# *Music Informatics*

Alan Smaill

Feb 15 2018

# Today

- Rule based systems
- Rule-based Counterpoint Systems
- Rule-based systems for 4-part harmonisation

# *Rule-based systems*

aka Knowledge-Based Systems.

This goes back to the idea of working with a declarative representation of the knowledge of some particular domain of interest (medical, geological, legal, musical . . . ).

It is standard to distinguish between:

- declarative knowledge: in the form of explicit statements about the object domain: “Paris is the capital of France”.
- procedural knowledge: knowing how to do something “how to tie shoe-laces”.

A rule based system for a particular domain will need to use both sorts of knowledge to solve problems.

## Declarative Knowledge

A standard answer is to use a logic-based representation
Translation from English is fairly easy.

- "All donkeys are stupid" gives
  $\forall x \; donkey(x) \rightarrow stupid(x)$
- "Fred is a donkey" gives
  $donkey(fred)$
- "Is Fred stupid?" gives
  $stupid(fred)$ ?

Logic gives an answer.
We can build up a knowledge base for a given domain in this sort of language – richer than database languages, usually, but along the same lines.

# *Procedural Knowledge*

This is harder to represent.
There is a problem about how to *use* declarative knowledge – eg that it is better to use one available rule that another to solve a particular problem.

Some possible answers:

- annotations to prioritise rules
- order of the data in Knowledge Base
- as a specialised control structure that analyses problems to determine the best route to a solution.

## Rules in music

Students are taught various rules for tackling some musical tasks, eg:

- counterpoint
- harmonisation

The rules do not lead to a unique answer (unlike rules for sudoku problems, which are set up to have a unique answer).
Also, the rules can be absolute, or describe preferences that can be broken.

There are a good number of rule-based systems that tackle these tasks; aside from being an interesting question in its own right, these systems can also be useful in a teaching context, to guide or critique students' work.

Rules from Fux's Gradus ad Parnassum on 2-part counterpoint (16th century) (following material taken from):

*http://homepage.eircom.net/~gerfmcc/SpeciesOne.html*

There are general rules, and rules specific to particular styles. Examples:

- Augmented or diminished intervals between succeeding notes are not allowed
- Leaps greater than an octave, & leaps of a major 6th or 7th are prohibited
- An ascending leap of a minor sixth or an octave must be followed by a step back down within the compass of the leap
- ...

# Specialised language

The rules use some definitions specific to this style (pre WTM):

- Perfect consonant intervals are unisons, fifths, and octaves.
- Imperfect consonant intervals are thirds and sixths.
- Seconds, fourths, sevenths, and all augmented and diminished intervals are dissonances.
- When the two voices move in the same direction, movement direct.
- When the two voices move in different directions, movement is contrary

general rule:
The two parts may not move in direct motion to a perfect consonance.

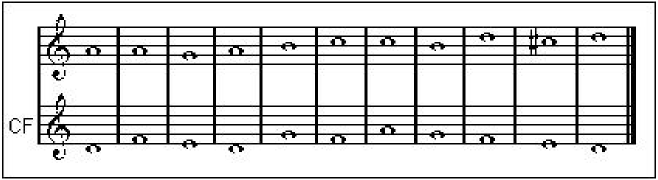
# First species rules

- The counterpoint consists of a single semibreve against each note of the cantus firmus.
- No dissonances are allowed.
- In the penultimate bar the counterpoint must be a major sixth above the cantus firmus. This may require an accidental.
- In the final bar the counterpoint must be an octave above the cantus firmus.
- Unisons are not allowed, except in the first bar.
- The counterpoint in the first bar must be an octave or a fifth above the cantus firmus, or a unison

## The task

Given a "cantus firmus", produce a counterpoint that obeys the rules.

For example, lower line given, upper to be written:




Above is not the full set of rules, and there are other rules about the range of the upper voice for example.

The link gives an applet to check if a counterpoint obeys all the rules.

# Comments

There are other musical considerations than getting the counterpoint right according to the rules, but this is a good musical exercise.

What about generating counterpoint automatically?

