likely music

Probabilistische Musiknotation

Lukas Epple
post@lukasepple.de

26. September 2017

Zusammenfassung

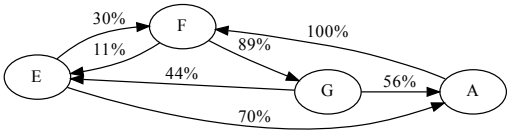
likely music ist eine Software, um probabilistische Musik zu notieren und abzuspielen. Probabilistische Musik bedeutet in diesem Falle, dass die Interpretation der vorliegenden Notation deutlich freier ist als bei herkömmlicher Musik und auch die Reihenfolge der Noten betrifft. Um dies zu erreichen, wird ein eigenes Modell von Musiknotation verwendet. Anstelle von linearer Reihenfolge von Noten bzw. Akkorden tritt ein gerichteter Graph, in dem die Noten (bzw. Akkorde) die Knoten und die möglichen Übergange zwischen diesen die Kanten darstellen, wobei jeder Kante eine gewisse Wahrscheinlichkeit zugeordnet ist. Dieses Modell ist unter anderem sehr gut von einem Computer zu fassen, wodurch es möglich ist, solche Notationen automatisch zu „interpretieren“ oder abzuspielen: Eine konkrete Notenabfolge wird gemäß der Notation ausgewürfelt.

Die Software *likely music* kann sowohl probabilistische Noten erstellen und editieren, als auch mittels MIDI diese abspielen oder als Audiodateien exportieren.

Idee

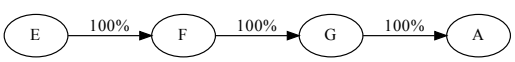
Der eigentlichen Idee ging ein mehr oder minder gescheitertes Projekt für diesen Wettbewerb voraus. Im Frühjahr diesen Jahres entschied ich mich, dieses – eine Demo [1] – abzubrechen, einfach weil ich befürchtete, es nicht bis zur Frist fertigstellen zu können. Die Motivation für dieses Projekt speiste sich aus meiner Faszination für Demos an sich, denn ich hatte mich bereits im Vorfeld öfters mit diesen beschäftigt und beim Ansehen der Einsendung von Demo-Wettbewerben ein Bedürfnis entwickelt auch eine zu entwickeln. Das neue Projekt speiste sich aus einer weiteren Faszination von mir, nämlich einer für Kunst, die durch Zufall entsteht. Ich erinnere mich besonders oft an Kunstinstallationen, die ihr gestaltendes Element durch Zufall, einen undurchschaubaren oder chaotischen Prozess bezieht. Beim Nachdenken über Zwölftonmusik, die – meiner Meinung nach – ein wenig jenen Elements hat, kam mir die Grundidee für *likely music* – wie ich mich erinnere – auf dem Gang zwischen zwei Schulstunden: Nämlich ein Modell, um Musik zu beschreiben, die zufällig im Vortrag ist.

Das Modell, das ich aus Angst, es zu vergessen, mehrmals aufschrieb, sieht Musik als gerichteten Graphen, wobei die Knoten Musiknoten einer bestimmten Länge und die Kanten zwischen ihnen die Wahrscheinlichkeit des Wechsel von der einen Note zu anderen sind. Vorstellen kann man sich es in etwa wie in der folgenden Grafik.



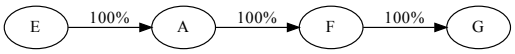
In diesem konkreten Graphen sind die Noten E, F, G und A als Knoten vertreten (der Einfachheit halber sind die Notenlängen weggelassen). Beispielsweise vom E führen zwei Kanten weg, eine zum F mit dreißigprozentiger Wahrscheinlichkeit und eine zum A mit siebzigprozentiger Wahrscheinlichkeit, d. h. nach dem E kommt in sieben von zehn Fällen das A und in den drei übrigen das F. Analog verhält es sich mit den anderen Noten.

Diese Darstellung ist in gewisser Weise auch nur eine ausdrucksstärkere Form einer normalen Notation, denn ein Weg durch den obigen Graphes könnte so aussehen:



Diese Interpretation, die eine Wahrscheinlichkeit von ca. 15% hat, entspricht einer einfachen, linearen Notation, wie sie in einem Gesangsbuch ste-

hen könnte. Wir sehen also, dass solche probabilistiche Noten (wie unser Graph von vorhin) durch ein Verfahren, das ich einfach in einer Erweiterung des Begriffs als Interpretieren bezeichne, auf eine lineare Notation reduziert werden können, die mit einem Instrument oder vom Computer gespielt werden können. Es ist sogar nicht nur eine lineare Notation, sondern – je nach vorgegebenem Graph – eine Vielzahl ihrer möglich. Beispielsweise wäre eine weitere:



Ähnlich enthält der ursprüngliche Graph weitere Möglichkeiten von klassischen Tonabfolgen. Insofern stellt eine probabilistische Notation eine ausdruckstärkere und mächtigere Notation dar, da sie beliebig viele klassische fassen kann.

Zu beachten ist bei den beiden Beispielinterpretationen noch: Sie sind nach vier Noten abgeschnitten, denn, da von jedem Knoten mindestens eine Kante ausgeht, könnte man den Graphen potentiell unendlich lang ablaufen und würde somit eine unendlich lange Interpretation generieren.

Was aus dieser Grundidee zu machen war, schien mir von Anfang an recht klar: Als Software implementieren, um ein graphisches Interface bereitzustellen, das es erlaubt, probabilistische Notation zu erstellen, zu editieren und abzuspielen.

Umsetzung

Gleich zu Beginn war klar, dass Haskell die Programmiersprache der Wahl werden sollte. Sie ist die Sprache, die ich in den letzten Jahren am aktivsten verwendet habe und mir einiges bietet: Statische Typisierung, um Fehler vorzubeugen, ein expressives Typsystem, das es erlaubt, Daten besser zu strukturieren, und funktionale Programmierparadigmen, die sich für mich sehr natürlich anfühlen und das Testen von Programmen erleichtern, um einige Vorzüge zu nennen.

Zunächst konzentrierte ich mich darauf, den Graphen und den Interpretationsalgorithmus als Bibliothek zu implementieren. In der ersten Iteration dieser Bibliothek, noch *probable music* genannt, begann ich auch einen eigenen Softwaresynthesizer zu implementieren, der flexibel auf verschiedenen Plattformen und zu verschiedenen Zwecken verwendet werden kann. Der Synthesizer konnte jegliche Darstellungen von Klängen, Tönen oder Musik dank flexibler Architektur in tatsächliche Töne bzw. Audiowellen umwandeln. Dies ergab interessante Möglichkeiten, sich außerhalb des Zwölftonsystems zu bewegen. Die Tonerzeugung basierte dann auf einer freien Monade [2], die die Instruktionen ›Warten‹ und ›Abspielen‹ kannte. Indem man diese Instruktionen für verschiedene Audiosystem, wie SDL [4], Jack [3] oder auch Audiodateien wie WAV [5] implementierte, konnte man verschiedene Plattformen unterstützen. Allerdings gestaltete es sich schwierig, einen gut klingenden Synthesizer zu schreiben, denn die Messlatte ist im Vergleich zu realen Instrumenten hoch. Hinzu kamen noch einige Performance-Probleme mit meinem maschinennahen Audio-Code.

Also entschied ich mich, die Library vor allem auf den Graphen und die dazugehörigen Algorithmen zu fokusieren und zur Tonerzeugung eine geeignete Abstraktion zu verwenden, um diese zu vereinfachen. Ich habe hierfür MIDI gewählt, eine Technologie, die schon lang in allen Arten von Software und Hardware zur Musikproduktion verwendet wird. MIDI basiert auf einer Abfolge von zeitlich abgestimmten Nachrichten, wie zum Beispiel ›Note C an‹ oder ›Note C aus‹. Aufgrund dieser Nachrichten kann man die Erzeugung und das Abspielen von Musik zwischen mehreren Programmen aufteilen. Außerdem erlaubt es, die bereits existierende Infrastruktur für MIDI-Verarbeitung zu verwenden, die sehr beachtlich ist. Für MIDI verwendet *likely music* die Open-Source-Bibliothek Euterpea[1] [8], die unter anderem eine kleine Abstraktion über MIDI enthält. Sie erlaubt es, in einem internen Format Musik zu konstruieren und anschließend als MIDI zu exportieren bzw. an ein anderes Programm zur Weiterverarbeitung zu schicken.

