Évaluation de la correction rythmique des partitions numérisées

Francesco Foscarin, Raphaël Fournier-S'Niehotta, Florent Jacquemard, Philippe Rigaux

ÉVALUATION DE LA CORRECTION RYTHMIQUE DES PARTITIONS NUMÉRISÉES

Francesco Foscarin
CEDRIC, CNAM, Paris
francesco.foscarin@cnam.fr

Raphaël Fournier-S'niehotta
CEDRIC, CNAM, Paris
fournier@cnam.fr

Florent Jacquemard
Inria, Paris
florent.jacquemard@inria.fr

Philippe Rigaux
CEDRIC, CNAM, Paris
philippe.rigaux@cnam.fr

Résumé

Nous proposons une représentation des éléments de gravure musicale ayant trait au rythme – figures de notes, tuplets, ligatures, *etc.* – sous la forme d'arbres. Cette modélisation permet de lever les ambiguités et redondances contenues dans les différents encodages de partitions numériques, qui sont des sources d'incohérences majeures.

Le modèle est développé théoriquement, puis nous présentons son intégration dans des procédures de vérification d'encodages. Il est également montré que l'on peut l'utiliser pour des tâches de génération de partitions.

1. INTRODUCTION

Aujourd'hui, produire des partitions musicales numérisées de bonne qualité demande un effort important. En effet, il faut pour cela inspecter à l'œil le rendu graphique après numérisation pour y dénicher les erreurs, ce qui s'avère difficile en raison de la complexité sémiologique de la notation musicale.

Dans le cadre de l'édition collaborative, ce problème est particulièrement important, puisque chaque contributeur est susceptible d'utiliser son propre logiciel de gravure. Les encodages de documents musicaux comme MusicXML [6] ou MEI [13, 10] sont largement utilisés comme formats d'échange, ainsi que pour le stockage dans des bases de partitions numériques dont le contenu peut être inspecté, modifié et indexé. Ces encodages XML sont très permissifs et n'effectuent pas de vérification de la correction, de la cohérence ou de la complétude d'une partition. Leur grande flexibilité est bien adaptée à la prise en compte des évolutions et variations de la notation musicale, mais en contre-partie, laisse aux humains une partie délicate et couteuse de la production de partitions.

Nous proposons dans cet article une modélisation d'éléments de gravure, en vue d'assister les créateurs de partitions dans des tâches de vérification des données encodées. Nous nous limitons à la partie rythmique de la notation. Les problématiques auxquelles nous apportons une réponse sont détaillées dans la section 2. Nous présentons ensuite notre modèle dans la section 3, des projets en cours autour de leur application à l'analyse de partitions (section 4) ainsi que des perspectives d'applications dans le cadre de tâches impliquant la génération de partitions numériques (section 5). Nous développons en particulier les contributions suivantes :

– un formalisme de modélisation des éléments rythmiques de notation musicale, reposant sur deux structures sous-jacentes complémentaires : les arbres de *ligature* et les arbres de *tuple* (pour les groupements de notes tels que les triolets) ;

– un développement de ce formalisme pour *valider* des partitions existantes, c'est-à-dire en éliminer les ambiguïtés et redondances présentes dans l'encodage XML ;

– une proposition de génération de partitions dans un format XML, à partir de ces structures d'arbres ;

– l'intégration de cette modélisation dans une plateforme en ligne proposant d'ores et déjà de l'analyse automatisée de la qualité de partitions musicales.

2. ÉLÉMENTS DE NOTATION RYTHMIQUE

Les encodages MusicXML et MEI permettent de décrire dans un même document à la fois des informations relatives au contenu musical d'une pièce, indépendamment de la manière dont ce contenu est rendu sur une partition imprimée, et des informations relatives à l'apparence graphique de ce contenu.

Par exemple en MusicXML la *durée* de chaque événement est décrite dans un élément XML `<duration>`, qui contient une valeur exprimée en fraction de noire (*quarter-length*), ou plus précisément en nombre de divisions élémentaires de la noire. Le nombre maximal de divisions d'une noire est par ailleurs spécifié en début de voix (avec l'élément `<divisions>`), suivant l'approche du standard MIDI. Par exemple, pour un

Francesco Foscarin, Florent Jacquemard, Raphaël Fournier-S'Niehotta, Philippe Rigaux, *Évaluation de la correction rythmique des partitions numérisées*, Journées d'Informatique Musicale (JIM 2018), Amiens, France, éd. L. Bigo, M. Giraud, R. Groult, F. Levé, pp. 87–95, 2018.

nombre de divisions de la noire de 120, dans une mesure à 4/4, une durée de 120 correspond à une ♩, 60 à une ♪, 30 à une 𝅘𝅥𝅯, 40 à une ♪ de triolet, *etc.*

Par ailleurs, le *type* de chaque note est également spécifié dans un élément `<type>`, qui prendra par exemple la valeur `quarter` pour ♩, `eighth` pour ♪, `16th` pour 𝅘𝅥𝅯, *etc.* Cette valeur peut être déduite de la durée spécifiée comme ci-dessus (en prenant en compte le nombre de divisions de la noire), mais elle est aussi écrite explicitement dans le document, pour décrire la figure de note qui sera imprimée dans la partition. Ainsi, pour l'exemple du paragraphe précédent, le même type de note ♪ correspond aux deux durées 60 et 40.

