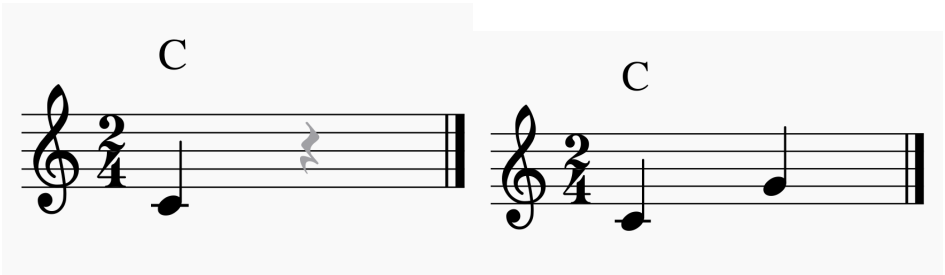
- Notes are converted to grades of the scale given the key and mode
- Chords are assignated to the harmonizable times of the score
- Errors are calculated and solver determines the fittest chords for each section.



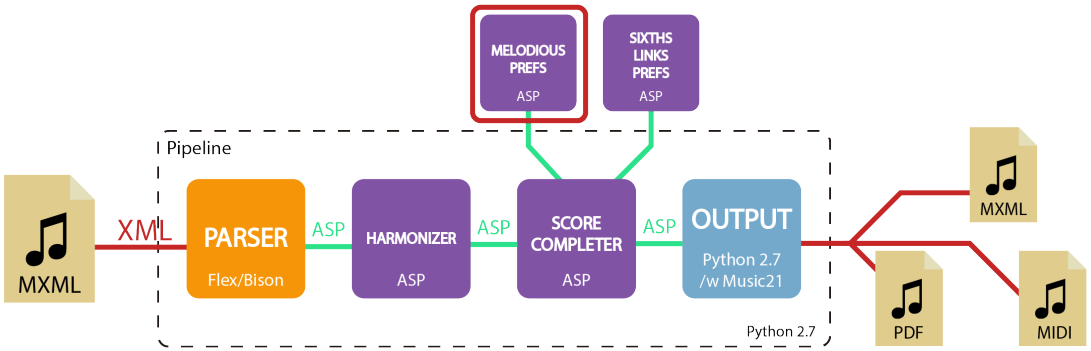
Score Completion



- 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

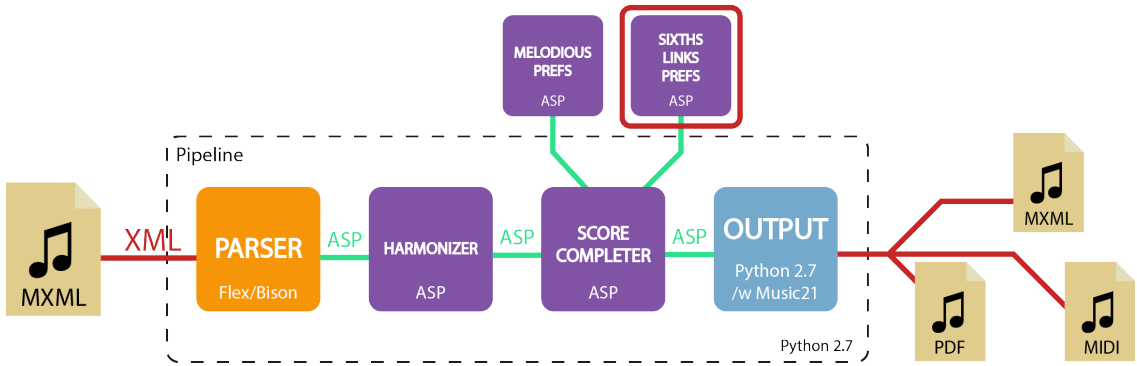


Melodious Preferences Module



- 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



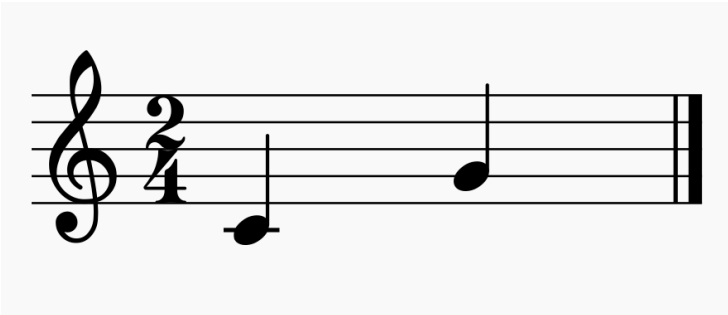
- Progressions of the second inversion of chords are very common in choral music
- Creates a per-time harmonization of the score
- Finds patterns of second inversion of chords linked in other voices
- Tries to continue the progression and creates new progressions of this kind if able

