Constraints-Based Music Harmonization

Maëlle Lise Colussi
LARA, 22 January 2015

Music Harmonization

- Add accompaniment to a melody

  -> add chords to be played at the same time as the melody

- Not all notes of the melody are necessarily « harmonized » :

  - some notes are played with a chord that makes sense for a previous or a subsequent note, but not for itself

  -> further I will call «the melody » the melody to which not harmonized notes were removed

Music Harmonization

- Several constraints for harmonization

  - Constraints on chords which will be played with each harmonized note (no dissonance)

  - Constraints on chord sequence (harmonic progression, cadences)

  - Constraints on note placement within a chord

- Note : cadences : constraints on chords at the end of a musical part

Music Harmonization

- Here we chose to use only the two first kind of constraints : gives the color to the melody

  -> link between chords and harmonized notes + chord sequence

- Constraints on note placement make transitions between chords softer, but we can do without (at least for a first try)

First Constraints

Link between chords and harmonized notes :

- Avoid dissonances between notes to harmonize and notes in chord
- Get a list of possible chords for each note : chords that contain this note (modulo octaves)
- Chords supported in the project for now (for harmonization): triads on notes without accidentals and seventh (4 notes chord) of dominant (V of scale)
- We have then 3 or 4 chords allowed for each note (if it has no accidentals, and is not a silent); silent notes have a possible «empty» chord (containing silences)

Second Constraints

- Constraints on chord sequence
- List of allowed pairs of adjacent chords and list of chords allowed for the end according to the chosen cadence, both extracted from classical harmony rules
- Also, if possible, some chords should not happen too often (because not very stable, for example the triad on degree VI of a scale, A-C-E in C scale)

  -> I will call these ones the «optional» constraints later

Linear Solver

- Solves problem of harmonization with both types of constraints, without the «optional » ones

- It begins from the end of the given melody, and goes step by step to the beginning

  - this is done this way to enforce constraints on cadences

- Silent notes are not considered from the harmony viewpoint : an «empty» chord is given; same for the other solver

Linear Solver

- To harmonize a note at the end :

  - get the allowed chords from the given cadence

  - take intersection with the possible chords of the note

  - if intersection non-empty : take randomly a chord in the resulting set

  - if intersection empty : take randomly a chord from the allowed chords

  -> cadence constraint is enforced, at the expense of possible dissonances

Linear Solver

- To harmonize a note that is not at the end

  - get allowed chords from subsequent note (harmonized before) and allowed pairs

  - take intersection with possible chords of the note

  - if intersection non-empty : take randomly a chord in the resulting set

  - if intersection empty : see next slide

Linear Solver

- To harmonize a note that is not at the end (continued)

  - if empty intersection:

    - If because no possible chords (note with accidental) : keep the one that was given just before to the subsequent note
    - Otherwise, take randomly a chord in the possible ones

  -> try to avoid dissonance at expense to harmony

- Each time the intersection is empty (also for last note) : a warning is given

Solving with SAT solver

- Reduction of the problem to solving formulas in conjunctive normal form (CNF)

  -> conjunction of clauses, each clause a disjunction of literals, literals are variables negated or not

- Then try to solve with a SAT solver; here CafeSat was used
- If a solution is found with basic constraints, try to solve with the «optional» constraints added

Reduction To SAT

- Each possible chord is associated with a boolean variable; at the end : cadence allowed chords are forced in the same way as for the linear solver

- Also, each possible harmonically allowed pair of chords for a particular pair of subsequent notes in the melody is associated with a boolean variable

Reduction To SAT (continued)

- We want that for each note exactly one chord is set to true, and for each pair of subsequent notes, only one allowed pair of chords is chosen
- We need constraints « exactly one » in a set of boolean variables (either representing possible chords for a note, or possible pair for two subsequent notes)
- No more than one : $not(a_i)||not(a_j)$

  for all a_i and a_j in the set, for $i! = j$ and $i < j$
- At least one : $a_0||a_1||...||a_{last}$

Reduction To SAT (continued)

- We need constraints to link a pair of two variables representing chords, with the variable that represent their pair (if it is an allowed one)

- Pair p true $<=>$ chords a and b true :

$not(p)||a$ and $not(p)||b$

and $p||not(a)||not(b)$

Composer Constraints

- Composer would want to impose set of possible chords for a note
- By default, what he/she wants is forced in the actual set of possible chords, then solving goes the same as before

  -> the linear solver will not necessarily respect all composer constraints (for the last notes), since it enforces cadence

  -> the composer must follow harmonic constraints, otherwise no solution from SAT solver

Composer Constraints

- An option exists in the project that can be used when some imposed sets from the composer contain only one element («strong constraints»): the derivation of chords is made part by part

  -> between two strongly imposed chords, the harmony rules are not enforced, composer can do what he/she wants

Conclusion

- We have two solvers

  - the linear one always give a solution, but it does not enforce all basic constraints

  - the one which uses SAT solver enforces all basic constraints when it can, and can sometimes enforce the optional ones, but it does not always give a solution

Conclusion (continued)

- Composer constraints can be specified
- Warnings from the linear solver can help choose differently the notes to harmonize, to make the problem solvable by the SAT solver, if this is wanted

  -> fusion some notes together, separate one note in two, add composer constraints

- Result : for simple melodies, sounds quite well

  -> some examples to be listened to

References

- Framework for music in Scala : from Valerian Pittet « MusicInterface », 2014 https://github.com/vtpittet/ScalaMusicGeneration
- CafeSat : from Régis Blanc, «CafeSat: A Modern SAT Solver for Scala», 2013 https://github.com/regb/scabolic
- Classical harmony : from Marjory Merriman, « The Music Theory Handbook », Thomson Schirmer, 1997.