La correspondance entre durée et type de note dépend aussi du contexte de la note, et plus précisément du groupe dans lequel elle figure. Par exemple, les trois premières noires de la première mesure de la figure 1 ont une durée de $\frac{2}{3}$ de noire, car elles forment un triolet de noires.

Dans la suite de cet article, nous utiliserons le terme *tuple*, issu du jargon informatique, pour désigner les groupes de notes correspondant à une valeur rythmique quelconque (noire, blanche croche), tels que les triolets notés avec le chiffre 3, ou plus généralement d'autres divisions composées d'un ensemble impair de notes [5].

Tous les tuples de notes définissent des durées musicales suivant une logique de division d'un intervalle temporel en parts égales. La notation d'un tuple par $n : m$ (`<time-modification>`, en MusicXML) signifiant une division en n parts dans le même temps que m, précise à la fois la longueur de l'intervalle divisé (m valeurs) et le nombre n de divisions du tuple. Cependant, du point de vue de la gravure, suivant le contexte et la signature rythmique en vigueur, les entiers m et n devront figurer explicitement ou non dans une partition. Ainsi, en figure 1, tous les 3 signifient 3 : 2, le 2 étant omis.

Les tuples de notes peuvent être imbriqués suivant un système hiérarchique de divisions. Les limites d'un tuple sont explicitées par des crochets ou des *ligatures* (*beams*). Les ligatures ne sont pas formellement nécessaires pour la définition des durées de notes en notation musicale (les figures de notes et indications de tuples étant a priori suffisantes), leur usage est essentiel pour faciliter la lecture d'une partition, en particulier la compréhension de la métrique. Par exemple, dans la figure 1, les tuples divisant des durées d'un temps à la fin des mesures 1 et 3 sont ligaturés. En MusicXML, les début et fin de tuples et de ligatures (ainsi que les continuations de ligatures ou les ligatures partielles) doivent être indiqués explicitement à chaque note, par des éléments `<tuplet>` et `<beam>`, dont la valeur (pour le second) ou une valeur d'attribut (pour le premier) indique s'il s'agit d'un début, d'une fin, d'une continuation, *etc.* Cependant, les imbrications de tuples ou ligatures (tels que ceux de la figure 1) ne sont pas représentés par des imbrications d'éléments XML. Ainsi, un document XML valide pourra contenir des éléments marquant les début et fin de tuple/ligature qui ne sont pas bien appariés (en général, le chargement d'un tel document dans un outil comme Finale va échouer). Cela peut induire des incohérences et complique l'implémentation de conventions.

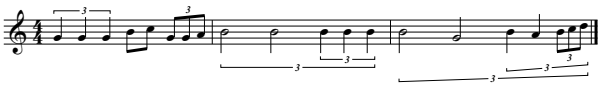

Figure 1 : Tuples imbriqués avec ou sans ligatures. Dans cet article, le terme *ligature* (*beams*) se réfère aux barres utilisées pour regrouper des croches, à ne pas confondre avec les liaisons.

En outre, la relation entre les durées et les types de notes peut aussi être modifiée par l'ajout de *liaisons* (`<tie>`) et de *points*.

Les interdépendances décrites ci-dessus induisent des redondances pour les données présentes dans les encodages de partitions, en particulier en ce qui concerne les aspects rythmiques. Il existe alors des risques d'incohérences dans les encodages, par exemple si les durées (de l'élément `<duration>`) ne correspondent pas aux figures de notes spécifiées (élément `<type>`). D'autres risques d'incohérences structurelles sont liés aux éventuels problèmes d'appariement mentionnés ci-dessus, par exemple avec la spécification d'un tuple ouvert et non fermé.

Une manière d'éviter de telles incohérences pourrait passer par l'expression de *contraintes d'intégrité* (dans un formalisme adéquat) et leur vérification au chargement d'un document. Les schémas XML utilisés actuellement pour les partitions numériques n'incluent pas de telles contraintes. Un document valide d'un point de vue syntaxique (*i.e.* valide par rapport au schéma XML considéré) pourra donc contenir des erreurs, le rendant inutilisable.

Modifier les encodages existants pour ajouter des contraintes d'intégrité ne semble pas une solution réaliste. Premièrement, en raison de la multiplicité des formats considérés comme standards. Deuxièmement, parce qu'en plus des règles strictes d'intégrité (par exemple pour les durées), un certain nombre de règles d'usage, comme la notation des tuples et des ligatures (*cf.* figure 8), relèvent plus de conventions, et les coder de façon définitive dans un schéma serait trop restrictif. De plus, les règles pourront varier suivant le style musical considéré – on peut citer l'exemple des valeurs ajoutées en musique contemporaine.