Bei der Darstellung des Graphen habe ich mich vor allem darauf konzentriert, den Interpretationsalgorithmus, also das (zufällige) Ablaufen des Graphen, möglichst effizient zu gestalten. Da es sich

[1] Ich musste allerdings aufgrund von Inkompatibilitäten mit den aktuellen Haskell-Paketen diese selbst beheben [9]. Diese Änderung wartet [10] aktuell (Stand 23.09.2017) darauf, vom Hauptentwickler in den Code von Euterpea übernommen zu werden.

um einen gerichten Graphen handelt, ist es besonders wichtig zu wissen, wohin man von einem gegebenen Knoten aus gelangen kann bzw. welche Kanten von einem Knoten weggehen. So gelangt man in unserem Beispiel aus dem vorherigen Kapitel vom Knoten mit dem E zu den Knoten mit F und A. Es muss also möglichst effizient sein, die Kanten nachzuschlagen, die von einem Knoten *wegführen*. Mit der Datenstruktur *Map* [11] (im deutschen Sprachgebrauch typischerweise *assoziative Datenfeld* bzw. *assoziatives Array*) kann man genau das sehr leicht realisieren: Man verwendet die Knoten als Schlüssel und eine Liste von Kanten, die vom Schlüssel weggehen, als Elemente. Wenn der Algorithmus nun einen Knoten nachschlägt, erhält er direkt die Kanten, die von diesem Knoten weggehen und somit auch die nächsten möglichen Knoten. Dies ist die einzige Information, die in jedem Schritt benötigt wird. Die Operation des Nachschlagen hat in einem *Map* die Komplexität $O(\log n)$ [12], d. h. die Zeit, die benötigt wird, um ein Element nachzuschlagen, steigt mit dem Wachsen der Datenstruktur logarithmisch (d. h. weniger starkes Wachstum als linear!). Damit bleibt auch das Interpretieren großer Graphen ziemlich schnell. Der Code für die Datenstruktur findet sich im Abschnitt Library, Zeile 30 bis 43.

Der Interpretationsalgorithmus selbst ist rekursiv [15] gestaltet und findet sich in der Funktion `interpretation`, siehe Abschnitt Library, Zeile 52 bis 60. Diese Funktion benötigt einen initialisierten Pseudozufallszahlengenerator [13, 14], den zu interpretierenden Graphen in der eben besprochenen Datenstruktur und einen Startknoten. Nach Ablauf der Berechnung gibt die resultierende Interpretation im MIDI-Format von Euterpea [8] zurück. Zunächst wird der Startknoten im Graphen nachgeschlagen, so werden die Kanten bzw. die nächsten möglichen Knoten erhalten. Nun gibt es zwei Möglichkeiten für den weiteren Verlauf:

1. Es gibt keine Kanten, die von diesem Knoten ausgehen. Also wird die bisher generierte Interpretation einfach zurückgegeben, die Funktion terminiert.

2. Wenn es eine oder mehr Kanten vom Knoten aus gibt, wird eine (reelle) Zufallszahl zwischen 0 und 1 berechnet und mittels der Hilfsfunktion `edgeForRoll` (siehe Abschnitt Library, Zeile 62 - 67) die Kante erhalten, die gemäß des zufälligen Ergebnis als nächstes abgelaufen werden soll. Nun ergibt sich das gleiche Problem wie zu Beginn der Interpretation: Man kennt einen Knoten und will wissen, wie es weitergeht. Also wird nach der Ermittlung des zweiten Knotens die MIDI-Nachrichten aus dem Startknoten extrahiert und dann der Interpretationsalgorithmus nochmal bzw. rekursiv aufgerufen – nur mit dem Folgeknoten als Startknoten. Dessen Ergebnis wird an die aktuellen MIDI-Nachrichten angehängt, was jener Aufruf auch seinerseits wieder macht. So entsteht rekursiv eine (potentiell unendliche) Verkettung von MIDI-Nachrichten, die letztlich die finale Interpretation ergeben.

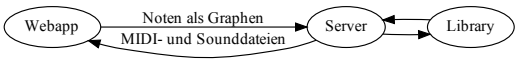
Da die meisten Graphen vermutlich vollständig untereinander verbunden sein werden, wie zum Beispiel der Beispielgraph im ersten Abschnitt, entstehen unendlich lange Interpretationen. Diese zu erstellen benötigt natürgemäß natürlich auch unendlich viel Zeit – der Interpretationsalgorithmus terminiert also nicht. Die einfache Antwort auf dieses Problem ist die Begrenzung der Länge der Interpretation auf eine gewisse Anzahl von Noten, was sich dank eines Sprachfeatures von Haskell – Lazy Evaluation [16] – leicht umsetzen lässt. Denn mit Lazy Evaluation wird nur das berechnet, was im Moment benötigt wird. Somit werden zum Beispiel nur die ersten vier benötigten Noten berechnet und nicht die unendlich vielen die eigentlich noch darauf folgen würden – genau dies wird durch die Funktion `takeNotes` (siehe Abschnitt Library, Zeile 79 - 86) realisiert.

Nun können wir probabilistische Musik in Graphen darstellen, diese automatisch interpretieren und dank Euterpea nach MIDI exportieren. Was fehlt, ist eine angenehme Benutzerschnittstelle.

Zur Technologie für die Benutzerschnittstelle gab es für mich folgende Überlegungen: Zum einen sollte es leicht portabel bzw. auf jedem System laufen sowie außerdem einen begrenzten Entwicklungsaufwand mit sich bringen, damit es bis zum Einsendeschluss auch fertig sein würde. Ich selbst entwickle meine Software auf GNU/Linux, aber zur Abgabe müsste es auf macOS und / oder Windows laufen. Alle größeren Frameworks für Graphische Interfaces für GNU/Linux, wie zum Beispiel Qt [21] oder GTK [22], laufen auch auf den anderen großer

Betriebssystemen. Allerdings bin ich nicht besonders vertraut mit irgendeinem dieser Frameworks. Außerdem war ich mir nicht sicher, wie stressfrei die Verwendung dieser von Haskell aus sein würde (denn klassischerweise verwendet man C oder C++). Also entschied ich mich *likely music* als Webapplikation, die einfach in gängigen Browsern läuft, zu implementieren. Das hat einige Vorteile für mich, unter anderem, dass es leicht zu testen ist, weil die Browser eigentlich überall gleich sind, und, dass ich schon einige Erfahrung in Webentwicklung hatte.

Allerdings hatte ich die Library schon in Haskell implementiert, in Browsern läuft aber nur JavaScript (ohne größeren Aufwand zumindest). Also musste ein Programm her, um die Kommunikation zwischen der Library und der Webapplikation zu realisieren. Ich entschied mich für eine Client-Server-Architektur [17], also einen Server, der die Interpretation und den Export von Sounddateien für den Client, also die Webapplikation, übernimmt. Der Client wiederum müsste sich ausschließlich um ein ansprechendes Interface kümmern. Die ungefähre Gesamtarchitektur sieht also nun so aus:



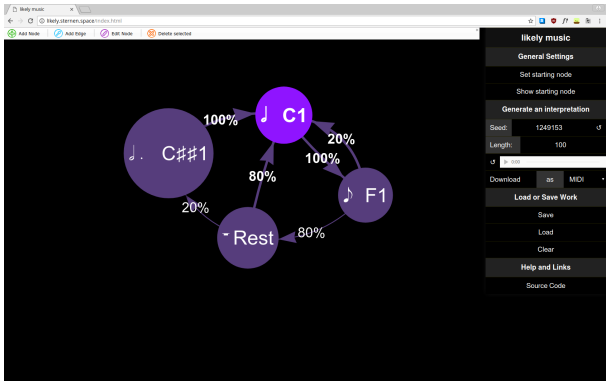
Der Server basiert auf den Libraries servant [18] als Webframework. Wie im Abschnitt Backend zu sehen, besteht das Serverbackend aus zwei Dateien Quelltext: In `Api.hs` wird die Struktur der REST-API [19] definiert, mittels der die Webapplikation mit dem Server kommuniziert. Der Server bietet folgende Funktionalität an:

- `/interpretation/mid` An diesen Endpunkt schickt die Webapplikation einen Graphen plus einiger Parameter in Form von JSON [20] und erhält eine Interpretation auf Basis des Algorithmus als MIDI-Datei zurück.
- `/interpretation/wav` Gleich wie der obige Endpunkt, allerdings wird vorher noch das MIDI mittels des MIDI-Synthesizers fluidsynth [23] in eine WAV-Datei konvertiert, so dass man die Interpretation direkt anhören kann.
- Außerdem liefert der Server die statischen Dateien der Webapplikation, wie das nötige HTML, JavaScript und CSS.

Die erwähnten Parameter sind nur folgende drei:

- Der Anfangsknoten der Interpretation im Graphen, den der Algorithmus benötigt (wie oben besprochen).
- Die Länge der Interpretation als die maximale Anzahl an Noten in der Interpretation.
- Der Startwert für den Pseudozufallszahlengenerator [14], der für die Interpretation verwendet werden soll. Da derselbe Startwert in die selbe Interpretation resultiert, erlaubt dies, sich interessante Interpretationen zu merken und zum Beispiel zu einer Interpretation noch die MIDI-Version zusätzlich herunterzuladen.