User Configuration

- The style of the resulting scores produced by the tool is determined by the optimization of many predicates
- These optimizations 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

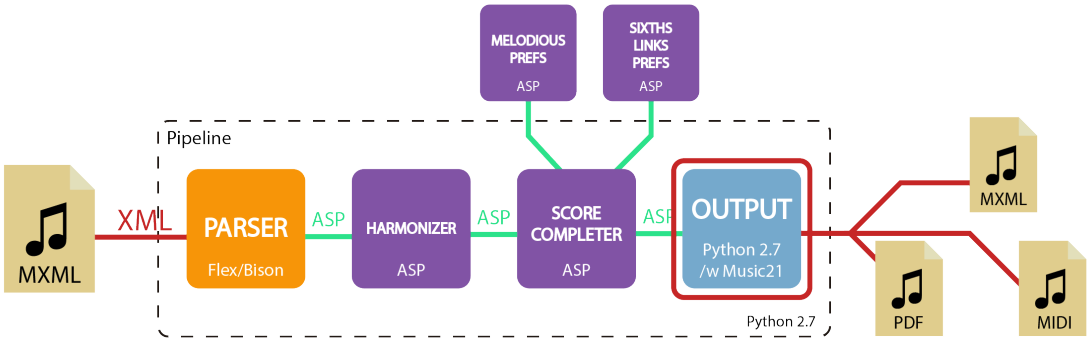
Parser

- 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



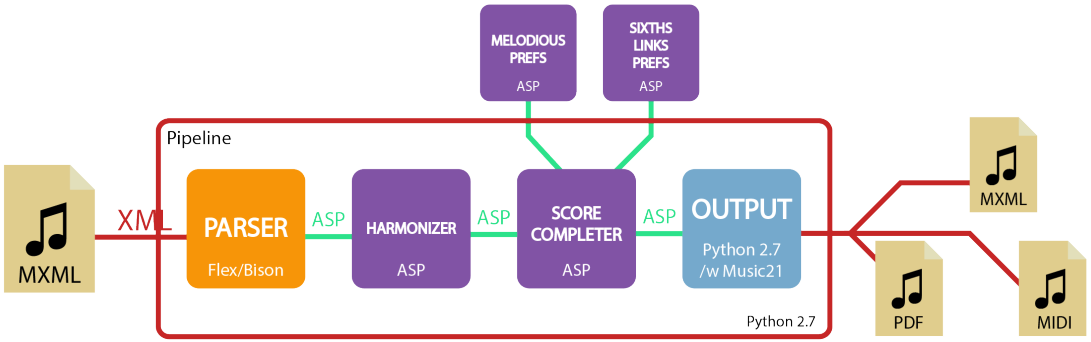
```
voice_type(1, violin).
figure(1,1,1).
note(1, 60, 1).
figure(1,1,2).
note(1, 67, 2).
measure(2, 0).
real_measure(2, 4, 0).
```

