Modélisation et visualisation de schémas d'analyse musicale avec music21

Guillaume Bagan, Mathieu Giraud, Richard Groult, Emmanuel Leguy

MODÉLISATION ET VISUALISATION DE SCHÉMAS D'ANALYSE MUSICALE AVEC MUSIC21

Guillaume Bagan [1] *Mathieu Giraud* [1] *Richard Groult* [2] *Emmanuel Leguy* [1]
[1] CRIStAL, UMR 8189, CNRS, Université de Lille
[2] MIS, Université de Picardie Jules-Verne, Amiens
{guillaume, mathieu, richard, emmanuel}@algomus.fr

RÉSUMÉ

Comment modéliser et représenter une analyse musicale de musique notée sur des partitions ? Traditionnellement, il est d'usage d'*annoter les partitions*, en utilisant plusieurs symboles graphiques. Cet article présente une formalisation du concept de *schéma d'analyse*, vu comme un ensemble de *lignes d'analyse* groupant des *étiquettes* (ou *labels*). Les étiquettes peuvent encoder des informations d'occurrence de motifs, des sections, ou d'autres événements musicaux. Les schémas sont créés par un analyste, ou, dans le cas de la recherche en analyse musicale par ordinateur (computational music analysis, CMA), calculés de manière algorithmique. Nous proposons plusieurs méthodes de visualisation de ces schémas représentés comme des objets `music21` : sur partition Lilypond, image, via une page web interactive avec des extraits musicaux produits par music21j et Vexflow, ou enfin sur une vidéo produite par ly2video. Les schémas que nous présentons ici ont été conçus dans l'objectif de rendre une analyse claire et compréhensible, à destination du musicologue ou du musicien, mais aussi à destination de l'amateur de musique, même non lecteur de partition.

1. INTRODUCTION

Analyser une partition, c'est apporter un éclairage sur différents points de vue (motifs, mélodies, thèmes, harmonie, sections) et renouveler son écoute sur une pièce de musique [8, 9]. L'analyse est une activité essentielle de l'interprète, de l'auditeur tout comme du musicologue : celui qui joue, entend ou étudie une pièce le fait avec sa propre appréhension de la musique qu'on peut chercher à formaliser [14].

Comment modéliser et représenter une analyse musicale de musique notée ? Il est certes difficile de modéliser un commentaire esthétique, mais certains *éléments d'analyse* peuvent être formalisés : thèmes et motifs, harmonie et cadences, structuration... Traditionnellement, l'analyste *annote les partitions* avec plusieurs symboles graphiques. On peut ainsi voir cette analyse comme un ensemble de calques qui regrouperaient chacun un ensemble de symboles concernant une facette particulière de l'analyse.

Plusieurs suites logicielles permettent déjà de calculer des éléments d'analyse et/ou de les visualiser sur des partitions.

Analyse automatique. Humdrum est un ensemble de scripts pour traiter le format `.krn`, dédié à l'analyse musicologique [22]. Plus récemment, music21 [12], une bibliothèque python, propose une boite à outils pour développer des logiciels d'analyse musicale. music21 manipule des données musicales représentées sous forme symbolique, et comporte de nombreuses fonctions d'import (MusicXML, `.krn`, MIDI, ...), d'analyse (motifs, ambitus, tonalités...) et d'export (y compris vers Lilypond).

Enfin, on peut mentionner certaines tentatives de formats d'annotation interopérables, comme JAMS (JSON Annotated Music Specification) [21] qui est plutôt destiné à l'annotation de fichiers audio.

Visualisation de partitions. De nombreux logiciels de gravure musicale existent. Si l'on se concentre sur les logiciels prenant en entrée une notation symbolique et produisant des partitions, on peut tout d'abord citer le logiciel libre Lilypond [4], développé depuis 20 ans et se caractérisant par une très grande qualité typographique ainsi que par une excellente flexibilité. VexFlow [6] est une bibliothèque Javascript pour afficher du contenu musical. Guido [20] propose une bibliothèque C/C++ cross-plateforme permettant la mise en page et le rendu de partitions. Elle est aussi accessible via un service web REST pour générer en ligne l'affichage de partitions.

Certains logiciels permettent de transformer une partition en vidéo. Nous détaillerons dans cet article ly2video (voir section 3.4). On peut mentionner aussi certaines applications commerciales comme NoteZilla [5], construit lui aussi sur Lilypond, qui permet de visualiser une partition animée avec certaines interactions comme la navigation et le zoom, sans notion d'analyse.