Dies ist auch schon alles, was das Serverbackend tut, denn es ist nur als minimaler Aufsatz auf die Library konzipiert. Das meiste für Benutzer*innen relevante passiert in der Webapplikation, die folgendermaßen aussieht:

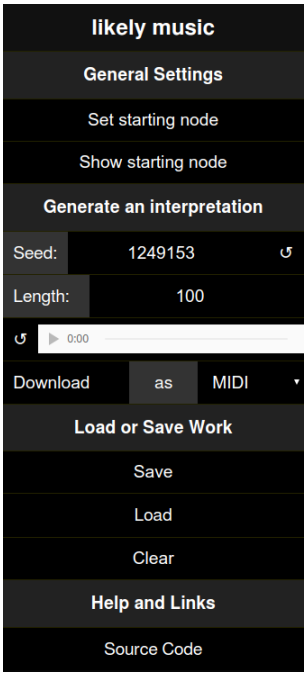


Den Kern der Applikation bildet der Grapheditor links, der auf der Library vis.js[2] [24] basiert. vis.js kümmert sich um einen sehr gut anpassbaren Grapheneditoren, in dem der*die Benutzer*in Knoten und Kanten hinzufügen, löschen und ändern kann. Da die Library Callbacks [26] bereitstellt, ist es leicht den Rest der Applikation mit dem Editor zu integrieren.

[2]Eigentlich nur ein Teil von vis.js namens *network* [25], aber ich werde vis.js immer der Kürze halber synonym für *vis.js network* verwenden.

Wenn ein Knoten oder eine Kante geändert wird, wird diese Änderung in eine Zustandsvariable der Applikation mitübernommen und die Zusatzinformationen der Knoten und Kanten, also Notenlänge und Tonhöhe (Knoten) bzw. Wahrscheinlichkeit (Kante), von dem*der Benutzer*in in einer Einblendung abgefragt und ebenfalls abgespeichert. So gelingt es, den Grapheditor so zu integrieren, dass der Graph zur Kommunikation mit dem Server und sonstiger Verarbeitung zur Verfügung steht. Die doppelte Speicherung der reinen Graphdaten kommt daher, dass vis.js es leider nicht erlaubt, die bereits im Editor vorhandenen Daten abzufragen, daher büßt die Architektur der Applikation leider ein wenig an Eleganz ein.

In der Seitenspalte passiert dann alles, was relevant für die Verarbeitung der links entstehenden Notation ist. Zum einen kann der Notationsgraph abgespeichert oder ein gespeicherter geöffnet werden, zum anderen ist es möglich, Interpretationen generieren zu lassen, diese direkt im Browser abzuspielen oder als MIDI oder WAV herunterzuladen. Die Seitenspalte ist im folgenden abgebildet.



Das Speichern und Öffnen von Notationen basiert auf JSON-Dateien [20] in bestimmten Format, die als `<dateiname>.score.json` abgespeichert werden. Eine solche enthält eine Liste aller Knoten plus Zusatzinformationen und eine Liste aller Kanten plus Zusatzinformationen. Wie eine solche aussehen kann, sieht man im Abschnitt Web (letzte Datei). Genau dieses Format wird übrigens auch zur Kommunikation mit dem Server verwendet, da es den Graphen verlustlos beschreiben kann.

Der Rest der Applikation kümmert sich vor allem um Interpretation und Export dieser. Oben in der Seitenleiste kann man die drei erwähnten Parameter setzen. Der Startknoten wird über markieren dessen im Editor und klicken des entsprechenden Buttons gesetzt und kann durch Hervorhebung im Graphen auch angezeigt werden. Der Startwert kann manuell eingegeben (etwa, wenn man sich einen besonderen notiert hat) oder ein zufälliger durch Verwendung des Buttons neben dem Feld verwendet werden. Die maximale Interpretationslänge ist dann darunter und wird ganz unspektakulär eingegeben.

Darunter befindet sich ein Audioplayer, mit dem erstellte Interpretationen direkt im Browser angehört werden können. Wenn man den Aktualisierungsbutton links betätigt, nimmt die Applikation alle Parameter sowie den aktuellen Graphen und sendet mithilfe der JavaScript Fetch API [27] den Graphen mitsamt der Parameter an den bereits erwähnten Endpunkt `/interpretation/wav`. Nach diesem Vorgang, der merklich Zeit benötigt, da fluidsynth [23] erst das WAV generieren muss, wird die Audiodatei in den Player geladen und kann direkt angehört werden.

Gleich unter dem Player kann man die Interpretation als MIDI oder WAV herunterladen. Dazu wählt man rechts eines der beiden Formate aus und klickt links auf „Download“. Intern funktioniert dies genau gleich wie der Player, bloß dass die jeweils der Endpunkte für das entsprechende Format verwendet wird und die Datei dann direkt heruntergeladen wird statt im Browser weiterverwendet wird.

Lizenzierung

Der gesamte Quelltext von *likely music* ist unter der *GNU Affero General Public License Version 3* lizenziert. Die AGPL ist eine Freie-Software-Lizenz [29], das heißt, sie sichert dem*der Benutzer*in gegenüber dem Entwickler verschiedene Rechte (typischerweise nennt man vier) zu. Diese Rechte haben alle emanzipatorischen Charakter für den Nutzer: Das Recht die Software so auszuführen, wie der Nutzer es mag, natürlich offensichtlichlerweise. Das Recht, den Quellcode zu erhalten und zu untersuchen hilft vor allem dem*der Benutzer*in zu verstehen, was eigentlich auf seinem*ihrem Com-

puter vor sich geht, und kann auch der Weiterbildung dienen. Die Freiheit, die Software frei und ohne Lizenzgebühren an andere weiterzugeben, ist mir besonders wichtig. Aufgrund diesen Umstandes kann freie Software unentgeltlich an jede*n weitergegeben werden, was Zugang zu Software unabhängig des eigenen Geldbeutels erlaubt – vorausgesetzt man besitzt einen Computer. Diese Freiheit geht sogar noch weiter, dahingehend, dass auch die Modifikation ausdrücklich erlaubt (und erwünscht) ist. Somit kann nicht nur jede*r freie Software erhalten, sondern auch mitgestalten und verbessern. Auch andere freie Software kann profitieren, indem sie von anderen Projekten Code übernimmt. Dank der restriktiven Weitergabeklauseln kann aber nie freie Software verwendet oder verändert werden, ohne dass sie wieder freie Software wird. Freie Software erhält sozusagen ihre eigene Freiheit.

Mir ist dies an dieser Stelle ein besonderes Anliegen, weil ich – mit Sicherheit im Gegensatz zu den allermeisten anderen Wettbewerbteilnehmer*innen – mein Projekt komplett mit freier Software erstellen konnte. Ich war nicht auf eine von drei teuren Softwarelösungen großer Konzerne angewiesen, um meinen Beitrag anzufertigen, wie das zum Beispiel im Bereich Videoschnitt der Fall ist (auch weil es kaum ausgereifte freie Software in dem Bereich gibt).

Insofern sehe ich auch den emanzipatorischen Charakter von freier Software, denn Zugang zu Computern ist größtenteils auch dank von Bibliotheken selbstverständlich geworden, Zugang zu Software, die mehrere hundert Euro kostet, aber mit Sicherheit nicht. Der Preis von Software, die ein Konzern vielleicht auch irgendwann verwahrlosen lässt, ist sicher für viele eine Hürde, vielleicht sogar eine Hürde an diesem Wettbewerb teilzunehmen.

Zukünftige Weiterentwicklung

likely music als fertig zu bezeichnen wäre nicht ganz falsch und nicht ganz richtig. Es handelt sich zwar um eine voll funktionsfähige Software, aber dennoch ist noch einige Weiterentwicklung, für die ich keine Zeit mehr hatte, denkbar. Folgende Gedanken hatte ich bisher:

- **Unterstützung für Akkorde im Interface.** Zwar unterstützen Euterpea und die Library beide Akkorde, aber im Frontend gibt es keine Möglichkeit, solche hinzuzufügen, da ich die Euterpea-MIDI-Datenstruktur nicht vollständig in JavaScript nachgebaut habe. Dies zu beheben wäre für die Zukunft auf jeden Fall wünschenswert.
- **Mehrstimmige bzw. parallele probabilistische Musik.** Denkbar wäre es, eine Möglichkeit hinzuzufügen mehrere Startknoten auszuwählen, von denen dann zwei gleichzeitige Pfade durch den Graph ausgingen. Dies scheint mir die interessantes Möglichkeit zu sein, Mehrstimmig für *likely music* umzusetzen.
- **Import bereits durchkomponierter Musik.** Indem man die Möglichkeit schafft, bereits in konventionellen Notationsprogrammen erstellte Musik zu importieren, könnte man ein für den*die Benutzer*in angenehme Möglichkeit bieten, konventionell notierter Musik ein probabilistisches Element zu geben bzw. sie probabilistisch umzusetzen.

