Demo: Counterpoint by Construction

Youyou Cong and John Leo

Composition Rules and Typing Rules

- Composition rules make music sound good
- Typing rules make programs meaningful

Well-typed music does not sound wrong
(Szamozvancev & Gale, Haskell '17)

Dependent Types as Precise Specification

`v : Vec ℕ n` means "v has n natural numbers"

```
nth : ∀ m. Vec ℕ m → {n : ℕ | n ≤ m} → ℕ
nth v n = ...
```

`nth [1] 1` ✓ `nth [1] 2` ✗

Dependent Types for Music Rules

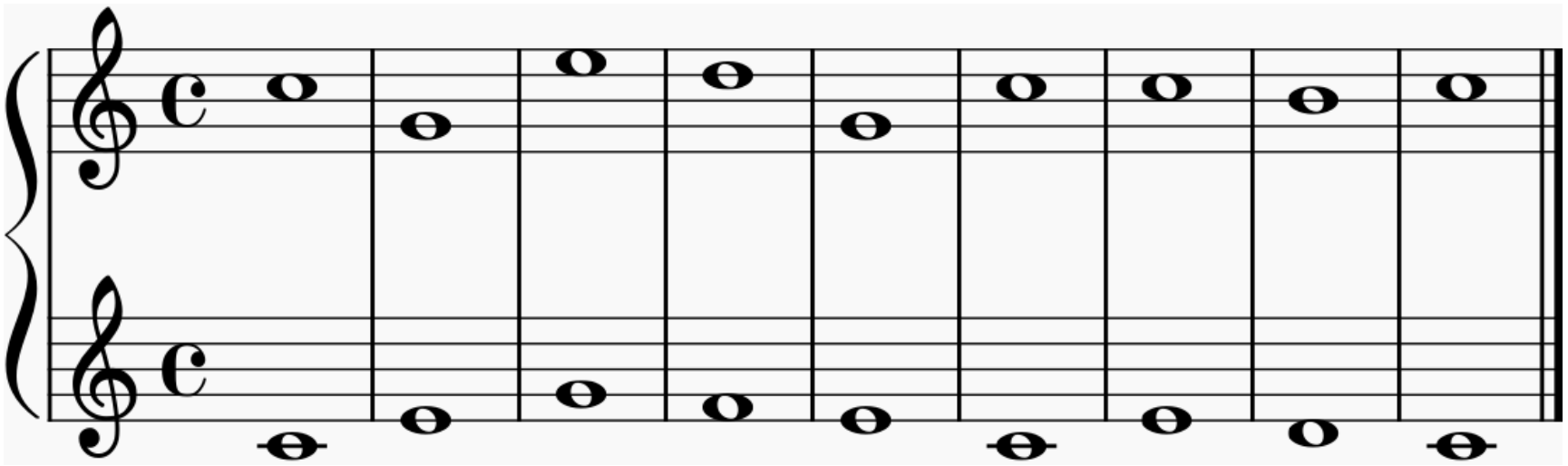
- Length-indexed vector type
  (for composition of parallel voices)
- Mode-indexed datatypes
  (for composition of phrases)