Pour vérifier qu'une règle ou une convention est satisfaite dans une partition donnée, ou en tenir compte dans des tâches générant une partition, nous choisissons de l'exprimer sous forme de règles simples, portées par un formalisme non ambigu et indépendant des encodages, présenté dans la section suivante.

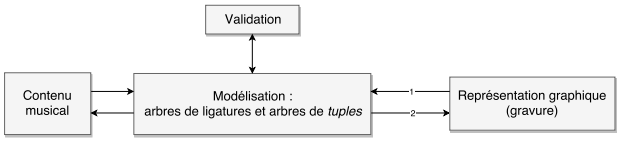

Figure 2 : Le modèle proposé et son intérêt pour la production de partitions. La flèche 1 correspond aux processus décrits dans la section 4 de cet article, la flèche 2 à ceux de la section 5.

3. MODÈLE ABSTRAIT DE GRAVURE

Afin de répondre aux problèmes présentés ci-dessus, nous proposons un modèle abstrait d'éléments de notation rythmique, orienté vers les informations graphiques relatives à la gravure. La figure 2 illustre l'intérêt de notre modélisation pour la production de partitions.

Suivant des travaux antérieurs [9, 1], nous choisissons des représentations de la notation rythmique dans des structures d'arbres. En effet, un tel codage est naturel pour le rythme dans la mesure où il reflète le principe de division hiérarchique des durées en musique [11]. Une autre motivation pour le choix de cette structure de données standard est de permettre d'exprimer simplement des règles de notation à satisfaire et d'en automatiser efficacement la vérification.

Nous avons pris soin d'éviter la redondance d'information dans les modèles proposés. Ainsi, les informations non représentées explicitement, telles que les durées de notes, peuvent être calculées à partir des informations présentes dans le modèle.

Nous avons implémenté des procédures de traduction depuis des encodages XML de partitions vers des arbres et dans la direction inverse, de manière à exploiter ce modèle d'une part dans des tâches de vérification de contraintes d'intégrité appliquées à des partitions XML existantes, et d'autre part pour la génération de partitions XML correctes.

Nous considérons deux types d'arbres dans les traductions, correspondant à deux aspects : les arbres de ligature d'une part, les arbres de tuple d'autre part. Nous les présentons ci-dessous.

3.1. Arbres de ligature

À chaque mesure de chaque voix d'une partition nous associons un arbre représentant à la fois les figures de notes et les ligatures entre notes.

Définition. Dans un arbre, s'il existe une arête $\langle v, v' \rangle$, on dit que le sommet v' est le *successeur* de v. Un sommet qui n'est le successeur d'aucun autre est appelé *racine*. Il n'existe qu'une racine par arbre considéré. Un sommet sans successeur est appelé *feuille*. Les sommets qui ne sont pas des feuilles sont appelés *internes*.

Chaque feuille d'un *arbre de ligature* est étiquetée par un symbole représentant les informations suivantes :

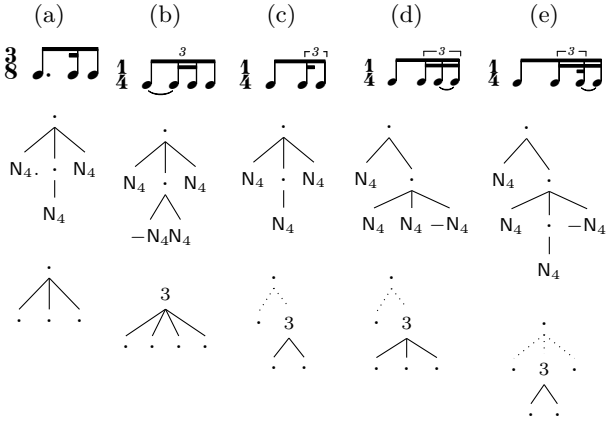

Figure 3 : Différentes gravures d'un même rythme (en haut), les arbres de ligature (milieu) et les arbres de tuple associés (en bas).

- une *figure de tête de note*, parmi 𝅜, 𝅝, 𝅗𝅥, et ♩. Ces figures sont notées respectivement $\mathsf{N}_{\frac{1}{2}}$, N_1, N_2, N_4, l'indice indiquant une durée en fraction de ronde (suivant l'usage dans certains encodages dont MusicXML). On pourrait ajouter des symboles plus longs que 𝅜 avec $\mathsf{N}_{\frac{1}{4}}$, *etc.* En revanche, les durées plus courtes que ♩ (représentées en notation musicale par des crochets – ou ligatures, et non des têtes de notes particulières) sont codées dans la structure de l'arbre comme nous le verrons plus bas.
- optionnellement, un ou plusieurs *points*, chaque point augmentant la durée de la moitié de sa valeur (pour le premier point) ou de la valeur du point précédent (pour les suivants).
- optionnellement, une *liaison* (*tie*), notée par '-' au début du symbole, indiquant que la durée du symbole s'ajoute à celle du symbole précédent dans l'arbre (suivant un parcours des feuilles en profondeur d'abord).