Diese Änderungen stehen nicht im Konflikt mit dem bisherigen Grundkonzept und -aufbau von *likely music*, dürften daher ohne größere Probleme umgesetzt werden können.

Links

- Der gesamte Quelltext `https://github.com/sternenseemann/likely-music`
- Eine laufende Instanz[3] von *likely music* `https://likely.sternen.space`

Danksagung

- Meinem Lehrer Bastian Walcher für seine Betreuung meines Projekt und derer meiner Mitschüler*innen.

[3] *likely music* ist bisher noch nicht auf Performance optimiert worden. Ich glaube nicht, dass genannte Server einen größeren Ansturm vor allem wegen des Exports zu WAV (fluidsynth [23] ist ziemlich langsam) aushalten würde. Daher möchte ich darum bitten, diesen Link nicht zu veröffentlichen, sondern, falls etwas in der Art gewünscht sein sollte, mit mir Rücksprache zu halten.

- Lukas G. für sein Korrekturlesen.
- Christine S. für ihr Korrekturlesen.
- kohlrabi dafür, dass er sich mit mir über Musikprogrammierung und -theorie unterhielt und Ideen zu meinem Projekt beisteuerte.
- all dafür, dass er mich in Richtung Musikprogrammierung stieß.

Literatur

[1] https://de.wikipedia.org/wiki/Demoszene

[2] http://www.haskellforall.com/2012/07/purify-code-using-free-monads.html

[3] http://www.jackaudio.org/

[4] https://www.libsdl.org/index.php

[5] https://de.wikipedia.org/wiki/RIFF_WAVE

[6] https://www.midi.org/

[7] https://de.wikipedia.org/wiki/Musical_Instrument_Digital_Interface

[8] https://hackage.haskell.org/package/Euterpea

[9] https://github.com/sternenseemann/Euterpea2

[10] https://github.com/Euterpea/Euterpea2/issues/16

[11] https://hackage.haskell.org/package/containers-0.5.10.2/docs/Data-Map-Lazy.html#t:Map

[12] https://hackage.haskell.org/package/containers-0.5.10.2/docs/Data-Map-Lazy.html#v:lookup

[13] https://hackage.haskell.org/package/random-1.1/docs/System-Random.html#t:RandomGen

[14] https://en.wikipedia.org/wiki/Pseudorandom_number_generator

[15] https://de.wikipedia.org/wiki/Rekursion

[16] https://de.wikipedia.org/wiki/Lazy_Evaluation

[17] https://en.wikipedia.org/wiki/Client%E2%80%93server_model

[18] https://hackage.haskell.org/package/servant

[19] https://de.wikipedia.org/wiki/Representational_State_Transfer

[20] http://json.org/

[21] https://www.qt.io/

[22] https://www.gtk.org/

[23] http://www.fluidsynth.org/

[24] http://visjs.org/

[25] visjs.org/docs/network/

[26] https://en.wikipedia.org/wiki/Callback_(computer_programming)

[27] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

[28] https://www.gnu.org/licenses/agpl-3.0.html

[29] https://www.gnu.org/philosophy/free-sw.de.html

Anhang

Quelltext

Library

lib/Sound/Likely.hs

```
1  --  Copyright 2017 Lukas Epple
2  --
3  --  This file is part of likely music.
4  --
5  --  likely music is free software: you can redistribute it and/or modify
6  --  it under the terms of the GNU Affero General Public License as published by
7  --  the Free Software Foundation, either version 3 of the License, or
8  --  (at your option) any later version.
9  --
10 --  likely music is distributed in the hope that it will be useful,
11 --  but WITHOUT ANY WARRANTY; without even the implied warranty of
12 --  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13 --  GNU Affero General Public License for more details.
14 --
15 --  You should have received a copy of the GNU Affero General Public License
16 --  along with likely music. If not, see <http://www.gnu.org/licenses/>.
17
18 {-# LANGUAGE OverloadedStrings #-}
19 {-# LANGUAGE FlexibleInstances #-}
20 module Sound.Likely
21   ( Probability
22   , ID
23   , Node (..)
24   , Edge (..)
25   , Graph (..)
26   , insertNode
27   , insertEdge
28   , interpretation
29   , takeNotes
30   , emptyMusic
31   , exampleGraph
32   ) where
33
34 import Control.Monad
35 import Data.Aeson
36 import Data.Aeson.Types (Parser ())
37 import Data.Maybe
38 import Data.Text (Text ())
39 import Euterpea
40 import System.Random
41 import qualified Data.Map as M
42 import qualified Data.Set as S
43
44 type Probability = Double
45 type ID = Text
46
47 data Node
48   = Node
49   { nId :: ID
50   , nMusic :: Music Pitch
51   } deriving (Show, Eq, Ord)
52
53 data Edge
54   = Edge
55   { eTo   :: Node
```

```
56    , eProb :: Probability
57    } deriving (Show, Eq, Ord)
58
59  newtype Graph = Graph { unGraph :: M.Map Node (S.Set Edge) }
60    deriving (Show, Eq, Ord)
61
62  insertNode :: Node -> Graph -> Graph
63  insertNode t = Graph . M.insertWith S.union t S.empty . unGraph
64
65  insertEdge :: Node -> Edge -> Graph -> Graph
66  insertEdge n e =
67    insertNode n . Graph . M.insertWith S.union n (S.singleton e) . unGraph
68
69  interpretation :: RandomGen g => g -> Graph -> Node -> Music Pitch
70  interpretation gen graph n = (nMusic n) :+:
71    recurse (fromMaybe S.empty (M.lookup n (unGraph graph)))
72    where (prob, gen') = randomR (0.0, 1.0) gen
73          recurse edges =
74            if S.null edges
75              then emptyMusic
76              else interpretation gen' graph
77                   . eTo . edgeForRoll prob $ edges
78
79  edgeForRoll :: Probability -> S.Set Edge -> Edge
80  edgeForRoll prob set =
81    let curr = S.elemAt 0 set
82      in if prob <= eProb curr
83           then curr
84           else edgeForRoll (prob - eProb curr) (S.delete curr set)
85
86  emptyMusic :: Music a
87  emptyMusic = Prim (Rest 0)
88
89  exampleGraph :: Graph
90  exampleGraph = Graph $ M.fromList
91    [ (Node "bla" (c 4 qn), S.fromList [ Edge (Node "blub" (d 4 qn)) 1 ] )
92    , (Node "blub" (d 4 qn), S.fromList [ ])
93    ]
94
95  -- | Take the first @n@ notes of a 'Music'
96  takeNotes :: Integer -> Music a -> Music a
97  takeNotes _ m@(Prim _) = m
98  takeNotes n (Modify c m) = Modify c $ takeNotes n m
99  takeNotes _ m@(_ :=: _) = m
100 takeNotes n (m1 :+: m2)
101   | n <  1    = emptyMusic
102   | n == 1    = m1
103   | otherwise = m1 :+: takeNotes (n - 1) m2
104
105 instance FromJSON Node where
106   parseJSON = withObject "Node" $ \v ->
107     Node <$> v .: "id" <*> (Prim <$> v .: "music")
108
109 lookupNode :: Text -> [Object] -> Parser Node
110 lookupNode id nodes = do
111   matches <- filterM (fmap (== id) . (.: "id")) nodes
112   case matches of
113     [node] -> parseJSON (Object node)
114     _ -> fail "Couldn't␣match␣node␣by␣id"
115
116 buildMap :: [Object] -> [Object] -> Graph -> Parser Graph
117 buildMap _ [] m = pure m
```

```
118 buildMap nodes (e:es) m = do
119   toId <- e .: "to"
120   fromId <- e .: "from"
121   edge <- Edge <$> lookupNode toId nodes <*> e .: "prob"
122   from <- lookupNode fromId nodes
123   buildMap nodes es $ insertEdge from edge m
124
125 instance FromJSON Graph where
126   parseJSON = withObject "Graph" $ \v -> do
127     edges <- v .: "edges"
128     nodes <- v .: "nodes"
129     buildMap nodes edges $ Graph mempty
130
131 instance FromJSON (Primitive Pitch) where
132   parseJSON = withObject "Primitive" $ \v -> do
133     -- TODO Ratio _Integer_ is easy DOSable
134     -- RAM consumption
135     duration <- v .: "dur"
136     octave <- v .: "octave"
137     pitchClass <- v .: "pitch"
138     case pitchClass of
139       "Rest" -> pure $ Rest duration
140       p -> pure $ Note duration (read pitchClass, octave)
```