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

Pipeline

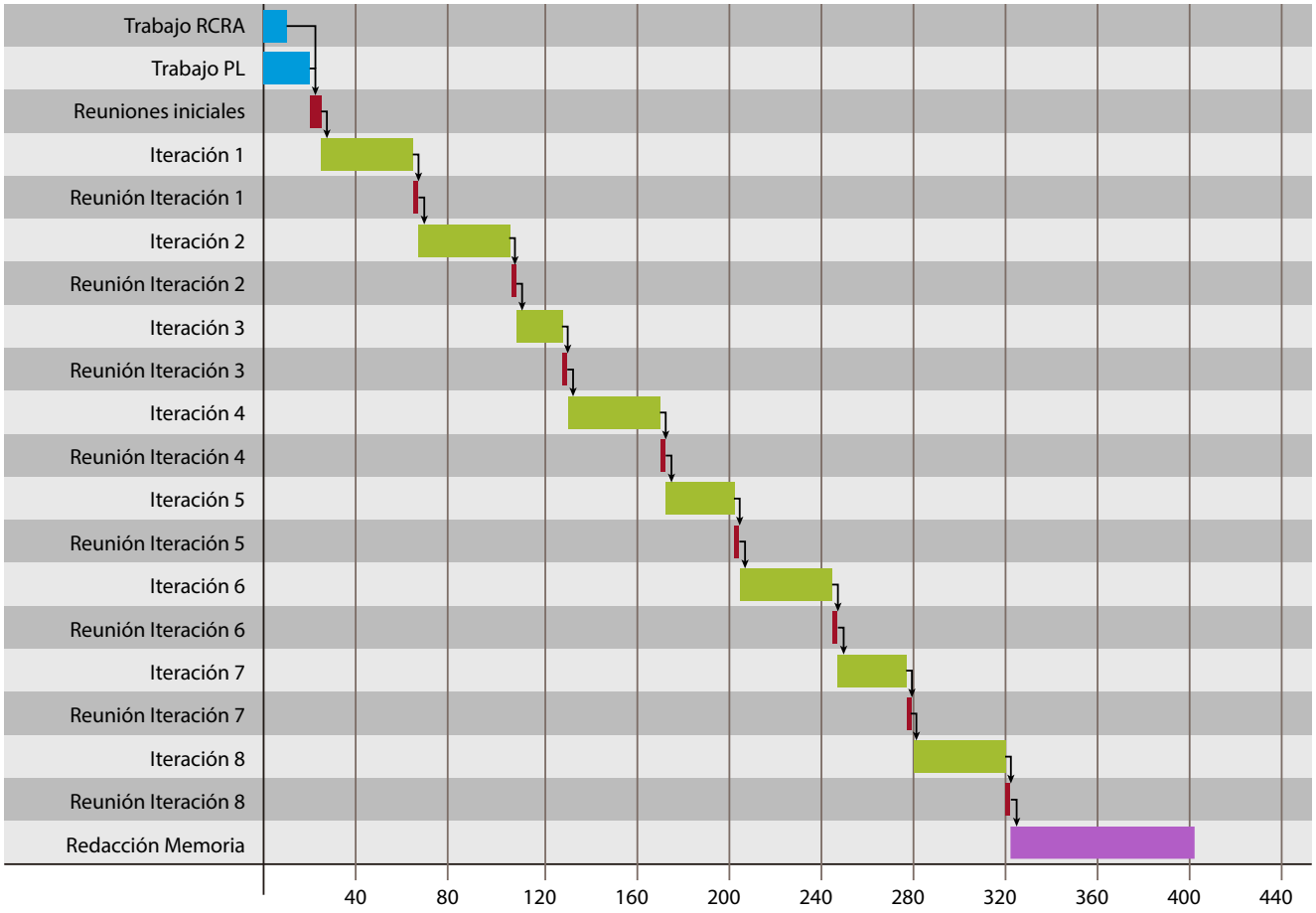


- Written in Python
- Coordinates the different modules secuentially
- Gives feedback to the user throught the command line
- Allows the user to pick the desired solution for harmonization and score completion
- Calls to the internal representation library to store the results of the Harmonization and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

Development Cycle

- Spiral development with prototypes
  - Traditional development cycles work better with pre-established architectures
  - ASP Development has huge exploration phases that are difficult to plan
- SCRUM
  - Agile development is more suited to the uncertain and evolutionary nature of ASP projects
  - SCRUM is meant for groups of developers coordinated by a SCRUM Master
- Custom development cicle
  - Each iteration revises previous works and evolves current prototype
  - Each iteration always has the same phases and these phases are planned beforehand
  - The work planned for each iteration is directed by the objectives
  - Very short iterations (1-2 weeks)
  - Allows objective redistribution for those that can't be achieved in one particular iteration
  - Prototypes are revised with the Director after each iteration