On utilise également des symboles de silences construits autour de $\mathsf{R}_{\frac{1}{2}}$, R_1, R_2, R_4 suivant le même principe. On trouve différents exemples de symboles de feuilles dans les arbres de ligature de la figure 3.

Les nœuds internes des arbres de ligature sont sans étiquette. Nous les représentons avec un · dans la figure 3.

Les arêtes d'un arbre de ligature peuvent être de deux types. Les *arêtes pleines* se propagent vers les feuilles : si $\langle v, v' \rangle$ est une arête pleine, toute arête sous v' est pleine. Chaque arête pleine représente un crochet dans le cas de notes isolées, ou une ligature dans le cas de notes regroupées. Plus précisément, pour une feuille étiquetée par N_4 (figure de note ♩), le nombre de crochets (ou ligatures) est le nombre d'arêtes pleines sur le chemin entre cette feuille et la racine. Donc une feuille N_4 avec deux arêtes pleines jusqu'à la racine représente une durée de 𝅘𝅥𝅯. C'est

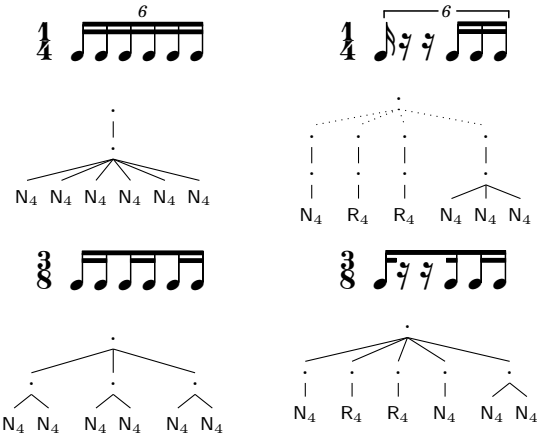

Figure 4 : Arbres de ligature.

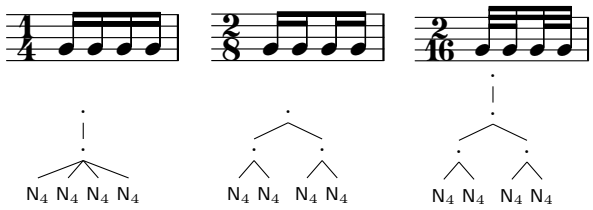

Figure 5 : Arbres de ligature (examples de [7]).

le cas par exemple de la première note, isolée, dans l'exemple en haut à droite de la figure 4 et des trois dernières notes de cette mesure (qui sont groupées par une ligature).

Le principe est le même pour les silences, comme on peut le voir dans la même mesure de la figure 4, et la mesure en bas à droite de la même figure.

Les arêtes du second type sont représentées en pointillés, et n'ont pas la signification des premières. Elle sont simplement utilisées pour ajouter des événements qui ne sont pas liés par des ligatures (c'est le cas en particulier des silences), dans un arbre où les événements sont lus dans un parcours en profondeur d'abord.

Les arêtes des arbres de ligature apportent des informations quand à la gravure des figures de rythmes, ainsi que décrit formellement dans le paragraphe ci-dessous. En revanche, les durées des notes représentées ne peuvent être calculées à partir des arbres de ligature seuls. L'utilisation conjointe des arbres de ligature et des arbres de tuple décrits au paragraphe 3.2 est nécessaire pour déduire les informations sémantiques de durées.

Propriétés. Appelons *profondeur pleine* d'un sommet d'un arbre de ligature le nombre d'arêtes pleines sur le chemin entre ce sommet et la racine.

Nous avons vu que dans le cas d'une feuille, la profondeur pleine indique le nombre de crochets ou de ligatures pour l'élément correspondant.

Le plus petit ancêtre commun de deux sommets n_1 et n_2 d'un arbre de ligature est noté $lca(n_1, n_2)$, et

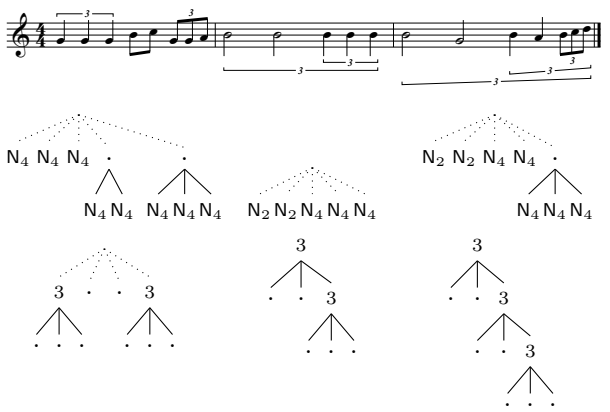

Figure 6 : Arbres de ligature et arbres de tuples pour les mesures de la figure 1.