Édition et visualisation. INScore [3, 17] est un éditeur de partition augmentée. Il est capable d'animer une partition et de lui ajouter des éléments graphiques (curseur, annotations...). L'animation est considérée comme une « partition interactive » car elle peut être modifiée en temps réel

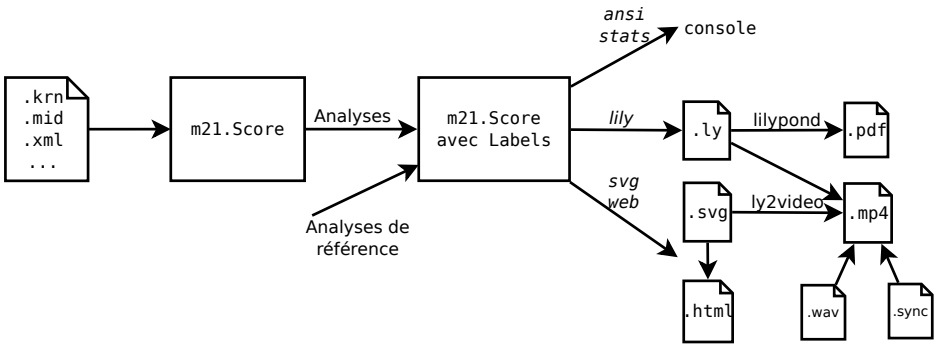


Figure 1. Modélisation et visualisation de schémas d'analyse avec music21. Nous avons étendu music21 en proposant un « schéma d'analyse », qui est une partition music21 (`Score`) annotée avec des étiquettes (`Label`). La partition peut avoir été créée à partir d'un fichier d'entrée (`.krn`, `.mid`, `.xml` ...), puis analysée avec les fonctionnalités de music21 ou de n'importe quel autre programme. On peut aussi considérer des analyses manuelles, par exemple pour des analyses de référence. Les visualisations de ces schémas (dans une console, partition `.pdf`, schéma `.svg`, page web `.html` et vidéo `.mp4`) sont décrites dans les différentes sections de cet article.

en interagissant soit via un panneau de contrôle (modification de tempo par exemple) soit directement en interprétant une pièce musicale. INScore est interfaçable en python et en pure data via OSC (Open Sound Control).

Le logiciel iAnalyse d'aide à l'analyse musicale est destiné à la présentation, à l'annotation et à l'analyse musicale [11]. Il permet de synchroniser puis d'annoter manuellement des fichiers images de partitions, et de produire à partir de ces annotations une vidéo de l'analyse. Les annotations sont ici des symboles graphiques et ne sont pas liées de manière logicielle aux notes de la musique sous-jacente.

Le projet "Écoutes signées" a exploré de nombreuses voies de représentation, de manipulation et de visualisation de partitions avec leurs annotations, en créant des "maquettes" transmettant une "manière d'écouter" particulière [15, 14].

Enfin, certaines plateformes d'analyse sont destinées à étudier de la musique électro-acoustique, comme l'Acousmographe [16], EAnalysis [7] et TIAALS [10].

L'activité centrale de l'équipe Algomus est de mettre en œuvre des analyses informatisées de musique tonale, notée sur des partititons. Comment rendre l'ordinateur capable d'accomplir cette tâche et surtout quelles sont ses limites ? L'ordinateur doit aussi être capable de transmettre ses résultats d'analyses à un humain. Il est donc nécessaire de disposer d'une suite logicielle qui couvre l'ensemble du processus automatique allant d'une partition formalisée (`.krn`, MIDI) jusqu'à la visualisation de l'analyse. Certains des logiciels que nous venons de citer permettent de couvrir une partie de la chaîne. Nous avons souhaité pouvoir valider nos algorithmes en les comparant de manière statistique ou graphique et cela a nécessité des développements spécifiques menant à la présente contribution.

Cette article présente une formalisation du concept de *schéma d'analyse*, vu comme ensemble de *lignes d'analyse* groupant des *étiquettes*. Les étiquettes peuvent encoder des informations d'occurrence de motifs, des sections, ou d'autres événements musicaux. Les schémas que nous présentons ici ont été conçus dans l'objectif de rendre une analyse claire et compréhensible, à destination du musicologue ou du musicien, mais aussi à destination de l'amateur de musique, même non lecteur de partition.

La prochaine section présentent notre modèle de *schéma d'analyse*, le principe de notre implémentation dans la bibliothèque python `music21`, et le moyen de créer ces schémas. La FIGURE 1 montre une vue d'ensemble de l'utilisation de ces schémas. Les sections suivantes détaillent différentes visualisations de ces schémas : sur partition, sous forme d'image vectorielle, via une page web interactive, ou sur une vidéo. Nous concluons en expliquant comment ces schémas servent dans l'évaluation de méthodes algorithmiques et en décrivant la diffusion de notre code.

2. MODÉLISATION ET CRÉATION DES SCHÉMAS

Nous détaillons ici comment formaliser certains éléments d'analyse musicale.

2.1. Étiquettes, lignes et schémas

Nous proposons d'utiliser les éléments suivants :

— l'*étiquette* est l'élément analytique de base, correspondant à une annotation graphique sur une partition. Cette étiquette peut avoir une durée (motif, section, pédale) ou non (point d'arrivée d'une cadence, autre événement).

