Automatic Music Composition

using Answer Set Programming

Marvin Beese

University of Potsdam

17. December 2019

Outline

Introduction

- composition through the use of rules
- computer aid for compositions:
  - automation
  - analysis
  - verification
- ANTON
  - composition of melodic, harmonic and rhythmic music
  - provides diagnose errors
  - serves as a computer-aided composition tool

Musical Background

- musical rules of Renaissance Counterpoint used for
  - composition: creating new musical pieces
  - turning a monody into a melody
  - harmonisation: accompaniment of a melody
- stages of compositions
  - creation of melody sections
  - harmonisation
  - structuring the piece around these sections

Automatic Composition

- improvisation systems vs composition systems
- focus on composition systems
  - requires predictions of succeeding notes
  - decisions in any order
  - time-independency of the decisions
  - simultaneous generation of melody and harmony

Automatic Composition

- melodic composition
  - probabilistic or learning strategies as a common approach
  - constraint programming systems
    - ★ generation of melodies with few rules
    - ★ improvement through additional rules

Automatic Composition

- melodic composition
  - probabilistic or learning strategies as a common approach
  - constraint programming systems
    - ★ generation of melodies with few rules
    - ★ improvement through additional rules
- harmonic composition
  - adding harmonic lines to a melody as a common approach
  - generation of melody and harmony together
    - ★ harmonisation effects the melody

Anton Composition System

- *AnsProlog* for representation and reasoning about compositional rules
- description of rules governing melodic and harmonic properties

Anton Composition System

- *AnsProlog* for representation and reasoning about compositional rules
- description of rules governing melodic and harmonic properties

- various files for the basic system:
  - *progression.lp*: progressions and selection of the next note
  - *melodic.lp*: rules for melodic parts
  - *harmonic.lp*: composing with multiple parts
  - further parts: *chord.lp*, *notes.lp*, *modes.lp*

Anton Composition System: Progressions

- defined over time step T
- key proposition: `chosenNote(P,T,N)`
  - part P plays note N at time T

```
time(1..t).
```

Anton Composition System: Progressions

- defined over time step T
- key proposition: `chosenNote(P,T,N)`
  - part P plays note N at time T

```
time(1..t).
```

- maximum change of one note per time step allowed

```
:- 2 { chosenNote(P,T,NN) : note(NN), rest(P,T) }.

1 { changes(P,T), repeated(P,T), toRest(P,T), fromRest(P,T),
  incorrectProgression(P,T) } 1 :- T != t.
1 { stepAt(P,T), leapAt(P,T) } 1 :- changes(P,T), T != t.
1 { downAt(P,T), upAt(P,T) } 1 :- changes(P,T), T != t.

stepDown(P,T) :- stepAt(P,T), downAt(P,T).
stepUp(P,T) :- stepAt(P,T), upAt(P,T).
```

Anton Composition System: Progressions

- pick a step amount

```
1 { stepBy(P,T,SS) : stepSize(SS) : SS < 0 } 1 :- stepDown(P,T).
1 { stepBy(P,T,SS) : stepSize(SS) : SS > 0 } 1 :- stepUp(P,T).
```

Anton Composition System: Progressions

- pick a step amount

```
1 { stepBy(P,T,SS) : stepSize(SS) : SS < 0 } 1 :- stepDown(P,T).
1 { stepBy(P,T,SS) : stepSize(SS) : SS > 0 } 1 :- stepUp(P,T).
```

- error rules for diagnosis and composition

```
#const err_ip="Incorrect progression".
reason(err_ip).
error(P,T,err_ip) :- incorrectProgression(P,T).
```

Anton Composition System: Progressions

- pick a step amount

```
1 { stepBy(P,T,SS) : stepSize(SS) : SS < 0 } 1 :- stepDown(P,T).
1 { stepBy(P,T,SS) : stepSize(SS) : SS > 0 } 1 :- stepUp(P,T).
```

- error rules for diagnosis and composition

```
#const err_ip="Incorrect progression".
reason(err_ip).
error(P,T,err_ip) :- incorrectProgression(P,T).
```

- definition of the chosen note

```
chosenNote(P,T + 1,N + S) :- chosenNote(P,T,N), stepAt(P,T),
                     stepBy(P,T,S), note(N + S).
chosenNote(P,T + 1,N + L) :- chosenNote(P,T,N), leapAt(P,T),
                     leapBy(P,T,L), note(N + L).
chosenNote(P,T + 1,N) :- chosenNote(P,T,N), repeated(P,T).
```

Anton Composition System: Melodies

- repetition of single notes is prohibited

```
#const err_nrmp="No repeated notes in melodic parts".
reason(err_nrmp).
error(MP,T,err_nrmp) :- repeated(MP,T).
```

Anton Composition System: Melodies

- repetition of single notes is prohibited

```
#const err_nrmp="No repeated notes in melodic parts".
reason(err_nrmp).
error(MP,T,err_nrmp) :- repeated(MP,T).
```

- octave leaps are not allowed

```
#const err_olnf="Leap of an octave from a note other than the fundamental".
reason(err_olnf).
error(MP,T,err_olnf) :- leapBy(MP,T,12), not chosenChromatic(MP,T,1).
error(MP,T,err_olnf) :- leapBy(MP,T,-12), not chosenChromatic(MP,T,1).
```