$plca(n_1, n_2)$ est le plus petit ancêtre commun de n_1 et n_2 en ne passant que par des arêtes pleines. Plus formellement, $plca(n_1, n_2) = lca(n_1, n_2)$ si le chemin entre n_1 et $lca(n_1, n_2)$ ainsi que le chemin entre n_2 et $lca(n_1, n_2)$ ne contiennent que des arêtes pleines, et $plca(n_1, n_2)$ n'est pas défini sinon.

La propriété suivante des arbres de ligature est utilisée dans les applications présentées ci-dessous :
Le *nombre de ligatures* entre deux feuilles ℓ_1, ℓ_2 est la profondeur pleine de $plca(\ell_1, \ell_2)$ plus 1, si $plca(\ell_1, \ell_2)$ est défini, et 0 sinon.

On peut par exemple vérifier cette propriété sur la figure 5. Le nombre de ligatures entre chaque paire de notes successives de la première mesure est 2 (car la profondeur pleine de leur *plca* est 1). Dans le cas de la deuxième (resp. troisième) mesure de la figure 5, le nombre de ligatures entre la première et deuxième note, ainsi qu'entre la troisième et quatrième note est 2 (resp. 3), mais le nombre de ligatures entre la deuxième et troisième note n'est que de 1 (resp. 2). Nous verrons en section 4.2 que cela correspond à des conventions de notation des ligatures.

3.2. Arbres de tuple

Les *arbres de tuple* représentent les éléments de notation des tuples, plus précisément les emplacements de chiffres et crochets définissant explicitement dans une partition la nature des tuples. Les figures 3 et 6 présentent des exemples d'arbres de tuple.

Définition. Chaque nœud interne d'un arbre de tuple est étiquetés ou bien par une paire d'entiers $n : m$, $n, m \geq 2$, signifiant n éléments à la place de m, ou bien par un seul entier n si m peut être omis, ou bien pas étiqueté (représentés avec un $\cdot$ dans les figures). Les feuilles de ces arbres sont sans étiquette.

Propiétés. La sémantique d'un arbre de tuple est la suivante. Un arbre de tuple et son arbre de ligature associé ont toujours le même nombre de feuilles, correspondants au nombre de figures de notes représentées. Lorsqu'un nœud interne est étiqueté par n

ou $n : m$, la suite des éléments correspondant aux feuilles sous le nœud interne se voit affectée (dans la notation rythmique) la même étiquette (n ou $n : m$). L'étiquette est notée ou bien au-dessus de la ligature définie par l'arbre de ligature, ou bien au dessus d'un crochet ajouté dans le cas où l'arbre de ligature ne définit pas de ligature entre les éléments concernés.

Par exemple, dans le cas de la première mesure figure 6, la suite des trois premières notes est étiquetée par 3 (avec un crochet), la suite des trois dernières notes est étiquetée aussi par 3 (sans crochet car ces trois notes sont ligaturées comme défini par l'arbre de ligature), et les deux notes du milieu n'ont pas d'étiquette de tuple.

3.3. Calcul des durées et complémentarité

Notons que, pas plus qu'un arbre de ligature, un arbre de tuple seul ne permet pas de déduire les informations sémantiques sur les durées de notes. L'utilisation conjointe d'un arbre de ligature et d'un arbre de tuple est nécessaire pour cela (cf. paragraph 4.2). En se sens, les deux types d'arbres sont complémentaires.

L'introduction de ces deux types d'arbres complémentaires s'est avéré nécessaire car il existe des cas où il n'est pas possible de coder simplement dans un même arbre les ligatures et les informations sur les notations de tuples. Considérons par exemple la mesure (c) de la figure 3. Les deux dernières feuilles correspondent à des éléments appartenant au même triplet (triolet de doubles). Mais dans l'arbre de ligature, les deux dernières feuilles ne sont pas à des positions frères. Par conséquent il n'est pas possible d'ajouter l'étiquette 3 de triolet à un nœud interne de l'arbre de ligature : le nœud parent de l'avant-dernière feuille ne concernerait que cette feuille, et la racine concernerait l'ensemble de la figure (un 3 à la racine correspondrait à un triolet de croches). Le rôle de l'arbre de tuple est ici donc d'offrir un nœud interne parent de ces deux dernière feuilles pour introduire la notation 3.

Les durées de notes peuvent être calculées à partir des étiquettes des arbres de ligature et tuple, et de leur structure : nous avons vu que ces deux types d'arbres fournissent les informations nécessaires pour la gravure de rythmes (figures de notes, points, liaisons, ligatures et tuplets). Les durées peuvent donc être déterminées à partir des rythmes gravés suivant les règles du solfège.