This Work

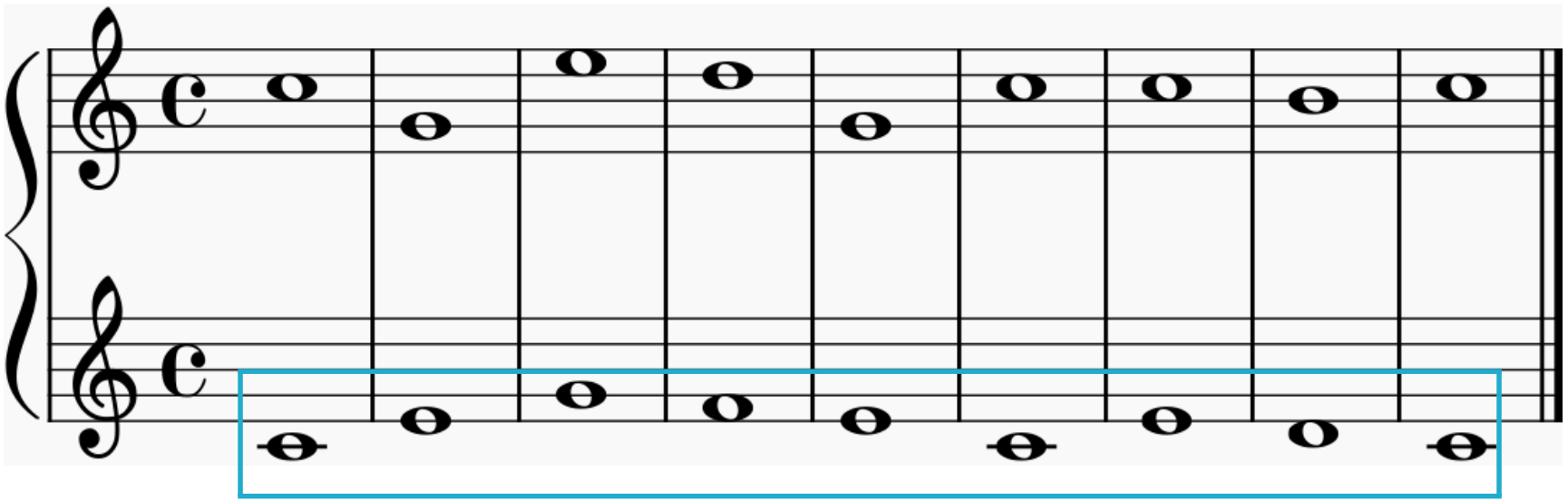
Dependently Typed Counterpoint
on **Music Tools**

- Agda library for music composition
- Use dependent types to encode rules
- Use Haskell FFI to generate MIDI

Counterpoint: Interaction of Multiple Voices



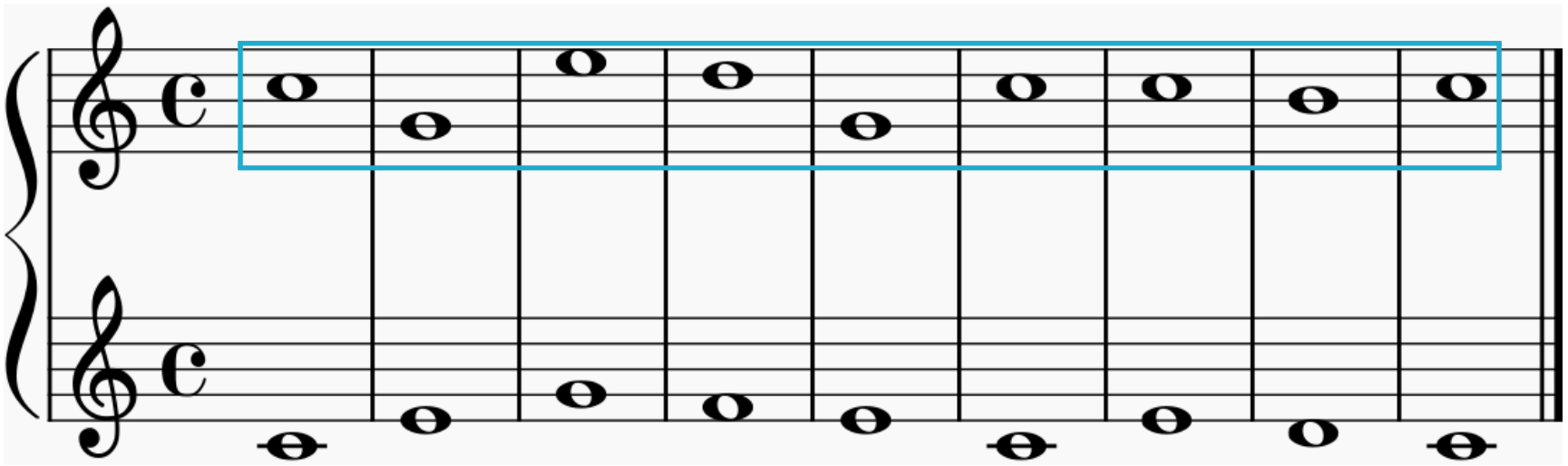
Counterpoint: Interaction of Multiple Voices



