SCALA++ FORMAT FOR MICROTONAL MUSIC PRODUCTION, ANALYSIS, AND RESEARCH

J.M. KARIMÄKI ET AL.

1. Introduction

The Scala++ format is meant to be a file format for musical tuning scales that supersedes and extends the original Scala file format used by the Scala desktop application[1] for microtonal music, and other musical software, by adding new features, making it possible to express tones in exact symbolic form, both as original numerical values, or in the cent units, and compactly express microtonal tuning systems of more general types than the original Scala format, which is basically just {List of intervals, Period}.

Furthermore, the Scala++ format is designed to be backwards compatible with the original Scala format, in the sense that any acceptable Scala file would also be an acceptable Scala++ file.

In some rare situations a file would be interpreted differently as a Scala++ than as a Scala file. This is necessary for ensuring a consistent interpretation of Scala++ files, without introducing cumbersome additional syntax requirements. The reason for this inconvenience is the restricted nature of the original Scala format.

Date: 2021-03-11.

[1] `http://huygens-fokker.org/scala/`

2. Standard Scala Format

The full specifications of the original Scala file format can be found on the Scala homepages[2], but a short description is given here. The general structure of a typical Scala file is as follows:

```
! optional comment lines that start by '!' and usually
       contain the name of the file as filename.scl
a line which is empty, or contains a short description of
       the scale
! optional comment lines
an integer number n that indicates the number of notes in
       the scale per period
! optional comment lines
n-1 lines, each starting by a note of the scale as
       a rational number or cent value
a line that starts by the period of the scale as
       a rational number or as a cent value
! optional comment lines
```

Scala files are normal text files. The notes of the scale are written as numerical pitch values, either as rational numbers[3], i.e. integers or rational numbers containing one division symbol '/' between two integers, or as cent values, written as decimal numbers[4], and required to have a decimal period '.' somewhere, even at the end[5].

Each comment line is required to start with the exclamation mark '!'. Comments, such as note names, mathematically exact definitions of the pitches, etc., can also be added after the pitch values. Those comments do not require to start with the exclamation mark.

Each scale in Scala is, by definition, assumed to contain the pitch 1/1, and this value is normally omitted from the standard Scala files[6].

[2] `http://huygens-fokker.org/scala/scl_format.html`

[3] Such as 2, 2/1, 3/2, 3, 3/1, 5, 5/3, 7/4, etc.

[4] Such as 100., 133.333, 316., 700., 701.955, 1200., etc.

[5] So, for example 700 cent would have to be written as 700. or 700.0 (but not as 700) in a standard Scala file.

[6] In case a line contains the pitch 1/1, or 1, or 0. (as 0 cent), it is considered a multiplicity of the pitch 1/1. Such pitch lines should not be used under normal circumances in standard Scala files. They may produce unwanted errors when used by third-party musical software.

3. Examples of standard Scala files

Example 1. Olympos enharmonic scale

```
! olympos.scl
Scale of ancient Greek flutist Olympos, 6th century BC as reported by Partch
 5
!
 16/15
 4/3
 64/45
 16/9
 2/1
```

Example 2. Quarter-comma meantone scale

```
! meanquar.scl
1/4-comma meantone scale. Pietro Aaron's temperament (1523)
 12
!
 76.04900
 193.15686
 310.26471
 5/4
 503.42157
 579.47057
 696.57843
 25/16
 889.73529
 1006.84314
 1082.89214
 2/1
```

Example 3. Standard Western scale, 12edo

```
! 12edo.scl
12 equal divisions of the octave. The modern standard tuning scale of the West.
 12
!
 100.
 200.
 300.
 400.
 500.
 600.
 700.
 800.
 900.
 1000.
 1100.
 1200.
```

4. Scala+ format

The Scala+ format is here defined to be the standard Scala format extended by the notation $m\backslash n$ to indicate m degrees of n equal divisions of the octave, or the pitch value $2^{m/n}$. This notational convention has become standard among some segments of the microtonal community, and is, for example, used in Sevish's web app Scale Workshop[7]. It specifically uses the backlash symbol '\', not to be confused with the ordinary slash '/' used for fractions and divisions.

Example 4. Equal Heptatonic scale, 7edo

```
! 7edo.scl
!
7 equal divisions of the octave. Used in traditional Georgian and Thai music.
 7
!
 1\7
 2\7
 3\7
 4\7
 5\7
 6\7
 7\7
```

Example 5. Blackwood[10] scale in 15edo

```
! blackwood10.scl
!
Blackwood[10] in 15edo.
 10
!
 2\15
 3\15
 5\15
 6\15
 8\15
 9\15
 11\15
 12\15
 14\15
 15\15
```

[7] https://sevish.com/scaleworkshop/

5. Scala++ format

The Scala++ format is an extension of the Scala file standard, which is backwards compatible with both the original Scala and the above-described Scala+ standards. It has the additional features described in the following subsections.

5.1. **Generalization of the backslash formulation for any period.** Tthe $m\backslash n$ notation is extended to $m\backslash n; p$ to indicate m degrees of n equal divisions of the interval p, or the pitch value $p^{m/n}$. Furthermore, the following default values are assumed when an explicit value has been omitted: $m = 1$, $n = 12$, and $p = 2$. The default values $m = 1$ and $n = 12$ are also assumed for the shorter notation $m\backslash n$.