Plus précisément, l'arbre de ligature permet de déduire pour chaque feuille une durée *temporaire*, suivant l'étiquette de cette feuille et sa profondeur pleine dans l'arbre (qui indique rappelons-nous, le nombre le nombre de crochets ou de ligatures). Par exemple, pour le cas (b) de la figure 3, la durée temporaire de la première note est $\frac{1}{8}$ (de ronde), déduit de l'étiquette N_4 (correspondant à $\frac{1}{4}$ de ronde) et la profondeur pleine 1 (correspondant à une ligature, donc une division par 2 de $\frac{1}{4}$). C'est aussi le cas de la dernière note. La durée temporaires des deuxième et troisième feuilles est $\frac{1}{16}$. Puis cette durée temporaire est ajustée suivant les informations contenues dans l'arbre de tuple : si une feuille est sous un nœud étiquetté $n : m$ (avec $m = 2$ dans le cas où le second entier est manquant), on applique le coefficient $\frac{m}{n}$. Dans l'exemple ci-dessus, on obtient une durée de $\frac{1}{12} = \frac{1}{8} \times \frac{2}{3}$ pour la note correspondant à la première et la dernière feuille, et $\frac{1}{24}$ pour les deuxième et troisième notes.

Dans le cas (c) de la même figure, la durée de la première note est $\frac{1}{2}$ (ici le coefficient d'ajustement est 1), la durée de la seconde note est $\frac{1}{24} = \frac{1}{16} \times \frac{2}{3}$, et la durée de la dernière note est $\frac{1}{12} = \frac{1}{8} \times \frac{2}{3}$.

3.4. Autres représentations

L'environnement d'assistance à la composition OpenMusic [3] utilise une structure appelée *arbres de rythme* (OMRT) pour la représentation des données musicales temporelles symboliques. Les OMRT suivent le principe de la définition de durées symboliques par divisions récursives d'intervalles temporels, et sont donc proches conceptuellement de la notation rythmique traditionnelle. Cela les rend très utiles dans un contexte d'aide à la création et de manipulation algorithmique de structure rythmiques complexes. Leur usage dans un contexte de gravure des rythmes a aussi été étudié dans [1].

Les arbres de ligature et de tuple présentés ici diffèrent des OMRT dans la mesure où ils mettent l'accent sur la représentation graphique de la notation, et non sa définition logique (i.e. sa sémantique).

En section 5, nous suggérons des applications où les arbres de ligature et tuple sont utilisés comme un intermédiaire entre les OMRT (ou des arbres de rythmes similaires) et la notation finale dans un contexte d'aide à la gravure.

4. APPLICATIONS À L'ÉVALUATION DE PARTITIONS NUMÉRISÉES

Nous présentons dans cette section la validation de partitions numérisées à l'aide de notre modélisation sous forme d'arbres, ce qui correspond à la flèche 1 de la figure 2.

4.1. Importer des fichiers XML

Nous avons écrit des procédures d'extraction d'arbres de ligature et arbres de tuple à partir de fichiers MusicXML et MEI.

La construction des arbres se fait séparément (en deux passes) à partir des informations dans le fichier XML sur les structures de ligatures et de tuples, *e.g.* dans le cas de MusicXML à partir des éléments respectivement de la forme `<beam>` et `<tuple>`.

Nous passons pour cela par la boite à outils MUSIC21 [4], qui assure la lecture et l'analyse (*parsing*)

du fichier XML et la conversion sous forme de séquences d'événements. Ces événements contiennent en particulier des informations sur les début et fin de chaque ligature, tuple, *etc.* Des fonctions Python ont été écrites pour parcourir ces séquences d'événements de gauche-à-droite et construire les arbres de ligature et de tuple à l'aide d'une pile.

4.2. Évaluation de la qualité des données

Dans le cadre du projet GioQoSo portant sur l'étude de l'évaluation de la qualité des données dans les partitions numériques, nous avons proposé un outil qui réunit dans une interface unifiée différentes procédures d'évaluation qualitative de partitions musicales [1] , voir l'illustration sur la figure 7.

Ces évaluations s'appliquent à toute partition dans un format XML (MusicXML ou MEI) accessible par une URL. Nous donnons ci-dessous quelques exemples de propriétés sur les données relatives à la notation rythmique, vérifiées grâce à la construction d'arbres de la section 3.

Vérification des durées. Le calcul présentée au paragraph 3.3 est utile pour la vérification de l'intégrité des données contenues dans une partition numérique, plus précisément pour vérifier que chaque durée symbolique enregistrée (en *quarter-length*, XML `<duration>`, cf. paragraphsec :elements) correspond bien à la durée définie par les types de notes, les nombres de crochets ou ligatures, voire les tuples dans lesquelles ces notes sont incluses (cf. paragraph 3.3).

Une incohérence entre durée enregistrée et sa représentation graphique peut être considérée comme une erreur critique sur une partition, particulièrement si cette valeur de durée est utilisée dans un traitement de la partition.

Une autre procédure de GioQoSo vérifie que chaque mesure a la durée totale attendue, en faisant la somme des valeurs de durées des éléments contenus dans la mesure. La cohérence entre valeurs et représentations des durées est donc importante dans ce cadre.

Notation de tuple. En fonction du contexte (signature rythmique), l'étiquetage des tuples (par un entier ou une paire d'entier) doit être écrit explicitement ou pas, suivant des règles précises. Il peut être vérifié directement sur les arbres de tuple, par des tests sur les étiquettes, (cf. description au paragraphe 3.2), que les notations de tuples écrites devaient bien être explicites, et que ces notations sont correctes.