— plusieurs étiquettes peuvent se regrouper dans une même *ligne* d'analyse. Une ligne représente un objectif particulier d'analyse, et peut s'imaginer comme un ensemble symboles graphiques qui seraient sur un

```
score.id = "Exemple"

score.parts[0].id = "voix 1"
score.parts[0].insert(Label(offset=2, duration=4, kind="pattern-a", tag="a"))
score.parts[0].insert(Label(offset=9, duration=4.5, kind="pattern-b", tag="b"))

score.parts[1].id = "voix 2"
score.parts[1].insert(Label(offset=14, duration=4, kind="pattern-a", tag="a'")

part_cad = music21.stream.Part()
part_cad.id = "cadence"
score.insert(2, part_cad)
part_cad.insert(Label(offset=19, kind="cad", tag="I:PAC"))
```

Figure 2. Création d'un schéma en python/music21. Dans la première voix (`score.parts[0]`), deux `Label` sont insérés aux temps 2 et 9. Dans la seconde voix (`score.parts[1]`), un `Label` est inséré au temps 14. Une `Part` est ensuite créée pour y insérer un `Label` indiquant une cadence (`cad`).

même calque. Une ligne peut être attachée à une voix de la partition (e.g étiquettes pour la voix soprano) ou être indépendante (l'ensemble des marches harmoniques, ou bien une structuration de la partition).

— un *schéma* est un ensemble de lignes d'analyse concernant une même pièce. Un schéma peut s'imaginer comme une superposition des calques des différentes lignes d'analyse. Le schéma peut représenter une analyse de référence ou bien une analyse produite par ordinateur.

2.2. Attributs des étiquettes

Type. Une étiquette possède un type (*kind*) particulier. Il pourra, par exemple, désigner un motif musical (type « pattern ») ou encore une cadence (type « cad »). Ce type est choisi librement à la création de l'étiquette et il est utilisé lors de la visualisation par les feuilles de style (voir Section 3.5).

Poids et annotation. Une étiquette peut comporter un poids numérique (*weight*). Un motif peut être trouvé avec une confiance plus ou moins grande. Enfin, certaines annotations dépendent du type de l'étiquette et prennent la forme d'une chaîne de caractères laissée libre (*tag*). Par exemple, les cadences sont annotées du type de la cadence (PAC/IAC/HC/DC, pour cadence parfaite parfaite, imparfaite, demi-cadence, et cadence rompue).

2.3. Implémentation et objets music21

Tout d'abord développés avec nos propres structures pour nos besoins de visualisation, ces concepts ont été implémentés au moyen d'objets python music21 [12]. Il s'agit donc d'annoter une partition qui est manipulable en music21 par l'objet `Score`. L'élément d'analyse de base, le `Label` (étiquette), dérive de l'objet music21 (`Music21Object`). En plus des attributs provenant de `Music21Object` (principalement un offset et une durée), il possède les attributs décrits au paragraphe précédent. Les `Label` sont regroupés en lignes d'analyse avec l'objet music21 `Part`. Nous distinguons deux types de lignes d'analyse :

— les lignes regroupant des étiquettes propres à une voix, comme la recherche d'un motif sur la soprano. La partition music21 (objet `Score`) contenant déjà un objet `Part` par voix, les `Label` y sont directement insérés.

— les lignes regroupant des étiquettes ne dépendant pas d'une seule voix, comme les tonalités locales ou la structure. De nouvelles `Part` sont crées et insérées dans le `Score`.

Ainsi un schéma d'analyse est une partition (objet `Score`) annotée par des étiquettes (objets `Label`) rangés dans des lignes d'analyse (objet `Part`). Créer un schéma d'analyse se fait donc en instanciant directement ces objets (voir FIGURE 2).

Un cas particulier est que l'objet `Score` peut être vide de tout contenu musical (notes) et ne contenir que des `Part` et des `Label`, par exemple dans le cas d'un schéma d'analyse réalisé manuellement sans avoir la partition complète sous forme informatique.

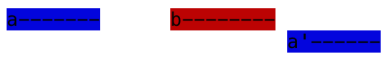
2.4. Création de schémas via des analyses de référence

Il n'existe pas une seule analyse correcte d'une pièce donnée – différentes approches musicologiques peuvent donner différentes analyses tout aussi pertinentes. Même des notions a priori simples (qu'est-ce qu'un thème, qu'est-ce qu'une cadence ?) sont souvent débattues. Cependant, sur de nombreux points, il peut y avoir consensus qui peut servir de vérité de terrain (« ground truth ») pour évaluer les algorithmes. Il est intéressant de pouvoir disposer de ces analyses de référence – ou mieux, de *plusieurs* analyses – sous forme informatique.