cantus firmus

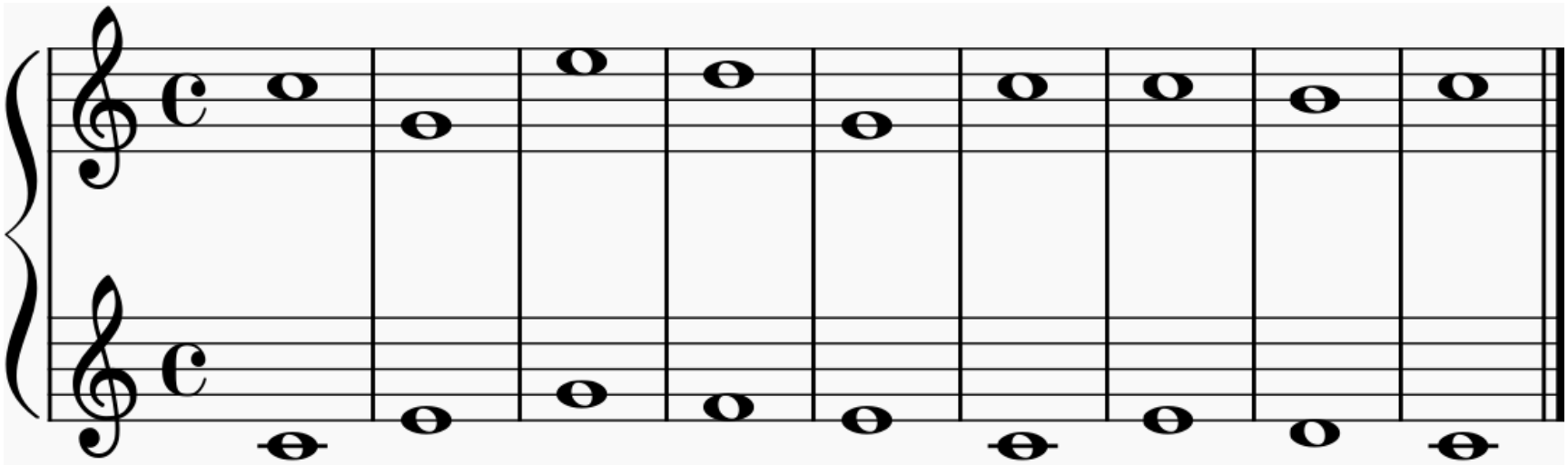
Counterpoint: Interaction of Multiple Voices