Conventions sur les ligatures. Certaines règles de bonne pratique pour l'écriture des ligatures, moins critiques que les problèmes d'intégrité présentés plus haut, sont néanmoins d'une grande importance en ce qui concerne la lisibilité des partitions. Les groupements de notes définis par les ligatures permettent en effet de donner au lecteur des informations visuelles immédiates sur les appuis métriques. Ces groupements doivent donc être cohérents avec la métrique.

Par exemple, il est convenu (voir [7] page 155) que certaines positions de la mesure (temps forts) ne doivent pas être croisées par une ligature car elles correspondent à des temps forts de la mesure (comme le troisième temps d'une mesure à 4/4).

Une telle propriété peut être vérifiée à l'aide de l'estimation des durées (paragraphe 3.3) et de la notion de nombre de ligatures entre notes, que nous avons présentée au paragraphe 3.1. Dans ce cas, le nombre de ligatures entre les deux feuilles situées de part et d'autre d'un temps fort à ne pas croiser devra être 0.

Il est également admis que les grands groupes de notes courtes (sous la double-croche, donc notés avec de multiples ligatures) sont plus faciles à lire lorsqu'ils sont divisés en sous-groupes dont la durée dépend de la métrique. D'après [7], le principe à suivre est que le nombre de ligatures séparant deux sous-groupes doit être égal à la durée des groupes qu'elles séparent. Cette recommandation est illustrée en figure 8 et peut aussi être vérifiée grâce à la propriété des arbres de ligature mentionnée au paragraphe 3.1.

Un problème détecté concernant les règles ci-dessus peut être signalé comme recommandation, il n'est pas du même niveau de criticité que les éventuelles incohérences sur les durées vues plus haut. Les fonctionnalités présentées plus haut sont été intégrées dans la bibliothèque Neuma, dans le cadre du projet GioQoSo.

4.3. Comparaison de partitions

La question d'une opération de recherche de *différences* dans des partitions numériques a été étudiée dans le cadre de l'édition collaborative en ligne de partitions [2]. Constatant d'une part que la comparaison manuelle est particulièrement fastidieuse, et que lancer la commande Unix diff sur les fichiers (texte) XML contenant les partitions ne donne pas de résultat exploitable, [2] propose d'appliquer sur des représentations hiérarchiques de partitions des algorithmes de calcul de distance d'édition sur les arbres [15, 12].

La représentation de la notation rythmique proposée ici pourrait être utilisée pour détecter des différences visuelles [2] entre partitions à un fin niveau de granularité, par simple comparaison (isomorphisme) d'arbres. Une représentation fidèle dans les arbres d'éléments graphiques de notation rythmique est avantageuse dans ce cadre, car elle assure une correspondance entre le résultat de la comparaison d'arbres et d'une comparaison visuelle de partitions. En d'autres termes, cette représentation des données rythmiques pourrait aider à l'automatisation de la tâche

1 . `http://neuma.huma-num.fr/quality`

2 . Notons que des notations sémantiquement équivalentes en terme de durées peuvent être différentes visuellement, cf. figure 3.

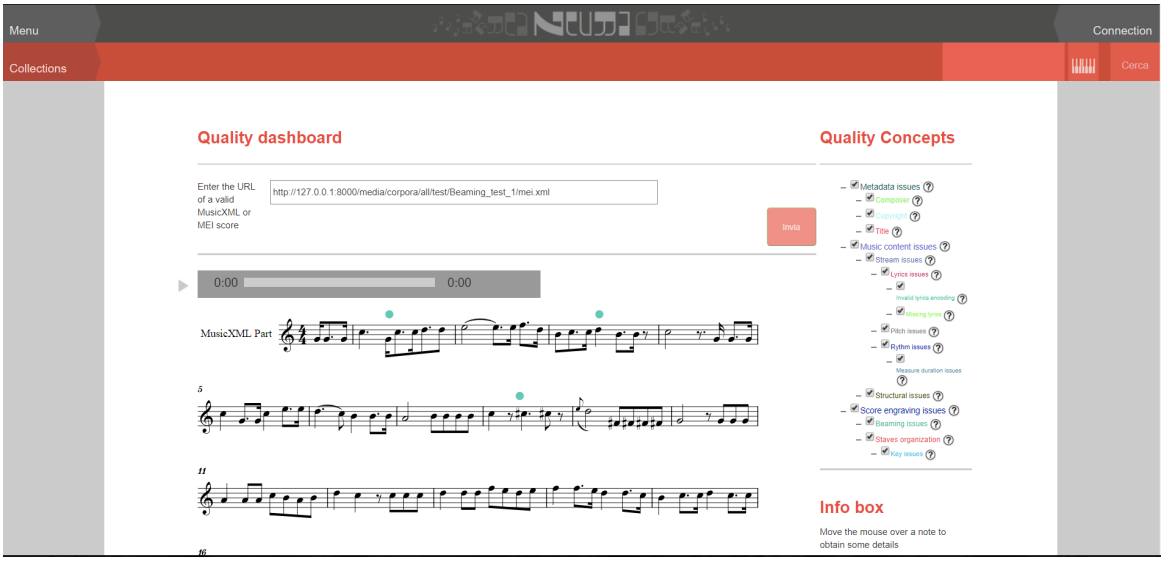

Figure 7 : Une capture d'écran de l'interface de GioQoSo, intégré à la plateforme Neuma. Des problèmes de notation rythmique sont indiqués par des points verts.