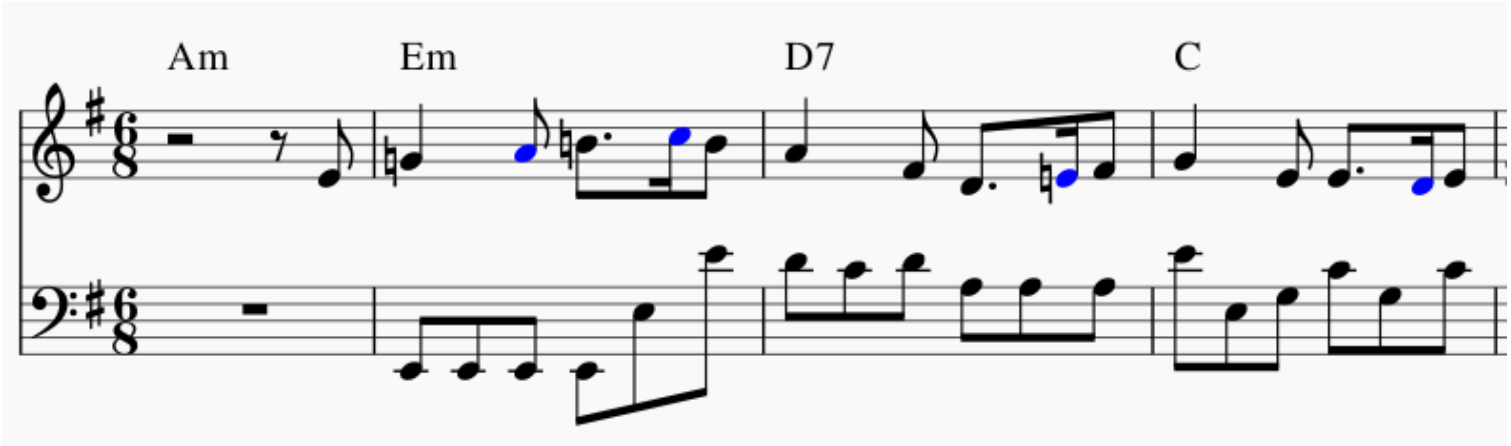
Iteration Breakdown and Costs



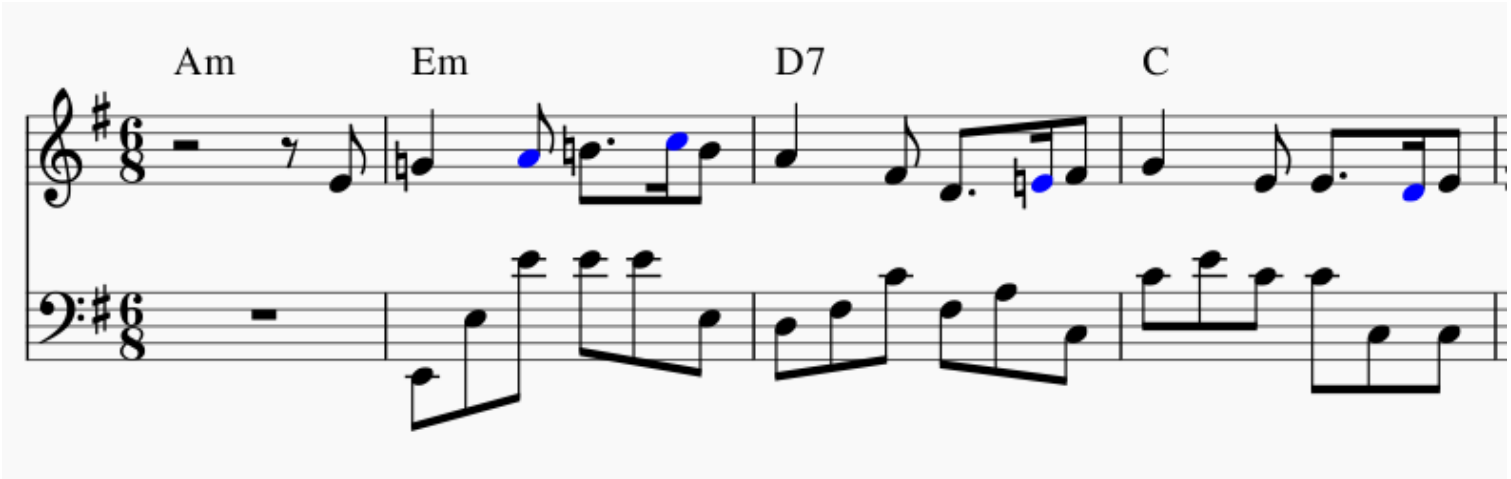
Profile	**Cost/Hour**	**Hours**	**Total**
Student	5.5€	362	1991€
Director	9€	12	108€
Total			2099€

Overview

Flexibility of results




```
:- ex_grade(V,1,B).
```

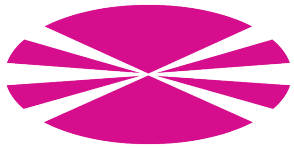


Conclusions

- Achieved maximum flexibility
- Accomplished 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

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



UNIVERSIDADE DA CORUÑA

Tool for musical harmonization through Answer Set Programming

Student:
Rodrigo Martín Prieto

Director:
José Pedro Cabalar Fernández

Thank you!

February 7, 2016