Les schémas d'analyse peuvent ainsi être donnés par un format texte encodant des analyses de référence (voir FIGURE 4). La syntaxe de ce format a été conçue pour faci-

```
out = ansi.AnsiSchema(score, myStyle)
print(out.render())
```

```
=== Exemple
    voix 1
    voix 2
   cadence
```




```
out = svg.SvgSchema(score, myStyle)
out.save('schema.svg')
```

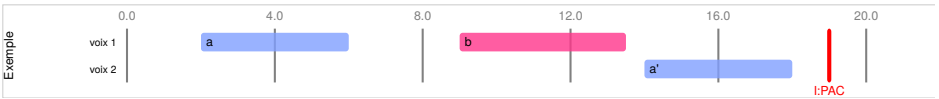


Figure 3. Visualisation du schéma défini par la FIGURE 2 dans le terminal, sous forme de texte coloré (haut), et par une image vectorielle SVG (bas).

liter une saisie musicologique (voir `www.algomus.fr/datasets`). Les repères temporels peuvent ainsi être entrés en mesure, temps, ou bien en fraction de mesures, ou en se servant de repères logiques. À ce jour, nous avons dans ces analyses de référence plus de 1 200 annotations (fugues, sonate, texture) encodées par 5 personnes [13, 19, 18]. Ces annotations recouvrent des indications de motifs, de structures, de cadences et de texture.

3. VISUALISATION DES SCHÉMAS

Les schémas d'analyse peuvent être visualisés par l'intermédiaire d'un schéma « de sortie ». On peut tout d'abord obtenir une sortie texte colorée affichée dans le terminal (haut de la FIGURE 3). Nous passons maintenant en revue les autres visualisations de ces schémas – partitions Lilypond, images SVG, pages web et vidéos.

3.1. Partition Lilypond

Un module permet de générer un fichier Lilypond (voir FIGURE 5). Chaque `Label` est représenté par une boite colorée ou bien par une barre verticale si sa durée est nulle. Le module `music21.lily` permet déjà de générer du code lilypond à partir d'une partition. Nous l'avons donc étendu pour pouvoir afficher les boites représentant un `Label`, en reprenant le `frameEngraver` écrit par David Nalesnik [2].

3.2. Image SVG

Un module permet de générer des images SVG (voir FIGURE 3, 6 et 7). Chaque `Label` est affiché suivant une procédure dépendant de son style. Cette sortie est l'une des bases pour les sorties web et vidéo.

Générer directement du texte SVG n'est pas très élégant et n'est pas pratique pour le débugage et pour les tests unitaires. De plus, le format SVG ne gère pas la profondeur des

```
==== wtc-i-02 BWV 847 ====
     C minor, 3 voices (SAT)

  == S [length 2 start +1/8]
        S  3, 11, 20, 29.5
        A  1, 15
        T  7, 26.5

  == CS1 [length 2 start +1/16]
        S  7 [start -1/4], 15 [start 0]
        A  3, 20
        T  11
        SA 26.5 [changeafter +1/2 end +1/8]

  == CS2 [length 1.5 start +1/8]
        S  27 [end +1/8]
        A  7.5, 11.5
        T  15.5, 20.5

  == cadences
        *  17 (v:PAC), 22 (i:rIAC)
        *  28.5 (i:IAC) [end +1/8], 29.5 (I:PAC)

  == pedals
        T  29.5 (I) [length 2.5]
```

Figure 4. Extrait de notre analyse de référence concernant la fugue en Do mineur BWV 847 de J.-S. Bach. Ce fichier décrit un sujet (S), deux contre-sujets (CS1 et CS2), des cadences et une pédale. Par exemple, les occurences du sujet sur la troisième voix (T) sont aux mesures 7 et 26.5. La syntaxe de ce format est décrite à l'adresse `www.algomus.fr/datasets`.