Backend

backend/Api.hs

```
1  --  Copyright 2017 Lukas Epple
2  --
3  --  This file is part of likely music.
4  --
5  --  likely music is free software: you can redistribute it and/or modify
6  --  it under the terms of the GNU Affero General Public License as published by
7  --  the Free Software Foundation, either version 3 of the License, or
8  --  (at your option) any later version.
9  --
10 --  likely music is distributed in the hope that it will be useful,
11 --  but WITHOUT ANY WARRANTY; without even the implied warranty of
12 --  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13 --  GNU Affero General Public License for more details.
14 --
15 --  You should have received a copy of the GNU Affero General Public License
16 --  along with likely music. If not, see <http://www.gnu.org/licenses/>.
17
18 {-# LANGUAGE OverloadedStrings #-}
19 {-# LANGUAGE FlexibleInstances #-}
20 {-# LANGUAGE DataKinds         #-}
21 {-# LANGUAGE TypeOperators     #-}
22 module Api where
23
24 import Data.Aeson
25 import Data.ByteString.Lazy (ByteString ())
26 import Data.Monoid ((<>))
27 import Data.Ratio
28 import Data.Text (Text ())
29 import GHC.Generics
30 import Servant.API
31 import Sound.Likely
32
33 type LikelyApi = "interpretation" :> Capture "format" OutputFormat
34                                   :> ReqBody '[JSON] GraphWithParams
35                                   :> Post '[OctetStream] ByteString
36                 :<|> "seed" :> Get '[JSON] Int
37                 :<|> Raw
38
39 data OutputFormat = Midi | Wav
40   deriving (Show, Eq, Ord)
41
42 instance FromHttpApiData OutputFormat where
43   parseUrlPiece "mid" = Right Midi
44   parseUrlPiece "wav" = Right Wav
45   parseUrlPiece x     = Left $ "Couldn't␣match␣" <> x <> "␣with␣{mid,␣wav}"
46
47 data GraphWithParams
48   = GraphWithParams
49   { gpParams :: Params
50   , gpGraph  :: Graph
51   } deriving (Show, Eq, Ord)
52
53 instance FromJSON GraphWithParams where
54   parseJSON = withObject "GraphWithParams" $ \v ->
55     GraphWithParams <$> v .: "params"
56                     <*> v .: "graph"
57
58 data Params
59   = Params
```

```
60    { pMaxHops      :: Int
61    , pStartingNode :: Node
62    , pSeed         :: Int
63    } deriving (Show, Eq, Ord)
64
65  instance FromJSON Params where
66    parseJSON = withObject "Params" $ \v ->
67      Params <$> v .: "maxhops"
68             <*> v .: "starting_node"
69             <*> v .: "seed"
```

backend/Main.hs

```
1   --  Copyright 2017 Lukas Epple
2   --
3   --  This file is part of likely music.
4   --
5   --  likely music is free software: you can redistribute it and/or modify
6   --  it under the terms of the GNU Affero General Public License as published by
7   --  the Free Software Foundation, either version 3 of the License, or
8   --  (at your option) any later version.
9   --
10  --  likely music is distributed in the hope that it will be useful,
11  --  but WITHOUT ANY WARRANTY; without even the implied warranty of
12  --  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13  --  GNU Affero General Public License for more details.
14  --
15  --  You should have received a copy of the GNU Affero General Public License
16  --  along with likely music. If not, see <http://www.gnu.org/licenses/>.
17
18  {-# LANGUAGE OverloadedStrings #-}
19  module Main where
20
21  import Api
22
23  import Codec.Midi (buildMidi)
24  import Codec.ByteString.Builder
25  import Control.Monad.IO.Class
26  import Data.ByteString.Lazy (ByteString ())
27  import qualified Data.ByteString.Lazy as B
28  import Euterpea hiding (app)
29  import GHC.IO.Handle
30  import Network.Wai
31  import Network.Wai.Handler.Warp
32  import Servant
33  import Sound.Likely
34  import System.Directory
35  import System.Exit
36  import System.Environment
37  import System.FilePath.Posix
38  import System.IO
39  import System.Process
40  import System.Random
41
42  api :: Proxy LikelyApi
43  api = Proxy
44
45  midiString :: ToMusic1 a => Music a -> ByteString
46  midiString = toLazyByteString . buildMidi . toMidi . perform
47
48  server :: Server LikelyApi
```

```haskell
server = genInterpretation :<|> randomSeed :<|> serveDirectoryWebApp "web/dist"

randomSeed :: Handler Int
randomSeed = liftIO newStdGen >>= return . fst . random

genInterpretation :: OutputFormat -> GraphWithParams -> Handler ByteString
genInterpretation Midi g = do
  let params       = gpParams g
      maxHops      = fromIntegral . pMaxHops $ params
      randomGen    = mkStdGen $ pSeed params
      song         = interpretation randomGen (gpGraph g) (pStartingNode params)
  return . midiString $ takeNotes maxHops song
genInterpretation Wav g = genInterpretation Midi g >>= synthWav

synthWav :: ByteString -> Handler ByteString
synthWav midi = do
  inName <- tempFile "mid"
  liftIO $ B.writeFile inName midi
  outName <- tempFile "wav"
  (_, _, _, ph) <- liftIO $
    createProcess_ "fluidsynth"
      (proc "fluidsynth"
        [ "-a", "file"
        , "-F", outName
        , "-i"
--        , "/usr/share/soundfonts/FluidR3_GM.sf2"
        , "/nix/store/591834mz365ccwyj3ah2d66ncsqvp8w9-Fluid-3/share/soundfonts/FluidR3_GM2-2.
            sf2"
        , inName ])
        { std_in = CreatePipe }
  code <- liftIO $ waitForProcess ph
  case code of
    ExitFailure _ -> throwError err500 { errBody = "fluidsynth␣failed" }
    ExitSuccess -> do
      out <- liftIO $ B.readFile outName
      liftIO $ removePathForcibly outName
      return out

tempFile :: String -> Handler FilePath
tempFile ext = try 0
  where maxtries = 100
        try :: Integer -> Handler FilePath
        try n
          | n < maxtries = do
            progName <- liftIO $ getProgName
            let path = "/tmp" </> addExtension (makeValid progName ++ "-" ++ show n) ext
            exists <- liftIO $ doesFileExist path
            if exists
              then try (n + 1)
              else pure path
          | otherwise = throwError err500
app :: Application
app = serve api server

main :: IO ()
main = newStdGen >> run 8081 app
```

Web

web/source/index.html

```
<!--

  Copyright 2017 Lukas Epple

  This file is part of likely music.

  likely music is free software: you can redistribute it and/or modify
  it under the terms of the GNU Affero General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  likely music is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Affero General Public License for more details.

  You should have received a copy of the GNU Affero General Public License
  along with likely music. If not, see <http://www.gnu.org/licenses/>.

-->
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <title>likely music</title>
        <link rel="stylesheet" type="text/css" href="custom.css">
        <link rel="stylesheet" type="text/css" href="vis.min.css">
        <script src="main.js"></script>
    </head>
    <body>
        <div id="network"></div>
        <div id="sidebar">
            <h1>likely music</h1>
            <h2>General Settings</h2>
            <button id="set-starting-node">Set starting node</button>
            <button id="show-starting-node">Show starting node</button>
            <h2>Generate an interpretation</h2>
            <div class="multi-inputs">
                <label for="seed">Seed:</label>
                <input type="number" id="seed">
                <button id="random-seed">&#8634;</button>
            </div>
            <div class="multi-inputs">
                <label for="hop-count">Length:</label>
                <input type="number" min="0" id="hop-count" placeholder="Max. note count">
            </div>
            <div id="player-container">
                <button id="reload-player">&#8634;</button>
                <audio id="player" controls></audio>
            </div>
            <div class="multi-inputs">
                <button id="download-audio">Download</button>
                <label for="format">
                    as
                </label>
                <select id="format">
                    <option value="mid">MIDI</option>
```

```
                    <option value="wav">WAV</option>
                </select>
            </div>
            <h2>Load or Save Work</h2>
            <button id="gen-score" class="save">Save</button>
            <label for="upload-score" class="custom-file">
                <input type="file" id="upload-score" >
                <span>Load</span>
            </label>
            <button id="clear-score" class="cancel">Clear</button>
            <h2>Help and Links</h2>
            <a href="https://github.com/sternenseemann/likely-music">Source Code</a>
        </div>
        <div id="edge-overlay" class="hidden␣dialog">
            <h2><span id="edge-operation"></span> edge</h2>
            <div class="multi-inputs">
                <label for="prob">Probability:</label>
                <input id="prob" type="number" min="0.0" max="100">
                <span>%</span>
            </div>
            <div class="multi-inputs">
                <button class="save" id="edge-save">Save</button>
                <button class="cancel" id="edge-cancel">Cancel</button>
            </div>
        </div>
        <div id="node-overlay" class="hidden␣dialog">
            <h2><span id="node-operation"></span> node</h2>
            <div class="multi-inputs">
                <label for="pitch">Pitch:</label>
                <select id="pitch"></select>
            </div>
            <div class="multi-inputs">
                <label for="octave">Octave:</label>
                <input id="octave" type="number" step="1">
            </div>
            <div class="multi-inputs">
                <label>Duration:</label>
                <input min="0" id="numerator" type="number" step="1">
                <span>/</span>
                <input min="0" id="denominator" type="number" step="1">
            </div>
            <div class="multi-inputs">
                <button class="save" id="node-save">Save</button>
                <button class="cancel" id="node-cancel">Cancel</button>
            </div>
        </div>
    </body>
</html>
```

