An ArrowLoop of Music

October 16, 2011

Introduction

We want to have a generalized musical representation. There are many "standard" formats, but none of these are really generalized. In fact, they are more or less derived from traditional musical notation.

Typical representation such as MusicXML, GUIDO or ABC notation has these flaws:

- Assumptions on pitch names (biased towards diatonic scale)
- Assumptions on tuning (biased towards enharmonic tunings)
- Assumptions on rhythm (biased towards simple divisions)
- Treats notes as the common case, ignoring continous values.
- Biased towards *on* and *off*, ignoring gradual onsets and offsets (so they must be approximated as discrete steps).
- Assumes synchronicity, making it difficult to represent "free" time.

It would be desirable to define a model that does not suffer from these shortcomings, yet still is able to represent everything the above models can represent. In short, a more general model. In thise general model, the traditional models emerge as a special case.

It would also be desirable to connect the new model to physical models of audio.

Preliminaries

We are going to describe our model as a literate Haskell program.

```
module Music.Model.General where

import Prelude hiding (seq)

import Control.Category
import Control.Arrow
```

Using Haskell, the problem of

```
type Duration = Rational
type Time = Rational
type Pitch = Rational

p = 1 / d
```

Requirements

Sequencing and parallelism

Given two events a and b:

- $a >>> b$ means that a and b occur in sequence
- $a *** b$ means that a and b occur in parallel, possibly starting and stopping at different time

Synchronousity

The basic idea is to embed one level of durational values into another. Grace notes and arpeggios are a simple instance of this.

As soon as durations are not synchronous, it seems reasonable to establish synchronization points (colla voca).

- $a \mathbin{\&\&\&} b$ means that a and b occurs in parallel, starting at the same time
- $a \mathbin{!!!} b$ means that a and b occurs in parallel, finishing at the same time

Repetition

- *loop a* means that *a* occurs repeatedly

```
-- | An object that occurs in time. We use the arrow framework to represent
--   composition of events.
class (Arrow t, ArrowPlus t, ArrowLoop t, ArrowApply t) => Temporal t where
    duration :: t a b -> Duration
```

Example events

```
type Note      = (Pitch, Duration)
type Rest      = Duration
```

We do not just want to represent pitches, but arbitrary actions.

Besides impulses, we also want to model continous change. The most general way to do this is as a function of time.

```
type Continous a = Time -> a
```

To model an analogue signal, we use a list of samples at time-domain or frequency-domain.

```
data Sampled a = TimeDomain Duration [a]
               | FreqDomain Duration [a]
```