



# *Music Informatics*

Alan Smaill

Feb 11 2016

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

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.

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

- ▶ “All donkeys are stupid” gives  
 $\forall x \text{ donkey}(x) \rightarrow \text{stupid}(x)$
- ▶ “Fred is a donkey” gives  
 $\text{donkey}(\text{fred})$
- ▶ “Is Fred stupid?” gives  
 $\text{stupid}(\text{fred}) ?$

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.

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 than 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.

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
- ▶ ...

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.

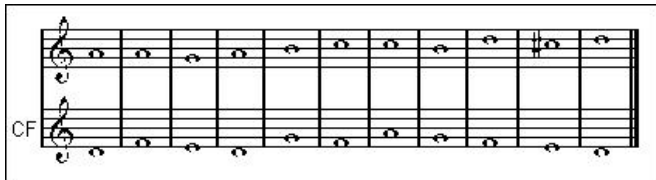


- ▶ The counterpoint consists of a single semibreve against each note of the cantus firmus.
- ▶ No dissonances are allowed.
- ▶ In the penultimate bar the counterpoint must be a major sixth above the cantus firmus. This may require an accidental.
- ▶ In the final bar the counterpoint must be an octave 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:



The image shows a musical score with two staves. The lower staff is labeled 'CF' and contains a sequence of notes: C4, D4, E4, F4, G4, A4, B4, C5, D5, E5, F5, G5, A5, B5, C6. The upper staff is empty, intended for the student to write a counterpoint.

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.

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?

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.
- ▶ ...

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:





This is a challenging example!

**364.** **Von Gott will ich nicht lassen**

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://tinyurl.com/yfjqdpm>

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



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.

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 3,4

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

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

Inner voices:

```
outlineInnerVoices
```

And 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) and **measures** (giving preferences according to various criteria).

Example tests:

▶ `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  $\text{vii}^{\circ}-\text{I}$  perfect cadence pattern, the bass should not be decorated with the root of the leading chord before the final tonic bass

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

Control definition#3



The image displays a musical score for the hymn 'Jesus, meine Zuversicht' (R175). It features two vocal parts, Soprano (SA) and Tenor (TB), and a piano accompaniment. The score is written in 4/4 time and consists of two systems of staves. The first system shows the vocal lines and the beginning of the piano accompaniment. The second system continues the vocal lines and piano accompaniment, ending with a double bar line. The piano accompaniment is written in treble and bass clefs, with a key signature of one flat (B-flat).

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

Musical notation for Tenor (TB) part of 'Aus meines Herzens Grunde'.

I I I V

SA

TB

Musical notation for Tenor (TB) part of 'Aus meines Herzens Grunde'.

I I IV<sub>b</sub> V<sub>b</sub> I I IV<sub>b</sub> I IV<sub>b</sub> I V

# Stages 3, 4



SA

TB

I I IV<sub>b</sub> V<sub>b</sub> I I IV<sub>b</sub> I IV<sub>b</sub> I V

SA

TB

## Stage 5



SA

TB

A musical score for Soprano (SA) and Tenor (TB) in 4/4 time. The key signature has one sharp (F#). The SA part is written on a treble clef staff, and the TB part is written on a bass clef staff. The music consists of a series of chords and melodic lines. The SA part starts with a half note G4, followed by a quarter note A4, a quarter note B4, and a quarter note C5. The TB part starts with a half note G3, followed by a quarter note A3, a quarter note B3, and a quarter note C4. The music continues with similar patterns, ending with a final chord of G4, B4, and C5.

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/>

A problem here is the sparseness of data!

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>

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.

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