web/source/custom.css

```
/*  Copyright 2017 Lukas Epple

  This file is part of likely music.

  likely music is free software: you can redistribute it and/or modify
  it under the terms of the GNU Affero General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  likely music is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Affero General Public License for more details.

  You should have received a copy of the GNU Affero General Public License
  along with likely music. If not, see <http://www.gnu.org/licenses/>.
*/

body {
    font-size:1em;
    font-family: sans-serif;
    margin: 0px;
    background-color: black;
}

#network {
    width: 79%;
    float:left;
    height: 100vh;
}

#sidebar {
    width: 20%;
    float:right;
    color: white;
    background-color: black;
    box-shadow: 0px 0px 20px #111;
    font-size: 1.2rem;
}

#sidebar > * {
    width: 100%;
    border-top: 1px solid #232200;
    color: white;
    padding-left: 0px;
    padding-right: 0px;
    margin: 0;
}

#sidebar button:hover, #sidebar input:hover,
#sidebar .custom-file:hover, #sidebar select:hover, #sidebar a:hover {
    background-color: #563d7c;
}

#sidebar button, #sidebar input, #sidebar .custom-file, #sidebar select, #sidebar a {
  background-color: #000;
}

#sidebar h1 {
    font-size: 1.5rem;
    padding-top: 0.75rem;
```

```
    padding-bottom: 0.75rem;
    text-align: center;
    background-color: #111;
}

#sidebar h2 {
    font-size: 1.2rem;
    padding-top: 0.9rem;
    padding-bottom: 0.9rem;
    text-align: center;
    background-color: #222;
}

#sidebar select {
  color: white;
  border: none;
  padding: 0.75rem;
  font-size: 1.2rem;
  width: auto;
}

#sidebar a {
  padding-bottom: 0.75rem;
  padding-top: 0.75rem;
  display: inline-block;
  text-decoration: none;
  color: white;
  text-align: center;
}

button {
    border: none;
    color: white;
    background-color:black;
    font-size: 1.2rem;
    margin:0;
    padding:0.75rem;
}

input[type="number"] {
    background-color: #333;
    color: white;
    border: none;
    text-align: center;
    font-size: 1.2rem;
    padding:0.75rem;
}

.custom-file {
    top:0;
    right:0;
    position: relative;
    display: inline-block;
    height: 3rem;
}

.custom-file input[type="file"] {
    position: relative;
    top:0;
    left:0;
    right:0;
    z-index:0;
```

```
    opacity: 0;
    width: 100%;
    height: 100% !important;
    margin:0;
    padding:0;
}

.custom-file span {
    text-align: center;
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    z-index: 1;
    width: 100%;
    height: 3rem;
    pointer-events: none;
    background-color: transparent !important;
    font-size: 1.2rem;
    line-height: 1.5rem;
    padding-top: 0.75rem;
    padding-bottom: 0.75rem;
}

.dialog {
    position: absolute;
    top: 10%;
    left: 25%;
    width: 30%;
    min-width:500px;
    padding: 10px;
    background-color: black;
    color: white;
    box-shadow: 0px 0px 10px #111;
}

.dialog > div {
    height: 3rem;
}

.hidden {
    visibility:hidden;
}

.dialog > div {
  width: 100%;
}

.dialog button {
    padding: 0.75rem;
    font-size: 1.5rem;
}

button.cancel {
    background-color: #a23a30;
}

button.save {
    background-color: #0ea92f;
}

.dialog .multi-inputs {
```

```
  font-size: 1.5rem;
}

.multi-inputs {
  display: inline-flex;
  flex-direction: row;
  flex-wrap: nowrap;
  justify-content: flex-start;
  align-items: baseline;
  width: 100%;
}

.multi-inputs > * {
  flex-grow: 1;
  flex-basis: auto;
  transition: width 0.7s ease-out;
  max-height: 100%;
  text-align: center;
}

.multi-inputs :nth-child(1) {
  text-align: left;
}

.multi-inputs label {
  display: inline-block;
  background-color: #333;
  padding: 0.75rem;
}

.multi-inputs input {
  display: inline-block;
  color: white;
  background-color: #111;
  padding: 0.75rem;
  border: none;
  min-width: 0px;
}

.multi-inputs span {
  display: inline-block;
  padding: 0.75rem;
  background-color: #222;
}

.multi-inputs button {
    padding: 0.75rem;
}

#player-container {
  display: inline-flex;
  align-items: center;
}

#player-container > * {
  flex: auto;
}
```