counterpoint



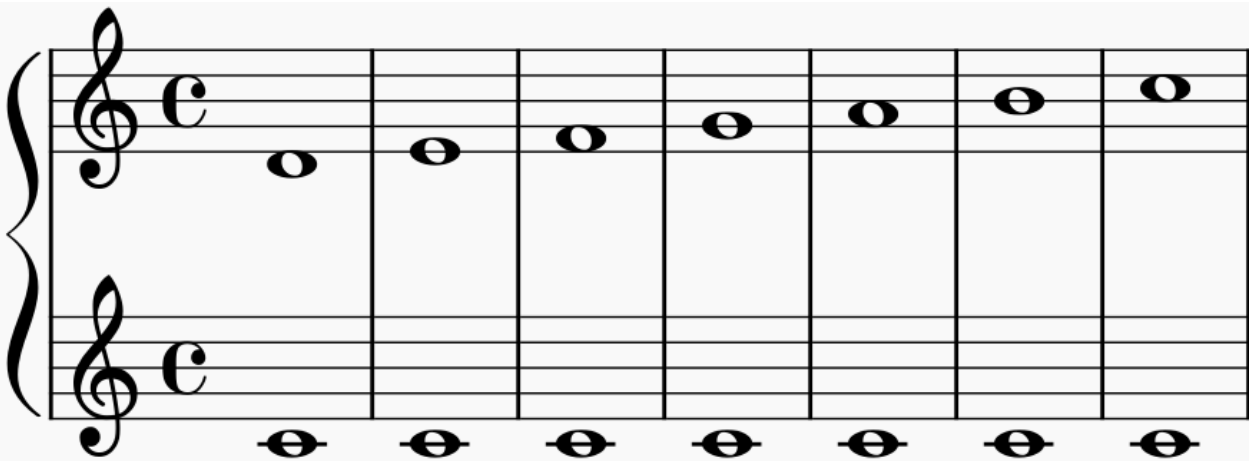
Key Concept 1: Intervals

per8 min3 maj6 maj6 min3 per8 maj6 maj6 per8



Rule 1 for First-Species Counterpoint

All intervals must be ***consonant***



2nd	3rd	4th	5th	6th	7th	8th
✗	✓	✗	✓	✓	✗	✓

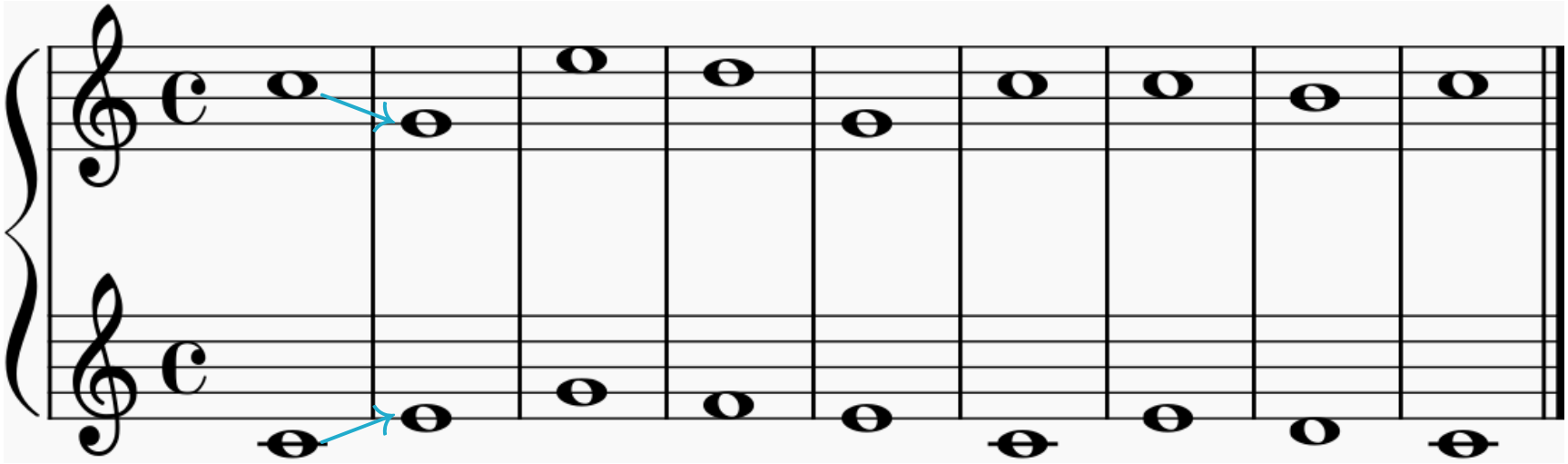
Representing Valid Intervals

```
data IntervalQuality : Set where
  min3  : IntervalQuality
  maj3  : IntervalQuality
  per5  : IntervalQuality
  min6  : IntervalQuality
  maj6  : IntervalQuality
  per8  : IntervalQuality
  min10 : IntervalQuality
  maj10 : IntervalQuality

PitchInterval : Set
PitchInterval = Pitch × IntervalQuality
```