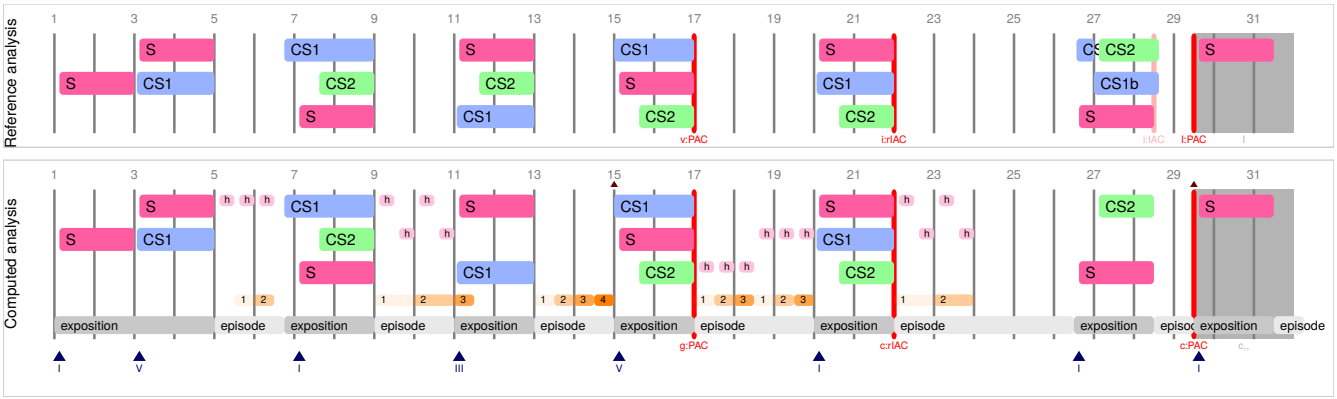


Figure 6. Visualisation en image vectorielle SVG de l'analyse de la fugue en Do mineur BWV 847 de J.-S. Bach, avec identification de sujets (S) et contre-sujets (CS), de marches harmoniques (petites étiquettes 1/2/3/4), de cadences (traits verticaux épais), d'une pédale à la fin de la partition, et détection d'une structure d'ensemble (ligne du bas). L'analyse obtenue ("Computed analysis", en bas) peut se comparer avec l'analyse de référence ("Reference analysis", en haut) [18].

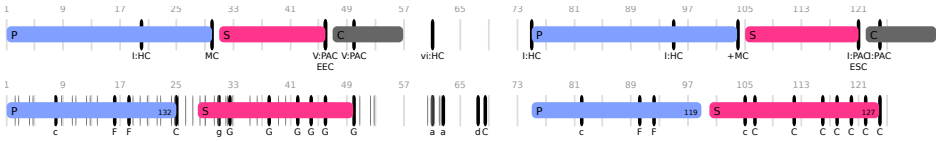


Figure 7. Structure en forme sonate, premier mouvement du quatuor à cordes K. 157 n° 4 de W. A. Mozart [13]. Les étiquettes représentent ici des parties (thème principal (P), secondaire (S) et conclusion (C)). Le schéma du haut est celui de l'analyse de référence, et celui de bas celui d'une analyse calculée. Les traits noirs représentent des fins de phrases potentielles, exceptés les plus épais qui représentent des cadences.

éléments. En effet, si deux éléments se superposent alors celui déclaré en dernier sera affiché au dessus de l'autre et il n'est pas possible de spécifier le contraire.

Nous avons donc utilisé une représentation intermédiaire. Un objet `SvgWriter` représente chaque élément SVG. Il possède un dictionnaire d'attributs et une liste d'enfants avec chacun une profondeur. La classe remet dans l'ordre les éléments enfants selon leur profondeur.

3.3. Visualisation web interactive

Nos outils permettent également de générer des pages web HTML5 dynamiques (voir FIGURE 8). Lorsque l'on clique sur une étiquette, la partition de celui-ci est affichée et lorsque l'on passe la souris sur une étiquette, la partition concernée est mise en évidence. De plus, la partition peut être écoutée si l'on clique dessus. Le module SVG est utilisé pour afficher le schéma. Les partitions de chaque étiquette sont générées avec la bibliothèque javascript `music21j`, qui utilise Vexflow pour ce rendu.

3.4. Vidéo

3.4.1. Une écoute augmentée

Il nous a paru essentiel de présenter les schémas dans un contexte d'écoute : jouer ou écouter une pièce en prenant conscience de sa structure est à la base de l'analyse musicale. Grâce à la vidéo, les schémas sont mis en mouvement et l'analyse devient presque naturelle. Cette « écoute augmentée » s'adresse à tous, lecteurs ou non.

Les plates-formes de vidéo à la demande sont une formidable ressource de divertissement mais permettent aussi de démocratiser la connaissance dans les domaines les plus variés. Être capable de produire des analyses sous format vidéo permet d'accéder à ce mode de diffusion.

3.4.2. Adaptation de ly2video

Quelques outils permettent déjà de réaliser manuellement des analyses musicales sous forme de vidéo. Étant capable de produire automatiquement nos visualisations, nous souhaitions naturellement faire de même pour les vidéos. Nous avons contribué au logiciel ly2video [1], logiciel développé par Jiří "FireTight" Szabó puis par Adam Spiers.