web/source/main.js

```
//  Copyright 2017 Lukas Epple
//
//  This file is part of likely music.
//
//  likely music is free software: you can redistribute it and/or modify
//  it under the terms of the GNU Affero General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//
//  likely music is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU Affero General Public License for more details.
//
//  You should have received a copy of the GNU Affero General Public License
//  along with likely music. If not, see <http://www.gnu.org/licenses/>.

import vis from 'vis';
import { Map } from 'immutable';
// types / internals

const valid_pitches = [
    'Rest',
    'Cff', 'Cf', 'C',
    'Dff', 'Cs', 'Df',
    'Css', 'D', 'Eff',
    'Ds', 'Ef', 'Fff',
    'Dss', 'E', 'Ff',
    'Es', 'F', 'Gff',
    'Ess', 'Fs', 'Gf',
    'Fss', 'G', 'Aff',
    'Gs', 'Af', 'Gss',
    'A', 'Bff', 'As',
    'Bf', 'Ass', 'B',
    'Bs', 'Bss'
];

const display_pitches = [
    'Rest',
    ' C', ' C', 'C',
    ' D', ' C', ' D',
    ' C', 'D', ' E',
    ' D', ' E', ' F',
    ' D', 'E', ' F',
    ' E', 'F', 'Gff',
    ' E', ' F', ' G',
    ' F', 'G', ' A',
    ' G', ' A', ' G',
    'A', ' B', ' A',
    ' B', ' A', 'B',
    ' B', ' B'
];

function displayPitch(pitch) {
    var i = valid_pitches.indexOf(pitch);
    if(i === -1) {
        throw 'Invalid pitch';
    } else {
        return display_pitches[i];
    }
}
```

```
62
63  function standard_rests(dur) {
64      if(dur.numerator === 1) {
65          switch(dur.denominator) {
66              case 1:
67                  return  ' ';
68                  break;
69              case 2:
70                  return  ' ';
71                  break;
72              case 4:
73                  return  ' ';
74                  break;
75              case 8:
76                  return  ' ';
77                  break;
78              case 16:
79                  return  ' ';
80                  break;
81              case 32:
82                  return  ' ';
83                  break;
84              case 64:
85                  return  ' '
86                  break;
87              case 128:
88                  return  ' '
89                  break;
90              default:
91                  return null;
92                  break;
93          }
94      } else {
95          return null;
96      }
97  }
98
99  function standard_notes(dur) {
100     if(dur.numerator === 1) {
101         switch(dur.denominator) {
102             case 1:
103                 return  ' ';
104                 break;
105             case 2:
106                 return  ' ';
107                 break;
108             case 4:
109                 return  ' ';
110                 break;
111             case 8:
112                 return  ' ';
113                 break;
114             case 16:
115                 return  ' ';
116                 break;
117             case 32:
118                 return  ' ';
119                 break;
120             case 64:
121                 return  ' '
122                 break;
123             case 128:
```

```
                return  ' '
                break;
            default:
                return null;
                break;
        }
    } else if(dur.numerator === 2 && dur.denominator === 1) {
        return  ' '
    } else {
        return null;
    }
}

function compute_dot_times(dur, den) {
    let term = den * ( (2 / den) - (dur.numerator / dur.denominator));
    return [ den, -Math.log2(term) ];
}

function musical_symbol(lookup, dur) {
    // unicode characters sometimes hide from you!
    const dot =  ' ';
    let isNat = n => {
        if (typeof n !== 'number')
            return false;
        return (n >= 0.0) && (Math.floor(n) === n) && n !== Infinity;
    };
    var standard_symbol = lookup(dur);
    var bla = [0, 1, 2, 3, 4, 5, 6, 7 ].map(compute_dot_times.bind(this, dur));
    console.log(bla);
    var dots = bla.filter(([den, dots]) => isNat(dots));
    console.log(dots);

    if(standard_symbol !== null) {
        return standard_symbol;
    } else if (dots.length !== 0) {
        var symbol = lookup(new Rational(1, dots[0][0])) + ' ';
        for(var i = dots[0][1]; i > 0; i--) {
            symbol = symbol + dot;
        }
        return symbol;
    } else {
        return dur.toString();
    }
}

class Music {
    constructor(dur, pitch_class, octave) {
        this.dur = dur;
        if(valid_pitches.indexOf(pitch_class) !== -1) {
            this.pitch = pitch_class;
        } else {
            throw `Invalid pitch class '${pitch_class}'`;
        }
        this.octave = octave;
    }

    toString() {
        if(this.pitch === 'Rest') {
            return `${displayPitch(this.pitch)} for ${this.dur.toString()}`;
        } else {
            return `${displayPitch(this.pitch)}${this.octave} for ${this.dur.toString()}`;
        }
```

```
186     }
187
188     nodeText() {
189         if(this.pitch === 'Rest') {
190             return `${musical_symbol(standard_rests, this.dur)} Rest`;
191         } else {
192             return `${musical_symbol(standard_notes, this.dur)}   ${displayPitch(this.pitch)}$
                {this.octave}`
193         }
194     }
195
196
197     static fromObject(obj) {
198         return new Music(Rational.fromObject(obj.dur), obj.pitch, Number(obj.octave));
199     }
200 }
201
202 class Rational {
203     constructor(a, b) {
204         this.numerator = a;
205         this.denominator = b;
206         this.reduce();
207     }
208
209     reduce() {
210         let gcd = (a, b) => !b ? a : gcd(b, a % b);
211         let div = function(a, b) {
212             if(b === 0) {
213                 throw 'Divide by zero';
214             } else {
215                 return Math.floor(a / b);
216             }
217         };
218
219         var d = gcd(this.numerator, this.denominator);
220         this.numerator = div(this.numerator, d);
221         this.denominator = div(this.denominator, d);
222     }
223
224     toString() {
225         return `${this.numerator}/${this.denominator}`;
226     }
227
228     static fromObject(obj) {
229         return new Rational(obj.numerator, obj.denominator);
230     }
231 }
232
233 function collectGraphData(nodeDate, edgeData) {
234     return {
235         nodes: [... nodeData.values()].map(x => ({
236             id: x.nodeData.id,
237             music: x.music
238         })),
239         edges: [... edgeData.values()].map(x => ({
240             id: x.edgeData.id,
241             from: x.edgeData.from,
242             to: x.edgeData.to,
243             prob: x.prob
244         }))
245     };
246 }
```

```
247
248 function importGraphData(g) {
249     nodeData = new Map();
250     edgeData = new Map();
251     var nodeSet = new vis.DataSet({});
252     var edgeSet = new vis.DataSet({});
253     for(let node of g.nodes) {
254         var music = Music.fromObject(node.music);
255         var data = { id: node.id, label: music.nodeText() };
256         nodeData = nodeData.set(node.id, { nodeData: data, music: node.music });
257         nodeSet.add(data);
258     }
259
260     for(let edge of g.edges) {
261         var data = {
262             id: edge.id,
263             from: edge.from,
264             to: edge.to,
265             label: `${edge.prob * 100}%`
266         };
267         edgeData = edgeData.set(edge.id, { edgeData: data, prob: edge.prob });
268         edgeSet.add(data);
269     }
270
271     network.setData({ nodes: nodeSet, edges: edgeSet });
272 }
273
274 // helper
275
276 function download(url, filename) {
277     var link = document.createElement('a');
278     link.setAttribute('href', url);
279     link.setAttribute('download', filename);
280     link.style.display = 'none';
281     document.body.appendChild(link);
282     link.click();
283     document.body.removeChild(link);
284 }
285
286 function downloadFile(content_type, filename, content) {
287     var data = `data:${content_type},${encodeURIComponent(content)}`;
288     download(data, filename);
289 }
290
291
292 // graph code
293
294 var nodeData = Map();
295 var edgeData = Map();
296 var network = null;
297 var starting_node_id = null;
298
299
300 function showOverlay(id) {
301     document.getElementById(id).classList.remove('hidden');
302 }
303
304 function genericEditNode(data, callback) {
305     function clearOverlay() {
306         document.getElementById('node-save').onclick = null;
307         document.getElementById('node-cancel').onclick = null;
308         hideOverlay('node-overlay');
```

```
309     }
310
311     function saveNode(data, callback) {
312         var duration = new Rational(document.getElementById('numerator').value,
313             document.getElementById('denominator').value);
314         var music = new Music(duration, document.getElementById('pitch').value,
315             Number(document.getElementById('octave').value));
316         data.label = music.nodeText();
317         clearOverlay();
318         callback(data);
319         nodeData = nodeData.set(data.id, { music: music, nodeData: data });
320     }
321
322     function discardNode(callback) {
323         clearOverlay();
324         callback(null);
325     }
326
327     showOverlay('node-overlay');
328     var node = nodeData.get(data.id);
329     if(node !== undefined) {
330         var music = node.music;
331         document.getElementById('pitch').value = music.pitch;
332         document.getElementById('octave').value = music.octave;
333         document.getElementById('numerator').value = music.dur.numerator;
334         document.getElementById('denominator').value = music.dur.denominator;
335     }
336     document.getElementById('node-save').onclick = saveNode.bind(this, data, callback);
337     document.getElementById('node-cancel').onclick = discardNode.bind(this, callback);
338 }
339
340 function genericEditEdge(data, callback) {
341     function clearOverlay() {
342         document.getElementById('edge-save').onclick = saveEdge.bind(this, data, callback);
343         document.getElementById('edge-cancel').onclick = discardEdge.bind(this, callback);
344         hideOverlay('edge-overlay');
345     }
346
347     function saveEdge(data, callback) {
348         // for some reason, editWithoutDrag
349         // sets from & to to the node respective
350         // node objects, which results in the edge
351         // disappearing.
352         if (typeof data.to === 'object')
353             data.to = data.to.id
354         if (typeof data.from === 'object')
355             data.from = data.from.id
356
357         var prob = document.getElementById('prob').value / 100;
358         data.label = `${prob * 100}%`;
359         clearOverlay();
360         callback(data);
361         edgeData = edgeData.set(data.id, { prob: prob, edgeData: data } );
362     }
363
364     function discardEdge(callback) {
365         clearOverlay();
366         callback(null);
367     }
368
369     showOverlay('edge-overlay');
370     var edge = edgeData.get(data.id);
```

```
371     if(edge !== undefined) {
372         document.getElementById('prob').value = edge.prob * 100;
373     }
374     document.getElementById('edge-save').onclick = saveEdge.bind(this, data, callback);
375     document.getElementById('edge-cancel').onclick = discardEdge.bind(this, callback);
376 }
377
378 function deleteFromMap(data, callback) {
379     for(let node of data.nodes) {
380         nodeData = nodeData.delete(node);
381     }
382
383     for(let edge of data.edges) {
384         edgeData = edgeData.delete(edge);
385     }
386
387     callback(data);
388 }
389
390
391 function hideOverlay(id) {
392     document.getElementById(id).classList.add('hidden');
393 }
394
395 function handleImport() {
396     var files = document.getElementById('upload-score').files;
397     if(files.length === 0) {
398         alert('Select a file first!');
399     } else {
400         var file = files[0];
401         var reader = new FileReader();
402         reader.addEventListener('loadend', function() {
403             var parsed = JSON.parse(this.result);
404             if(parsed === undefined) {
405                 alert('Could not parse likely score');
406             } else {
407                 var confirmation = window.confirm('Proceeding will overwrite the current graph
                        . Are you sure?');
408                 if(confirmation) {
409                     try {
410                         importGraphData(parsed);
411                     } catch(e) {
412                         alert(`Could not import likely score, probably the file was malformed.
                                Error: ${e}`);
413                     }
414                 }
415             }
416         });
417         reader.readAsText(file);
418     }
419 }
420
421 function saveDataToLocalStorage() {
422     const json = JSON.stringify(collectGraphData(nodeData, edgeData));
423     const params = JSON.stringify(gatherParams());
424     localStorage.setItem("score", json)
425     localStorage.setItem("params", params)
426 }
427
428 function showStartingNode() {
429     if(typeof starting_node_id === 'string') {
430         network.selectNodes([starting_node_id], false);
```

```
    } else {
        alert('No starting node selected yet!');
    }
}

function setStartingNode() {
    var selected = network.getSelectedNodes();
    if(selected.length > 1) {
        alert('Only select one node!');
    } else if(selected.length === 0) {
        alert('Select a node first!');
    } else {
        starting_node_id = selected[0];
    }
}

function fetchInterpretation(params, format) {
    var jsonRequest = JSON.stringify({
        graph: collectGraphData(nodeData, edgeData),
        params: params
    });

    var myHeaders = new Headers();
    myHeaders.set('Content-Type', 'application/json');

    var myInit = {
        method: 'POST',
        headers: myHeaders,
        mode: 'cors',
        body: jsonRequest
    };

    var myRequest = new Request(`/interpretation/${format}`, myInit);

    return fetch(myRequest).then(res => res.blob());
}

function gatherParams() {
    var starting_node_entry = nodeData.get(starting_node_id);
    if(starting_node_entry !== undefined && starting_node_entry !== null) {
        var starting_node = {
            id: starting_node_entry.nodeData.id,
            music: starting_node_entry.music
        };
    } else {
        var starting_node = null
    }

    var maxhops = document.getElementById('hop-count').value;
    if(maxhops === "" || Number(maxhops) === NaN) {
        maxhops = null;
    } else {
        maxhops = Number(maxhops);
    }

    var seed = document.getElementById('seed').value;
    if(seed === "" || Number(seed) === NaN) {
        seed = null;
    } else {
        seed = Number(seed);
    }