5.2. **Cent-valued pitches using the 'c' symbol.** Any pitch, or interval, preceded or followed by the 'c' symbol is interpreted as cent-valued. Thus 1200c is equal to 1200 cent and 3/2c is equal to 1.5c. For pitch values containing the decimal point '.' the use of 'c' at is optional. Alternatively, the 'Cent', 'cent', or '¢' can also be used.

5.3. **Decimal-valued pitches using the '#' symbol.** Any pitch, or interval, prefixed or postfixed by the '#' symbol is interpreted as a real (or complex) number. Thus #1.5 is equal to 3/2, and not to 1.5¢. For mathematical expressions and rational, or integer-valued pitches, the use of '#' is optional. Thus #2, #2/1, 2/1, 2, 2/1#, and 2# are all the same interval.

5.4. **Frequencies as pitches using 'Hz'.** Any pitch value prefixed or postfiex by 'Hz' is to be interpreted as a frequency in Herz units, 1/s. Alternatively, 'Herz', 'herz', ' 1/s', or '/s' can also be used.

5.5. **Mathematical expressions as pitch values.** Pitch values can also be defined using standard mathematical expressions and elementary functions. Mathematical expressions between curly brackets {, } are interpreted as standard mathematical expressions, with + indicating ordinary addition of scalars, * indicating ordinary multiplication of scalars, ^ or ** indicating ordinary exponentiation of scalars, etc. Thus $6/5 * 5/4$ is equal to 3/2, and $6/5 + 5/4$ is equal to 49/20. When the operators + and *, etc. appear without curly brackets they are to be interpreted as addition and multiplication in the logarithmic pitch space, irrespective of whether the pitches themselves are in the original pitch space or in a logarithmic space, such as cents. Thus $2 * 350$c is equal to 700c, 200c + 300c is equal to 500c, and $6/5 + 5/4$ is equal to 3/2. But what if we want to divide in the logarithmic space? One further specification must be added to avoid confusion. Now 700c/2 would naturally be equal to 350c. But what about 3/2/2? Would it be equal to 3/4, or sqrt(3/2)? The confusion is removed if we require a space before '/', if we mean division in the logarithmic space, and no space if we mean a fractional number. Thus 3/2 is equal to 1.5, but 3 /2 or 3 / 2 would be equal to sqrt(3). However 3/2, 3 /2, and 3 / 2 would all be equal to 3/2, since curly brackets imply ordinary mathematical operations. Alternatively, we use the convention nrp, or nRp, for n:th roots of p, with the square root, $n = 2$, as default, and also roots of 2, $p = 2$, as default. Thus $r3 = 2r3$ for 3^(1/2), and, according to all our conventions thus far: $r3 = 2r3 = 3\ /2 = 3\ /\ 2 = 3\hat{}(1/2) = 3 * *(1/2) = 1\backslash 2; 3 = \backslash 2; 3 = \text{sqrt}(3)$.

5.6. **Predefined mathematical constants.** The following mathematical constants are defined: i, phi, φ, o, e, pi, π, tau, τ , -inf, $-\infty$, +inf, $+\infty$. As they are complex scalars, they can have the pre/postfix '#', but it is optional. Upper and lower case letters are accepted. With nrp defined as above, and $n = 2$, $p = 2$, as defaults, r effectively becomes the constant r = sqrt(2).

5.7. **Exponential function.** Now, since elementary functions are assumed, $\exp(x)$ is the usual exponential function, but with the above conventions, $5 * e = e * *5 = e\hat{}5 = e^5 = \exp(5)$. We can further simplify things by simply using e, or E, as a prefix or postfix to a number signifies the exponential function, so $E5 = e5 = 5e = 5E = \exp(5)$. Now, the c pre/postfix can similarly be understood as a power, namely x c $= (2^{1/1200})^x = x\backslash 1200$.

5.8. **Logarithmic functions.** This is $\ln(x)$, or $\log(x)$, but using l or L as a prefix or postfix would be accepted. Thus $l2 = L2 = 2L = 2l = \log(2)$. A base-k logarithm $\log_k(x)$ or $\log(k, x)$ can be written using the shorthand klx, or $k\mathrm{L}x$. Thus the base-2 logarithm would be written $2\mathrm{L}x$. The capital L is preferred to avoid confusing the letter 'l' with the number '1'.

5.9. **Chains of generators.** In the standard Scala file the last pitch line indicates the period of the scale. It is assumed that all scales have a period, and it is usually equal to 2/1. Now, we could add extra periods after the last pitch line. Since a period actually just means a generator whose exponents vary between -inf and +inf, we can indicate periods alternatively as 2/1 -inf +inf, making the previous statement explicit. Thus we can generalize this to format: generator firstExponent lastExponent step. For example 3/2 -1 3 1 would indicate the chain (2/3, 1, 3/2, 9/4, 27/8). When those are folded into the octave, we get (1, 9/8, 4/3, 3/2, 27/16). With the octave as period this is the scale chin_5.scl from the Scala scale archive: Chinese pentatonic from Zhou period.