Anton Composition System: Melodies

- stepwise linear progression leads to an impulse

```
downwardImpulse(MP,T+1) :- leapDown(MP,T), time(T+1).
downwardImpulse(MP,T+3) :- stepDown(MP,T+2), stepDown(MP,T+1), stepDown(MP,T), time(T+3).
upwardImpulse(MP,T+1) :- leapUp(MP,T), time(T+1).
upwardImpulse(MP,T+3) :- stepUp(MP,T+2), stepUp(MP,T+1), stepUp(MP,T), time(T+3).
```

Anton Composition System: Melodies

- stepwise linear progression leads to an impulse

```
downwardImpulse(MP,T+1) :- leapDown(MP,T), time(T+1).
downwardImpulse(MP,T+3) :- stepDown(MP,T+2), stepDown(MP,T+1), stepDown(MP,T), time(T+3).
upwardImpulse(MP,T+1) :- leapUp(MP,T), time(T+1).
upwardImpulse(MP,T+3) :- stepUp(MP,T+2), stepUp(MP,T+1), stepUp(MP,T), time(T+3).
```

- dissonant contours and repetitions of multiple notes are not allowed

```
#const err_rn="Repeated notes".
reason(err_rn).
error(MP,T1,err_rn) :- chosenNote(MP,T1,N), stepBy(MP,T1,S1),
            chosenNote(MP,T2,N), stepBy(MP,T2,S1),
            T1 + 1 < T2, T2 < T1 + 2 + RW.
error(MP,T1,err_rn) :- chosenNote(MP,T1,N), leapBy(MP,T1,L1),
            chosenNote(MP,T2,N), leapBy(MP,T2,L1),
            T1 + 1 < T2, T2 < T1 + 2 + RW.
#const err_dc="Dissonant contour".
reason(err_dc).
error(MP,t,err_dc) :- lowestNote(MP,N1), highestNote(MP,N2),
            chromatic(N1,C1), chromatic(N2,C2),
            not consonant(C1,C2), N1 < N2.
```

Anton Composition System: Harmonies

- dissonant intervals between parts are not allowed

```
#const err_dibp="Dissonant interval between parts".
reason(err_dibp).
error(P1,T,err_dibp) :- chosenChromatic(P1,T,C1), chosenChromatic(P2,T,C2),
              P1 < P2, chromaticInterval(C1,C2,D),
              not validInterval(D).
```

Anton Composition System: Harmonies

- dissonant intervals between parts are not allowed

```
#const err_dibp="Dissonant interval between parts".
reason(err_dibp).
error(P1,T,err_dibp) :- chosenChromatic(P1,T,C1), chosenChromatic(P2,T,C2),
            P1 < P2, chromaticInterval(C1,C2,D),
            not validInterval(D).
```

- limitation for distances between parts
  - Octave + 4 Semitones = 16 Semitones

```
#const err_mdbp="Over maximum distance between parts".
reason(err_mdbp).
error(P,T,err_mdbp) :- chosenNote(P,T,N1), chosenNote(P+1,T,N2),
            N1 > N2 + 16, part(P+1).
```

Anton Composition System: Specification

- description for a quartet with 4 parts

```
style(quartet).
part(1..4).
```

Anton Composition System: Specification

- description for a quartet with 4 parts

```
style(quartet).
part(1..4).
```

- definition of the melodic part and of the lowest part

```
melodicPart(1).
lowestPart(4).
```

Anton Composition System: Specification

- definition for the range of the parts

```
#const quartetBottomNote=1.
#const quartetTopNote=68.
note(quartetBottomNote..quartetTopNote).
bottomNote(quartetBottomNote).
topNote(quartetTopNote).
```

Anton Composition System: Specification

- definition for the range of the parts

```
#const quartetBottomNote=1.
#const quartetTopNote=68.
note(quartetBottomNote..quartetTopNote).
bottomNote(quartetBottomNote).
topNote(quartetTopNote).
```

- starting positions are tonics and dominants

```
#const err_isn="Incorrect starting note".
reason(err_isn). error(1,1,err_isn) :- not chosenNote(1,1,44).
error(2,1,err_isn) :- not chosenNote(2,1,37).
error(3,1,err_isn) :- not chosenNote(3,1,32).
error(4,1,err_isn) :- not chosenNote(4,1,25).
```

Anton Composition System: Specification

- rests are not allowed

```
#const err_nrfw="No rest for the wicked".
reason(err_nrfw).
error(P,T,err_nrfw) :- rest(P,T).
```

Anton Composition System: Specification

- rests are not allowed

```
#const err_nrfw="No rest for the wicked".
reason(err_nrfw).
error(P,T,err_nrfw) :- rest(P,T).
```

- intervals of a major fourth between parts allowed with three or more parts

```
validInterval(5).
```

1. Introduction
2. Musical Background
3. Automatic Composition
4. Anton Composition System
   - Progressions
   - Melodies
   - Harmonies
   - Specification
5. Conclusion

Conclusion

- algorithmic composing system for melodic and harmonic composition
  - generation in an appropriate time frame
- simultaneous availability of all rules
  - evaluation of rule compliance
  - completion of partial systems
  - creating new melodies
  - producing melodies to given harmonies

Sources

- Georg Boenn, Martin Brain, Marina De Vos, John P. Fitch: Automatic music composition using answer set programming. TPLP 11(2-3): 397-427 (2011)

Thank you for your attention!