haspie – A Musical Harmonisation Tool based on ASP

Pedro Cabalar and Rodrigo Martín

Universidade da Coruña, Spain

August 8, 2018

Motivation

- Musical teaching is still very traditional nowadays.
- Self-teaching of music theory is hard.

Motivation

- Musical teaching is still very traditional nowadays.
- Self-teaching of music theory is hard.
- There are not many tools to aid and guide students and self-taught students.
- Composition tools seek results assuming that the user knows musical theory.
- There are intelligent composers: CHASP, Vox Populi, ANTON...

Example: Harmonisation

- Harmony is a very important subject in music theory learning
- Choral music is the root of this subject

Example: Harmonisation

- Harmony is a very important subject in music theory learning
- Choral music is the root of this subject
- Exercises consist in choosing chords sequences and completing musical pieces
- Already existing tools do not apply to this particular field

Goals

1. Harmonise and annotate chords over any musical score

Goals

1. Harmonise and annotate chords over any musical score
2. Given a certain harmonisation, be able to complete on purpose blank sections of any incomplete voice of the score

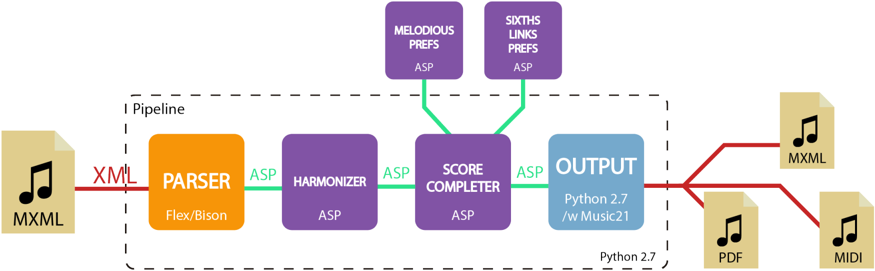
Goals

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

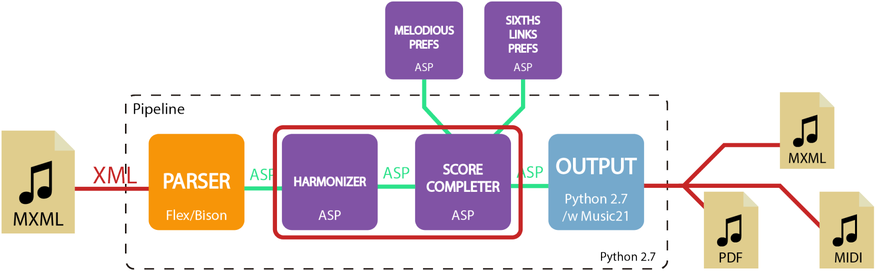
Overview

1. Motivation
2. haspie
    - Architecture
    - ASP Core
    - Input
    - Output
3. Conclusions & Future Work

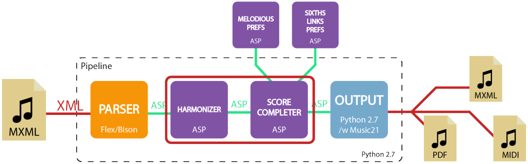
haspie's Architecture



haspie's Architecture



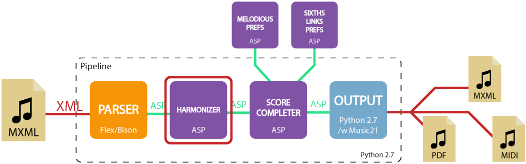
ASP Core



Answer Set Programming:

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

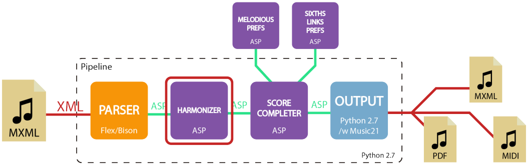
Harmonisation



- Notes are converted to grades of the scale given the key and mode

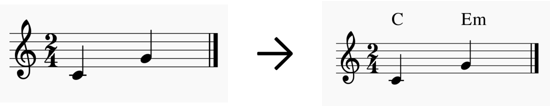
```
octave(V,((N - base) / 12),T) :- note(V,N,T), N >= 0.
sem_tones(V,((N - base) \ 12),T) :- note(V,N,T), N >= 0.
grade(V,1,T) :- sem_tones(V,3,T).
grade(V,2,T) :- sem_tones(V,5,T).
grade(V,3,T) :- sem_tones(V,7,T).
```