Une vidéo, générée à partir d'un fichier lilypond, permet de suivre la partition tout en l'écoutant. Le son pro-

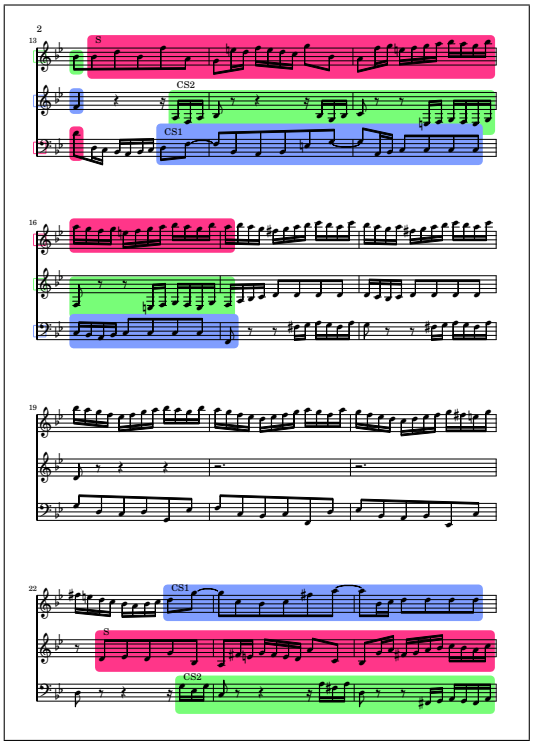


Figure 5. Extrait de la partition Lilypond générée à partir d'un schéma provenant de l'analyse de référence de la fugue en Sib majeur BWV 866 de J.-S. Bach.

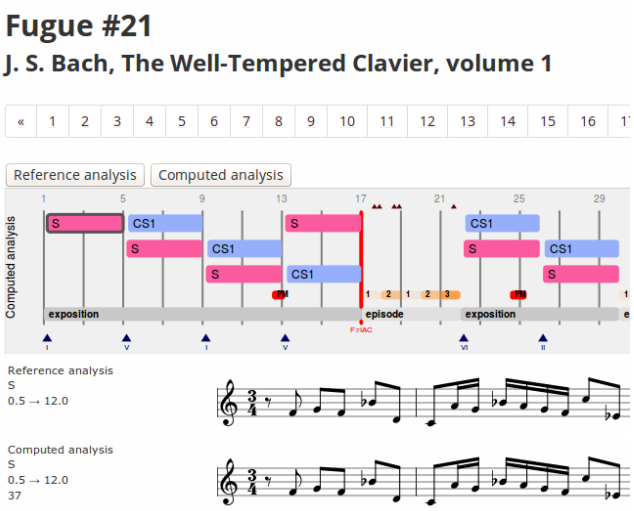


Figure 8. Visualisation interactive de la fugue en Sib majeur BWV 866 de J.-S. Bach (`www.algomus.fr/fugues`). Cliquer sur les étiquettes permet d'afficher et d'entendre les extraits de partition.

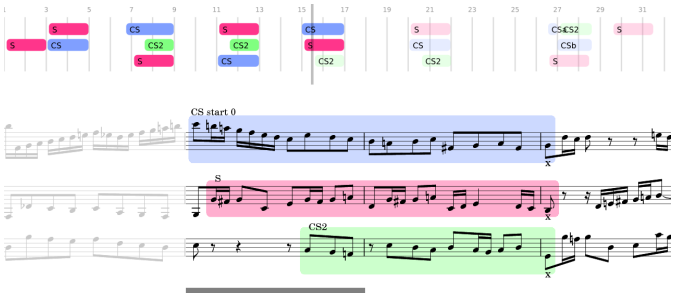


Figure 9. Extrait d'une vidéo produite par ly2video au milieu de la fugue en Do mineur BWV 847. La vidéo complète est disponible à l'adresse `algomus.fr/video`.

vient d'un flux MIDI ou d'un enregistrement réel. Le choix dans ly2video a été de présenter la partition comme un flux continu de notes tel que se fait naturellement l'écoute. La partition ne comporte donc ni reprise ni retour à la ligne, et deux modes de suivi de partition sont disponibles (FIGURE 10) :

1. La partition défile alors qu'un curseur de notes (ligne verticale) est placé au milieu de l'écran. Les notes n'étant pas positionnées proportionnellement, cette solution n'est pas toujours très fluide : elle occasionne quelques saccades, en particulier lors des changements de mesure. Nous avons donc ajouté la possibilité d'un *suivi par mesure* moins intrusif. Il se matérialise en atténuant la partie déjà écoutée de la partition.
2. Le flux de la partition est découpé en plusieurs fenêtres affichées successivement. Le curseur se déplace alors de notes en notes sur l'extrait courant. Cette option a l'avantage de reposer les yeux, chaque extrait étant fixe, et permet aussi un suivi par mesure.

Il est maintenant possible de découper l'écran en plusieurs zones et d'intégrer plusieurs animations. Les schémas d'analyse sont ainsi ajoutés sous forme de diaporama en utilisant des images SVG construites au fur et à mesure (FIGURE 9). Une nouvelle image est générée pour tout nouvel évènement musical.

3.4.3. Synchronisation

La synchronisation effectuée par ly2video consiste à relier plusieurs éléments dynamiques : le schéma, la partition et l'audio. Pour produire une vidéo fluide (25 images par seconde par défaut), il est nécessaire de déterminer le nombre d'images (frames) à générer entre deux notes contiguës et surtout lesquelles.