Key Concept 2: Motion

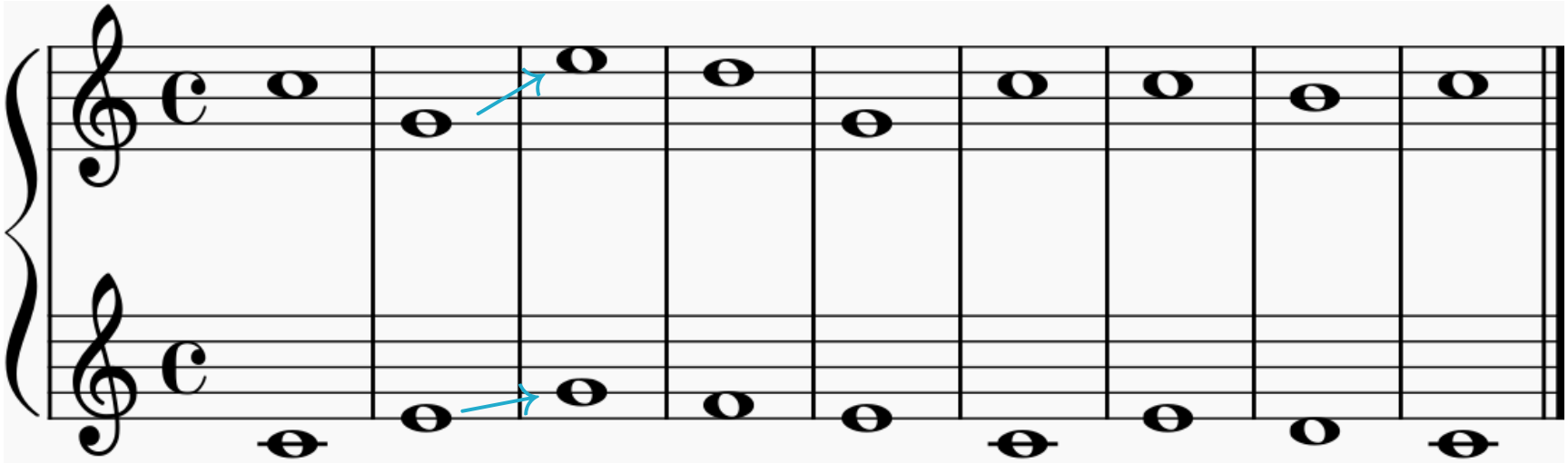
ctr sim par sim ctr obl par ctr



where ctr: contrary, sim: similar, par: parallel, obl: oblique

Key Concept 2: Motion

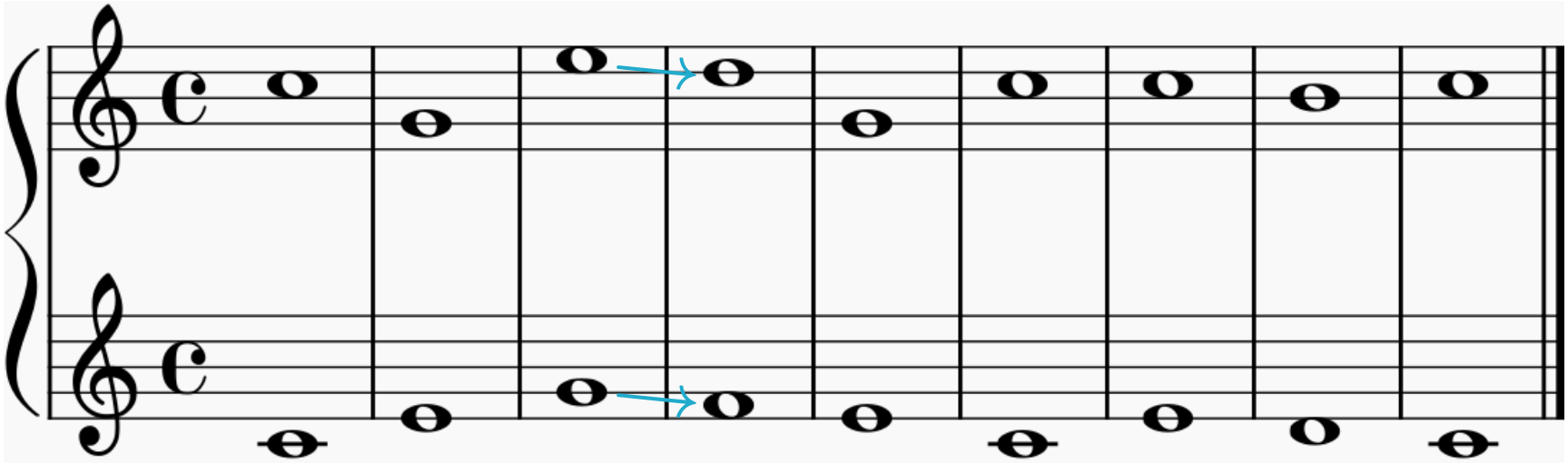
ctr sim par sim ctr obl par ctr



where ctr: contrary, sim: similar, par: parallel, obl: oblique

Key Concept 2: Motion

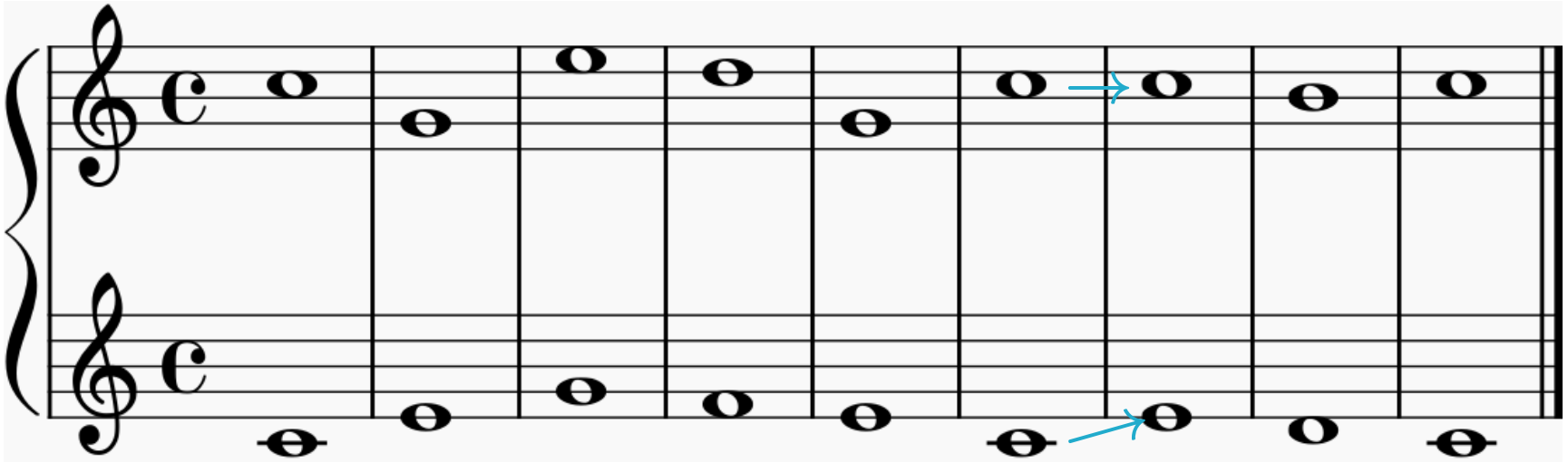
ctr sim par sim ctr obl par ctr



where ctr: contrary, sim: similar, par: parallel, obl: oblique

Key Concept 2: Motion

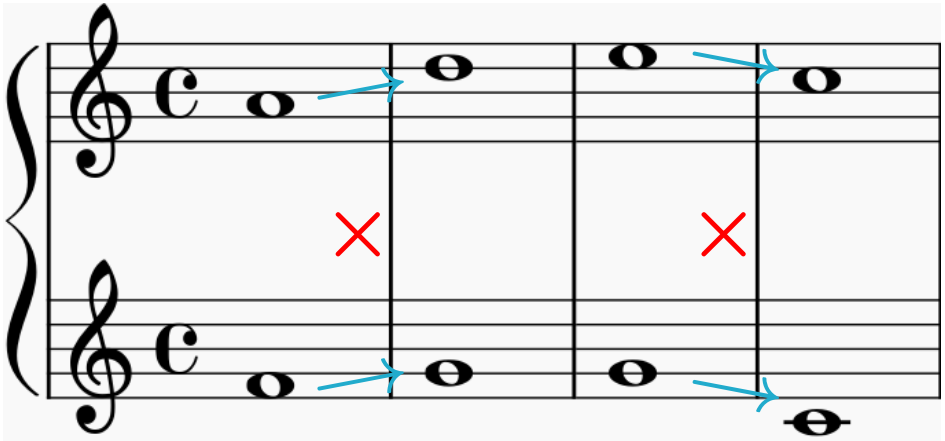
ctr sim par sim ctr obl par ctr



where ctr: contrary, sim: similar, par: parallel, obl: oblique

Rule 2 for First-Species Counterpoint

Perfect intervals must be approached
by ***non-similar motion***