Harmonisation

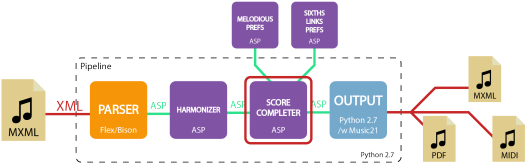


- Notes are converted to grades of the scale given the key and mode
- Chords are assigned to the harmonisable times of the score
- Errors are computed and the solver determines the fittest chords for each section

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

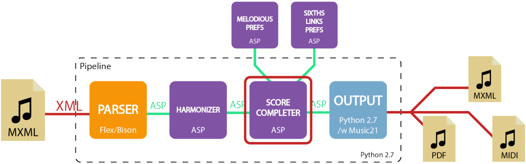


Score Completion



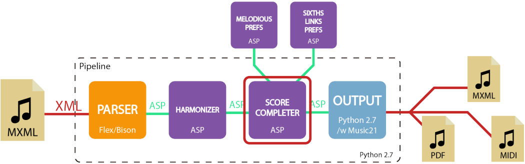
- Only used if there are new voices or sections that need to be completed

Score Completion

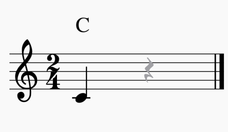


- Only used if there are new voices or sections that need to be completed
- Given the incomplete or new voices' *tessiturae* notes are assigned to the available positions

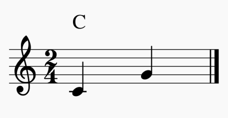
Score Completion



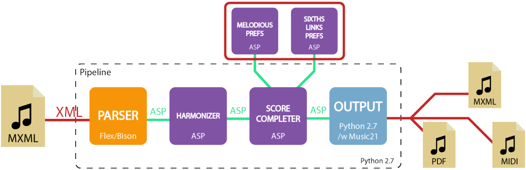
- Only used if there are new voices or sections that need to be completed
- Given the incomplete or new voices' *tessiturae* notes are assigned to the available positions
- Errors are computed and solver determines the fittest notes for each time



→

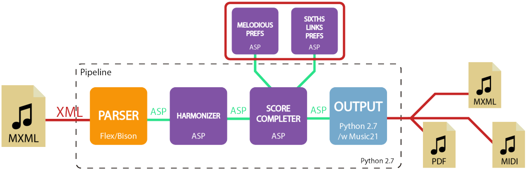


Melodious Preferences Modules



Despite not composing melodiously, haspie has modules that improve the melody

Melodious Preferences Modules



Despite not composing melodiously, haspie has modules that improve the melody

- Melodious Preferences:
  - Checks the tendency of the voices in the score and tries to imitate them
  - Reduces the melodic jumps between notes and the amount of repeated consecutive sounds
- Sixths Link:
  - Tries to find common progressions in choral music
  - If able, continues these common progressions of chords

User Configuration

ASP optimization:

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

User Configuration

ASP optimization:

- 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

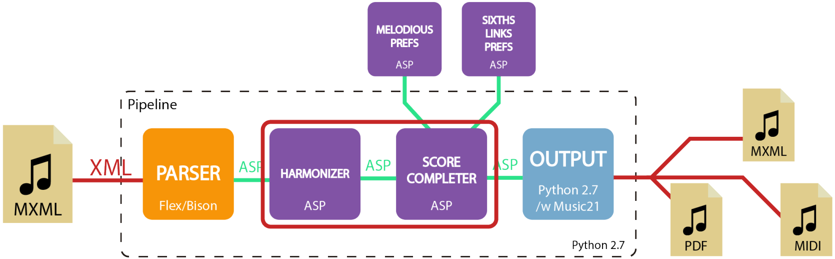
User Configuration

ASP optimization:

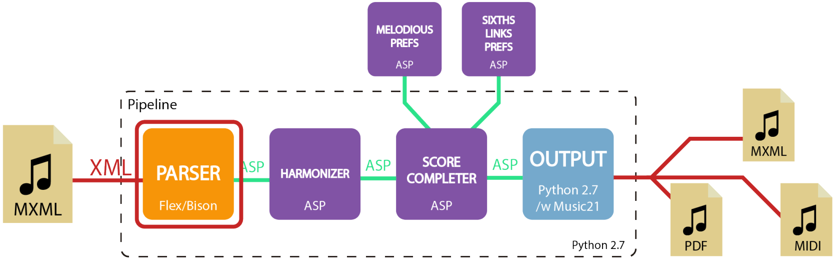
- The style of the resulting scores produced by the tool is determined by the optimization of many predicates
- These optimizations are weighted to be able to specify the significance of each of the measured predicates
- Users can define their own preferences by making use of configuration files

```
  #minimize[out_error(_,_) = chord_errorinstrongw
             @ chord_errorinstrongp].
  #minimize[same_chord(_,_) = chord_samechordw
             @ chord_samechordp].
  #minimize[out_error_weak(_,_) = chord_errorinweakw
             @ chord_errorinweakp].
```

haspie's Architecture



haspie's Architecture

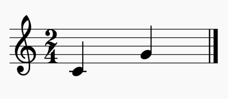


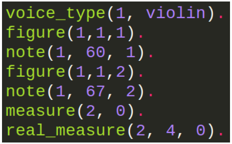
Parser and Preprocessor

- The project also included the development of a lightweight MusicXML 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

Parser and Preprocessor

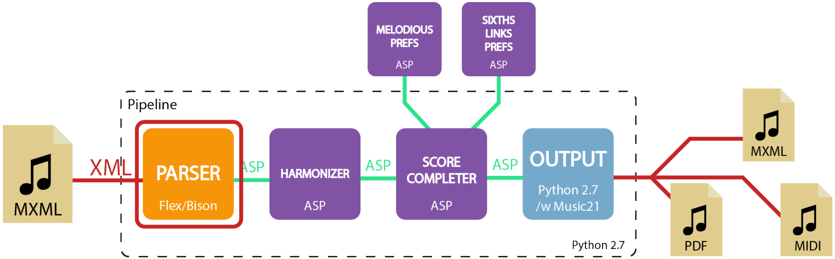
- The project also included the development of a lightweight MusicXML parser
- Written in C with the libraries Flex and Bison
- Transforms the score in MusicXML to the ASP logic facts that the ASP module uses later
- Performs various tasks as:
  - Subdivides notes to the length of the smallest figure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



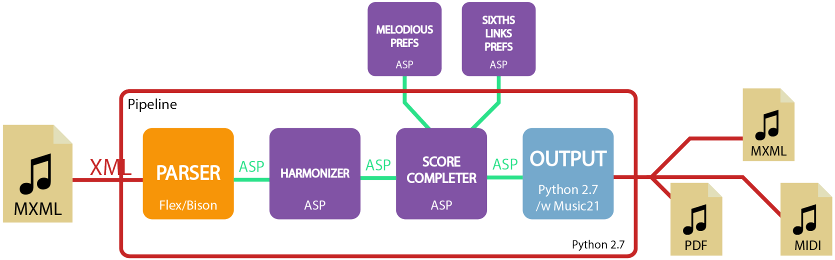


```
voice_type(1, violin).
figure(1,1,1).
note(1, 60, 1).
figure(1,1,2).
note(1, 67, 2).
measure(2, 0).
real_measure(2, 4, 0).
```

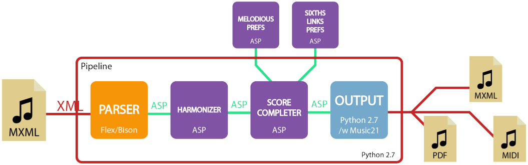
haspie's Architecture



haspie's Architecture



Pipeline & Output Module



- Written in Python with the toolkit Music21
- Gives feedback to the user and allows the selection of the desired solution
- Transforms the internal representation of the solution to a Music21 representation
- Some supported formats are Lilypond, PDF, Musescore, MusicXML or MIDI

Overview

1. Motivation

2. haspie

3. Conclusions & Future Work

Conclusions & Future Work

- About 200 ASP lines
- Good results in terms of harmony
- User still needs ASP knowledge to use it

Future Work:

- Research about modulation and implement it in the tool
- Reimplement preference-handling through asprin
- Improve the diversity of the solutions

PhD Work

"Efficient Generation of heterogeneous solutions to optimization problems in ASP"

- Takes off from the work developed for haspie
- Looking for better ways of representing preferences (i.e asprin)
- Measure distances between solutions to use them during optimization
- Use music as test ground through haspie

haspie – A Musical Harmonisation Tool based on ASP

Pedro Cabalar and Rodrigo Martín

Universidade da Coruña, Spain

August 8, 2018

Source available at github.com/trigork/haspie

Thank you!