# Generating counterpoint

Some possibilities:

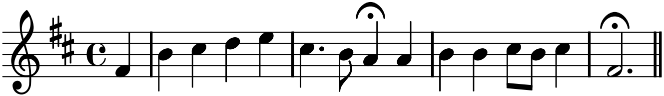
- generate 11 notes (randomly?) in the right range and see if the rules are OK; this "generate and test" is inefficient, and not how humans do this.
- generate notes incrementally from start, checking rules as much as possible at each step; this is computationally better, & more human-like;
  – might need to backtrack.
- incrementally from the end backwards.
- mixture of forward and backward.
- ...

## Comments ctd

We see the distinction again between the declarative knowledge (the rules of 1st species counterpoint), and the procedural knowledge of how to solve a given problem, while respecting the rules.

Different solution strategies give different ways to generate the counterpoint.
An important feature of a rule-based system here is that the rules remain the same when used with a different control regime:
this is an advantage of the declarative approach.

Here the space of possibilities is much larger. Given a melody line, the task is to supply 3 lower voices that provide a harmonisation in the style use by Bach in his Chorales.
Given:



# Bach's answer

This is a challenging example!

# Rules for harmonising

When this is taught, there are standard sorts of guidance that is given, as well as ideas in the best order in which to carry out a harmonisation.

Turning all of these into a set of rules for a rule-based system involves a serious amount of work on the part of anyone who wants to build such a system.

A system that carries out this task is by Somnuk Phon-Amnuaisuk, Journal of New Music Research, 2006.

`http://www.tandfonline.com/doi/full/10.1080/09298210701458835`

The following is based on that article, and other work by the first author; the work makes much use of earlier work by Ebcioglu.

# Rules

The task is organised around successive elaboration of the final output, in stages:

1. Analyse the input melody
2. Outline each phrase with a harmonic plan
3. Sketch outline voices
4. Fill in the actual notes and other detail.

The rules allow a partial solution to be extended or altered; they may lead down a blind alley, where there is no good solution, and then allow backtracking.

## *Steps 1 & 2*

Input analysis is very simple – just split into phrases, and indicate basic rhythm (this information is obvious from the presentation of the input, which has to be put into a declarative form).

For chord assignment, melody has to fit with chosen chord; a cadence is needed at the phrase end, and there is a set of possible cadence types; also start of phrase is treated differently.
Three version of this rule, with many possible outcomes:

```
outlineChord(intro);
outlineChord(body);
outlineChord(cadence);
```

These can be used in different orders.

# Steps 3: outline voices

To look for possible bass lines, given soprano and chords:

```
outlineBass(intro);
outlineBass(cadence);
outlineBass(body)
```

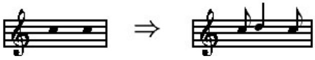
Inner voices:

```
outlineInnerVoices
```

# Stage 4: elaborate voices

16 different ways of filling in from basic chords to decorated versions, eg:

`fill(neighbourSuspension)`



In a normal state of any voice, when two outline pitches form a unison, fill an upper neighbour note and change the voice state to the suspension state.

Alongside rules for generating candidate partial solutions, use

- tests – constraints that must be satisfied
- measures – giving preferences according to various criteria

These will be used to guide the search for good solutions.

# *Tests*

- `constrain(doubleLeadingNote)`
  No doubling of a leading note allowed.
- `constrain(skipLeadingNote)`
  If a leading note does not move to a tonic note (between a crotchet beat), then it is forbidden to decorate between the two notes with a skip quaver.
- `constrain(cadenceLeadingNote)`
  In a perfect cadence, if the leading pitch moves to the dominant pitch, then it should not be decorated with a passing note. In the vii$^{o}$-I perfect cadence pattern, the bass should not be decorated with the root of the leading chord before the final tonic bass
- etc etc

# Measurements

Use these to indicate preferences. Examples:

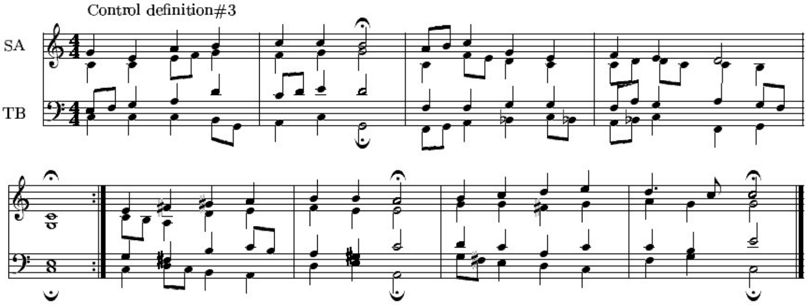
- `property(preferredRhythmicPattern(bass))`
  The pattern quaver/quaver/crotchet
  (8th note/8th note/quarter note) is undesirable, if the quaver/quaver starts on the strong beat of a bar.
- `property(preferredUnisonOrnamentation)`
  Unison in a weak quaver is undesirable.
- `property(preferredDissonantSuspension)`
  Dissonant second, fourth or seventh with the bass voice is desirable
- Others, eg on spacing of inner voices.

Can configure how the rules are used by building control definitions.

```
definition(outlineHarmonicProgression,
    repeat
        ( rule:selectPhrase(outlineHarmonicPlan) then
          filter outlinePhraseHarmonicPlan
          with test:constrain(harmonicPlanOutline)) ).
```

# Example

R175 Jesus, meine Zuversicht

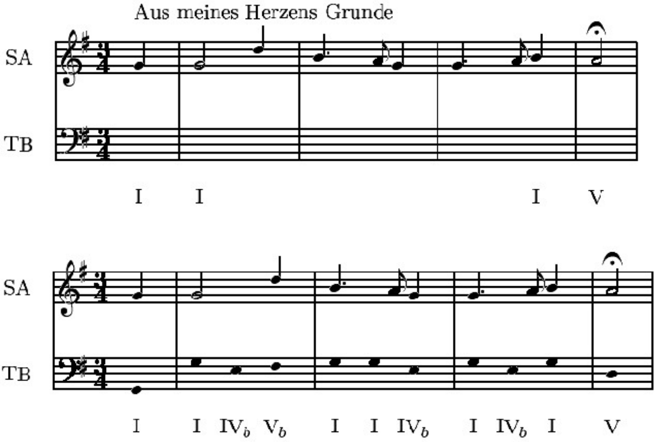


# Comments

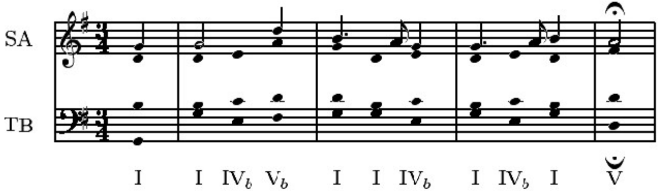
This shows that good quality output can be achieved by machine for this particular task. This approach

- takes a large amount of work on the part of the specifier of the rules, tests, etc, which is all done by hand
- makes the steps in finding harmonisation explicit
- gives an explanation of how the result was obtained
- potentially useful in teaching (what possibilities for next step?)

# Stages 1, 2


Aus meines Herzens Grunde
SA
TB
I I I V
SA
TB
I I IV$_b$ V$_b$ I I IV$_b$ I IV$_b$ I V

# Stages 3, 4




# Stage 5

## *Other approaches*

Building such a system takes a lot of effort and musical expertise.
Other approaches involve machine learning (ML), and/or combination of ML and rule-based for different aspects of the problem.
For ML approach to harmony, see demo at:

`http://www.anc.inf.ed.ac.uk/demos/hmmbach/`

and associated documents:

`https://tardis.ed.ac.uk/~moray/harmony/`

(You will probably have to download midi files to listen to them.)

For Bach chorales, there are machine-readable, analysed versions of the scores available.
Still, a problem here is the sparseness of data! (compared to natural language corpora)

## *Other approaches ctd*

A hybrid system: ML (for harmony) and rule-based (for laying out the individual voices):

"HARMONET: A Neural Net for Harmonizing Chorales in the Style of J. S. Bach", Hild, Feulner and Menzel, Advances in Neural Information Processing Systems 4 (NIPS 1991)

http://preview.tinyurl.com/gr3n5hq

# Harmonet ctd

This is a connectionist system, which works at 3 levels:

- Harmonic skeleton (ML);
- Chord skeleton (rule-based);
- Voice ornamentation (ML).

This produces more elaborate harmonisations than in Allan & Williams.

# Summary

- declarative rule-based systems
- control in search for solutions to problems
- musical examples (counterpoint, harmonisation)