Les notes (onsets) constituent le repère commun aux trois médias. Les synchroniser consiste alors à faire correspondre ce temps musical avec un espace graphique ou un signal sonore (lui-même considéré comme espace graphique par [17]).

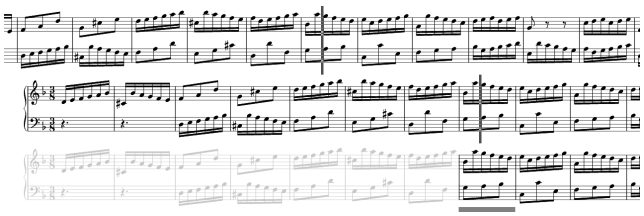


Figure 10. ly2video peut produire des vidéos contenant plusieurs zones synchronisées. Ici chaque zone montre une manière différente de faire défiler la partition : curseur fixe (haut), curseur mobile (milieu), curseur de mesure mobile (bas).

— Les schémas d'analyse sont pour l'instant graphiquement réguliers (le curseur parcourt toujours la même distance pendant une blanche).

— La partition est, quant à elle, irrégulière, les notes n'étant pas disposées proportionnellement. On doit ainsi prendre en compte le « trajet » que doit effectuer le curseur pour aller d'une note à la suivante.

— Le son, s'il est généré à partir d'un fichier lilypond via un fichier MIDI, est naturellement synchronisé. Lorsqu'il s'agit d'un enregistrement réel, la synchronisation s'effectue manuellement en réalisant un fichier `.sync` localisant tous les temps.

La synchronisation est implémentée par un design pattern *Observer*. À chaque nouvel onset, un objet central en notifie tous les médias. Chaque média change alors son apparence si besoin, en gérant éventuellement l'avancée du curseur, changement de fenêtre, ou l'ajout d'une étiquette.

3.5. Styles

Toutes les visualisations des schémas utilisent une feuille de style hiérarchique (FIGURE 11). Chaque élément (lignes et étiquettes) d'un schéma est traduit graphiquement, pour les sorties proposées (texte, lilypond, svg, html), en fonction de son *type*. Par exemple, il est possible de fixer une hauteur et une couleur pour toutes les étiquettes de type `pattern`.

Les propriétés graphiques peuvent être factorisées par des relations d'héritage entre deux types. Un motif particulier, tel que le sujet d'une fugue, pourra ainsi avoir la hauteur définie par défaut pour tous les motifs mais une couleur différente, comme pour le type `pattern-a` de la FIGURE 11.

Chaque visualisation possède ses propres contraintes. Par exemple, la hauteur d'un bloc dans le SVG (et donc dans la page web et la vidéo) n'a pas de signification pour la sortie lilypond, où la hauteur doit s'accorder avec la partition. Cependant, plusieurs paramètres sont conservés pour toutes les visualisations, tels que la couleur et les polices, et l'héritage entre éléments.

```
myStyle = style.StyleSheet()


myStyle.addStyle('pattern',
    {'svg': svg.Box, 'opacity': 0.8,
     'fontSize': 12, 'boxHeight': 14})

myStyle.addStyle('pattern-a',
    {'color': Color(128, 159, 255, Back.BLUE)},
    parent='pattern')

myStyle.addStyle('pattern-b',
    {'color': Color(255, 53, 139, Back.RED)},
    parent='pattern')


myStyle.addStyle('cad',
    {'svg': svg.Point, 'color': colors.red})
```

Figure 11. Feuille de style utilisée pour obtenir la FIGURE 3. Un style d'affichage pour chaque type de `Label` (`pattern` et `cad`) est défini.

4. CONCLUSION, DIFFUSION, PERSPECTIVES

Nous proposons une modélisation et plusieurs visualisations de schémas d'analyse pour l'étude de musique notées sur partitions grâce à la bibliothèque python music21. Les schémas servent à représenter facilement certains éléments d'une analyse mais aussi à comparer des analyses entre elles, par exemple lors de l'évaluation d'algorithmes d'analyse musicale (FIGURE 12).

L'ensemble du code décrit dans cet article est développé en licence libre (GPLv3+). Nous cherchons lorsque c'est possible à faire intégrer ce code aux projets open-source existants. C'est déjà le cas pour notre contribution à ly2video, et nous sommes en train de discuter avec l'équipe de music21 concernant notre module `music21.schema`. Une partie de ce code est déjà librement téléchargeable [1] . Enfin, afin de favoriser des évaluations d'algorithmes d'analyse, les fichiers de vérité réalisés par l'équipe sont eux aussi distribués sous une licence libre. À ce jour, sont ainsi disponibles des fichiers décrivant la structure fugues ainsi que des annotations de texture (1 200 annotations).

Nous espérons que ces outils permettront à d'autres de visualiser et de comparer leur analyses et leurs annotations de partitions musicales.