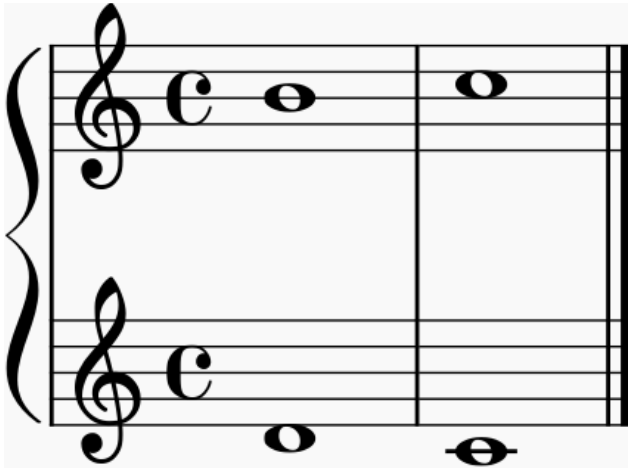
per5 per8

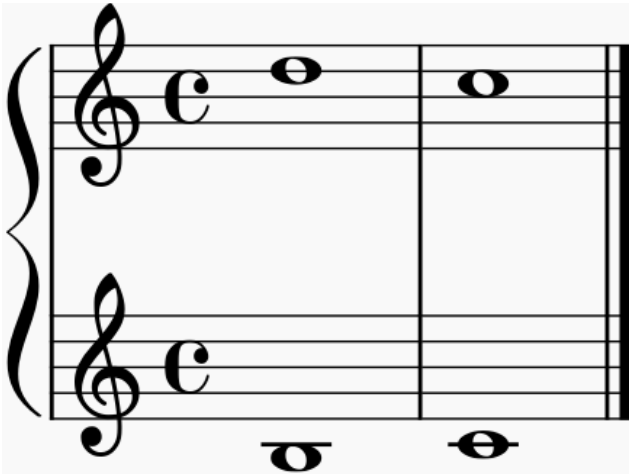
Constraining Motion

```
motionOK : (i1 i2 : Interval) → Set
motionOK i1 i2 with motion i1 i2
         | isPerfectInterval i2
motionOK i1 i2 | contrary | _     = ⊤
motionOK i1 i2 | oblique  | _     = ⊤
motionOK i1 i2 | parallel | false = ⊤
motionOK i1 i2 | parallel | true  = ⊥
motionOK i1 i2 | similar  | false = ⊤
motionOK i1 i2 | similar  | true  = ⊥
```

Rule 3 for First-Species Counterpoint

Final interval must be approached by
contrary step-wise motion





Building Correct Counterpoint

head interval

```
data FirstSpecies : PitchInterval → Set where
      ⋮
```

Building Correct Counterpoint

```
data FirstSpecies : PitchInterval → Set where
  cadence2 : (p : Pitch) →
    FirstSpecies (transpose (+ 2) p , maj6)
  cadence7 : (p : Pitch) →
    FirstSpecies (transpose -[1+ 0 ] p , min10)
      ⋮
```

Building Correct Counterpoint

```
data FirstSpecies : PitchInterval → Set where
  cadence2 : (p : Pitch) →
    FirstSpecies (transpose (+ 2) p , maj6)
  cadence7 : (p : Pitch) →
    FirstSpecies (transpose -[1+ 0 ] p , min10)
  _::_ : (pi : PitchInterval) →
         {pj : PitchInterval} →
         {_  : motionOK pi pj} →  -- implicit
         FirstSpecies pj →
         FirstSpecies pi
```

Demo

Future Work

- Higher-species counterpoint
- Automatic generation of counterpoint
  - ☺ Imperfect intervals, contrary motion
  - ☹ Big intervals, repeats
- Harmonic analysis and synthesis

Code: `https://github.com/halfaya/MusicTools`