```

```
493     return {
494         maxhops:  maxhops,
495         starting_node: starting_node,
496         seed: seed
497     };
498 }
499 
500 function completeGatherParams() {
501     var p = gatherParams();
502     if(p.starting_node === null) {
503         alert('Set a starting node first!');
504         return null;
505     }
506 
507     if(p.maxhops === null) {
508         alert('Set the maximum amount of hops to a valid number');
509         return null;
510     }
511 
512     if(p.seed === null) {
513         // TODO auto generate a random one, let the user confirm before
514         alert('Set the seed to a valid number!');
515         return null;
516     }
517 
518     return p;
519 }
520 
521 function importParams(p) {
522     if(p.starting_node !== null) {
523         starting_node_id = p.starting_node.id;
524     }
525     if(p.seed !== null) {
526         document.getElementById('seed').value = p.seed;
527     }
528     if(p.maxhops !== null) {
529         document.getElementById('hop-count').value = p.maxhops;
530     }
531 }
532 
533 function randomSeed() {
534     if(window.crypto) {
535         var array = new Int32Array(1);
536         window.crypto.getRandomValues(array);
537         document.getElementById('seed').value = array[0];
538     }
539 }
540 
541 function downloadInterpretation(format) {
542     var params = completeGatherParams();
543     if(params != null) {
544         try {
545             fetchInterpretation(params, format).then(file => {
546                 var url = URL.createObjectURL(file);
547                 download(url, `export.${format}`);
548                 URL.revokeObjectURL(url);
549             });
550         } catch(e) {
551             alert('An error occured while contacting the API: ' + e);
552         }
553     }
554 }
```

```
555
556 function reloadPlayer() {
557     var params = completeGatherParams();
558     if(params !== null) {
559         if(document.getElementById('player').src) {
560             URL.revokeObjectURL(document.getElementById('player').src);
561         }
562
563         document.getElementById('player').src = null;
564
565         try {
566             fetchInterpretation(params, 'wav').then(file => {
567                 var url = URL.createObjectURL(file);
568                 document.getElementById('player').src = url;
569             });
570         } catch(e) {
571             alert('An error occured while contacting the API: ' + e);
572         }
573     }
574 }
575
576 function init() {
577     var container = document.getElementById('network');
578
579     var options = {
580         manipulation: {
581             addNode: function(nodeData, callback) {
582                 document.getElementById('node-operation').innerHTML = 'Add';
583                 genericEditNode(nodeData, callback);
584             },
585             addEdge: function(edgeData, callback) {
586                 document.getElementById('edge-operation').innerHTML = 'Add';
587                 genericEditEdge(edgeData, callback);
588             },
589             editNode: function(nodeData, callback) {
590                 document.getElementById('node-operation').innerHTML = 'Edit';
591                 genericEditNode(nodeData, callback);
592             },
593             editEdge: {
594                 editWithoutDrag: function(edgeData, callback) {
595                     document.getElementById('edge-operation').innerHTML = 'Edit';
596                     genericEditEdge(edgeData, callback);
597                 }
598             },
599             deleteNode: deleteFromMap,
600             deleteEdge: deleteFromMap,
601             controlNodeStyle: {
602             }
603         },
604         nodes: {
605             borderWidth: 0,
606             color: {
607                 background: '#563d7c',
608                 hover: {
609                     background: '#8f14ff'
610                 },
611                 highlight: {
612                     background: '#8f14ff'
613                 }
614             },
615             chosen: true,
616             font: {
```

```
                color: 'white',
                size: 20,
                align: 'center'
            },
            shape: 'circle',
        },
        edges: {
            arrows: {
                to: { enabled: true }
            },
            color: {
                color: '#563d7c',
                hover: '#563d7c',
                highlight: '#563d7c',
            },
            font: {
                color: '#ffffff',
                strokeWidth: 0
            }
        }
    };

    network = new vis.Network(container, {}, options);

    try {
        const score = localStorage.getItem('score');
        if(score !== null) {
            importGraphData(JSON.parse(score));
        }
    } catch(e) {
        localStorage.removeItem('score');
    }

    try {
        const params = localStorage.getItem('params')
        if(params !== null) {
            importParams(JSON.parse(params));
        }
    } catch(e) {
        localStorage.removeItem('params');
    }

    const pitch_selector = valid_pitches.map((p, i) =>
        `<option value="${p}">${display_pitches[i]}</option>`)
        .reduce((acc, v) =>
            acc + v, '');
    document.getElementById('pitch').innerHTML = pitch_selector;

    /* event handling, order as in sidebar */
    document.getElementById('set-starting-node').onclick = setStartingNode;
    document.getElementById('show-starting-node').onclick = showStartingNode;

    document.getElementById('random-seed').onclick = randomSeed;

    document.getElementById('reload-player').onclick = reloadPlayer;
    document.getElementById('download-audio').onclick = () => {
        var format = document.getElementById('format').value;
        downloadInterpretation(format);
    };

    document.getElementById('gen-score').onclick = () =>
        downloadFile('application/json', 'score.likely.json',
```

```
679             JSON.stringify(collectGraphData(nodeData, edgeData)));
680     document.getElementById('upload-score').addEventListener('change',handleImport);
681     document.getElementById('clear-score').onclick = () =>
682         importGraphData({ nodes: [], edges: []});
683 
684     window.setInterval(saveDataToLocalStorage, 5000);
685 }
686 
687 document.addEventListener('DOMContentLoaded', () => init());
```

Graph im JSON Format der Webapplikation


```
{
  "nodes": [
    {
      "id": "d3c408d5-1ebb-4787-b510-22af5fe7093a",
      "music": {
        "dur": {
          "numerator": 3,
          "denominator": 4
        },
        "pitch": "Cf",
        "octave": 1
      }
    },
    {
      "id": "180159e7-527b-4b8a-b9b6-315dddc154d2",
      "music": {
        "dur": {
          "numerator": 2,
          "denominator": 4
        },
        "pitch": "C",
        "octave": 1
      }
    },
    {
      "id": "02e24c99-780e-45da-bd2f-ea600e4d863f",
      "music": {
        "dur": {
          "numerator": 1,
          "denominator": 1
        },
        "pitch": "Rest",
        "octave": 1
      }
    },
    {
      "id": "b9cd3f9d-134c-4c51-b325-d209b2529bd6",
      "music": {
        "dur": {
          "numerator": 1,
          "denominator": 8
        },
        "pitch": "F",
        "octave": 1
      }
    }
  ],
  "edges": [
    {
      "id": "f8d0cb23-00d1-49dd-961a-2114b8a89c1d",
      "from": "d3c408d5-1ebb-4787-b510-22af5fe7093a",
      "to": "180159e7-527b-4b8a-b9b6-315dddc154d2",
      "prob": 1
    },
    {
      "id": "283100d9-42ee-4001-b100-45b8c766cfc5",
      "from": "b9cd3f9d-134c-4c51-b325-d209b2529bd6",
      "to": "02e24c99-780e-45da-bd2f-ea600e4d863f",
      "prob": 0.8
    },
    {
```

```
      "id": "e6cceb76-40ed-49ac-8925-4534cf0854de",
      "from": "02e24c99-780e-45da-bd2f-ea600e4d863f",
      "to": "d3c408d5-1ebb-4787-b510-22af5fe7093a",
      "prob": 0.2
    },
    {
      "id": "0045bfda-3cde-4691-81c0-7a967be51e02",
      "from": "02e24c99-780e-45da-bd2f-ea600e4d863f",
      "to": "180159e7-527b-4b8a-b9b6-315dddc154d2",
      "prob": 0.8
    },
    {
      "id": "ec616a31-7fc0-4f27-ae31-79cf0fab224a",
      "from": "b9cd3f9d-134c-4c51-b325-d209b2529bd6",
      "to": "180159e7-527b-4b8a-b9b6-315dddc154d2",
      "prob": 0.2
    },
    {
      "id": "14735fda-b8e5-4567-aa1c-de04cc08ac24",
      "from": "180159e7-527b-4b8a-b9b6-315dddc154d2",
      "to": "b9cd3f9d-134c-4c51-b325-d209b2529bd6",
      "prob": 1
    }
  ]
}
```

Lizenz

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

"This License" refers to version 3 of the GNU Affero General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

   A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

   The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for

you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

    Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A

PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a computer network, you should also make sure that it provides a way for users to get its source. For example, if your program is a

web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see `http://www.gnu.org/licenses/`.