1. `git.algomus.fr`

```
Comparing analysis on 8 pieces, startDeltaOffset=2.0

a ==>  TP:  74  FP:  25  FN:   2    sens: 74/ 76   97.4%    prec: 74/ 99   74.7%    F1: 0.846
b ==>  TP:  36  FP:   3  FN:  25    sens: 36/ 61   59.0%    prec: 36/ 39   92.3%    F1: 0.720
c ==>  TP:  6   FP:   5  FN:  27    sens:  6/ 33   18.2%    prec:  6/ 11   54.5%    F1: 0.273
d ==>  TP:  2   FP:   0  FN:   0    sens:  2/  2  100.0%    prec:  2/  2  100.0%    F1: 1.000
```

Figure 12. Exemple de statistiques calculées lors de la comparaison de schémas entre une analyse calculée et une analyse de référence prise comme « ground truth ». Ces analyses sont effectuées sur plusieurs pièces d'un corpus. Au total, l'étiquette `a` possède 76 occurrences dans l'analyse de référence et 99 dans l'analyse calculée : 74 vrais positifs, 25 faux positifs et 2 faux négatifs, soit une sensibilité de 97,4% et une précision de 74,7%. Une étiquette est considérée comme vrai positif si son offset est à 2 noires (`startDeltaOffset`) d'une étiquette de même type dans l'analyse de référence.

5. REFERENCES

[1] Adam Spiers, ly2video. `https://github.com/aspiers/ly2video`.

[2] David Nalesnik, FrameEngraver. `http://lists.gnu.org/archive/html/lilypond-user/2013-07/msg00554.html`.

[3] INScore – an interactive augmented music score. `http://inscore.sourceforge.net/`.

[4] LilyPond – Music notation for everyone. `http://www.lilypond.org/`.

[5] Notezilla – Beautiful sheet music synced to high quality recordings. `http://www.notezilla.io/`.

[6] VexFlow – HTML5 Music Engraving. `http://www.vexflow.com/`.

[7] EAnalysis : aide à l'analyse de la musique électroacoustique. In *Journées d'Informatique Musicale (JIM 2012)*, pages 183–189, 2012.

[8] Ian Bent and William Drabkin. *Analysis*. 1987.

[9] Rémy Campos and Nicolas Donin. *L'analyse musicale : une pratique et son histoire*. Droz, 2009.

[10] Michael Clarke, Frédéric Dufeu, and Peter Manning. TIAALS : A new generic set of tools for the interactive aural analysis of electroacoustic music. In *Electroacoustic Music Studies Network (EMS 2013)*, 2013.

[11] Pierre Couprie. iAnalyse : un logiciel d'aide à l'analyse musicale. In *Journées d'Informatique Musicale (JIM 2008)*, pages 115–121, 2008.

[12] Michael Scott Cuthbert and Christopher Ariza. music21 : A toolkit for computer-aided musicology and symbolic music data. In *Int. Society for Music Information Retrieval Conf. (ISMIR 2010)*, 2010.

[13] Laurent David, Mathieu Giraud, Richard Groult, Florence Levé, and Corentin Louboutin. Vers une analyse automatique des formes sonates. In *Journées d'Informatique Musicale (JIM 2014)*, 2014.

[14] Nicolas Donin. Manières d'écouter des sons. quelques aspects du projet Écoutes signées (Ircam). *DEMéter*, 2013.

[15] Nicolas Donin, Samuel Goldszmidt, and Jacques Theureau. Instrumenter les opérations d'écoute analytique ? un bilan du projet "écoutes signées" (2003-2006). In *Journées d'Informatique Musicale (JIM 2010)*, pages 165–174, 2010.

[16] Emmanuel Favreau, Yann Geslin, and Adrien Lefèvre. L'acousmographe 3. In *Journées d'Informatique Musicale (JIM 2010)*, 2010.

[17] D. Fober, C. Daudin, Y. Orlarey, and S. Letz. Partitions musicales augmentées. In *Journées d'Informatique Musicale (JIM 2010)*, 2010.

[18] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. Computational fugue analysis. *Computer Music Journal*, 39(2), 2015.

[19] Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière, and Donatien Thorez. Modeling texture in symbolic data. In *Int. Society for Music Information Retrieval Conf. (ISMIR 2014)*, 2014.

[20] H. H. Hoos, K. A. Hamel, K. Renz, and J. Kilian. The GUIDO music notation format. In *Int. Computer Music Conference (ICMC 1998)*, pages 451–454, 1998.

[21] Eric J. Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M. Bittner, and Juan P. Bello. JAMS : a JSON annotated music specification for reproducible MIR research. In *Int. Society for Music Information Retrieval Conf. (ISMIR 2014)*, 2014.

[22] David Huron. Music information processing using the Humdrum toolkit : Concepts, examples, and lessons. *Computer Music J.*, 26(2) :11–26, 2002.