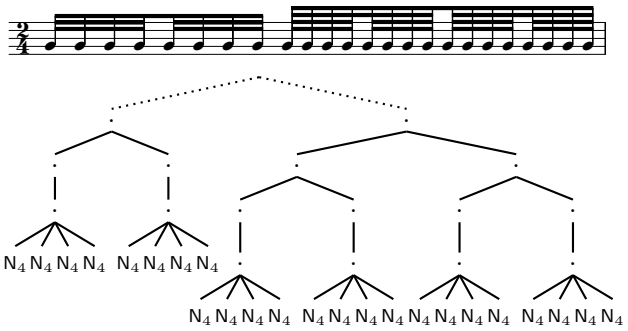

Figure 8 : Divisions dans les ligatures et arbres de ligatures correspondant. D'après [7] : *"le nombre de ligatures séparant deux sous-groupes doit être proportionnel à la durée des groupes qu'elles séparent"*.

laborieuse de comparaison de différentes versions de partitions.

En prévision d'un tel usage, et afin de proposer un retour précis à l'utilisateur sur les différences observées, les identifiants des éléments XML sont stockés dans les sommets des arbres de ligature et tuples (ce qui est possible avec MEI), dans la procédure d'extraction décrite dans la section 4.1.

5. APPLICATIONS EN MODÈLE GÉNÉRATIF (PERSPECTIVES)

Il est possible par un parcours d'arbres de ligature et de tuple d'extraire les éléments XML (`<beam>` et `<tuple>`) produisant les notations correspondantes. Nous discutons informellement dans cette section de plusieurs usages possibles (actuellement en cours de développement) des modèles présentés ci-dessus en modèle génératif, dans le cadre d'applications produisant des partitions numériques, ce qui correspond à la flèche 2 de la figure 2.

L'avantage de l'utilisation de représentations intermédiaires sous forme d'arbres comme ci-dessus est de pouvoir s'appuyer sur des procédures bien établies, par exemple de *parsing* ou de *transformations* d'arbres, qui s'intègrent aisément dans un processus complet de production de partitions.

Procédure générique. Plus précisément, considérons le scénario suivant dans lequel les arbres de ligature et de tuples apparaissent comme représentation intermédiaire.

On suppose qu'à un certain stade de production d'une partition, nous disposons d'une suite d'événements musicaux dont les durées sont exprimées en fraction de noire (*quarter-length*). Cette séquence de durées peut être prise en entrée par une procédure de *parsing* suivant une *grammaire formelle* décrivant différents choix possibles de subdivisions successives des durées. Une telle procédure est présentée dans [8]. Les arbres de syntaxes produits sont appelés *arbres de divisions* car ils décrivent les divisions choisies. Ils sont très similaires aux arbres de rythmes OpenMusic [1].

La figure 9 présente trois arbres de divisions possibles pour la suite de durées initiale $\frac{1}{2}\ \frac{1}{6}\ \frac{1}{3}$. Dans le premier des 3 arbres, on a une division initiale du temps en 3 parties égales. Le premier fils, feuille étiquetée par $\bullet$, correspond à une note de durée $\frac{1}{3}$. Le deuxième fils est à nouveau divisé en deux parties égales de durée $\frac{1}{6}$. Le premier de ses deux petits-fils est étiqueté par $-$, ce qui signifie qu'il s'agit d'une continuation de l'événement précédent (ce qui peut être représenté par une liaison ou aussi ici un point). Le premier événement représenté aura donc une durée de $\frac{1}{3} + \frac{1}{6} = \frac{1}{2}$. Le second petit-fils est étiqueté par $\bullet$ et correspond à une note de durée $\frac{1}{6}$ (deuxième événement en entrée). Enfin, le troisième fils est éti-

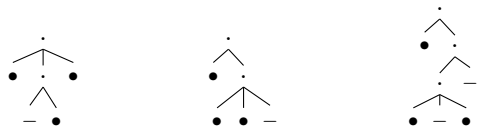

Figure 9 : Trois arbres de divisions possibles pour la suite de durées $\frac{1}{2}\ \frac{1}{6}\ \frac{1}{3}$.

queté par • et correspond donc à une note de durée $\frac{1}{3}$ (troisième événement en entrée).

Le premier arbre de division de la figure 9 correspond aux arbres de ligature et de tuple des cas (a) et (b) de la figure 3, à partir desquels sera généré le code XML donnant les notations (correspondantes). Le deuxième arbre de division de la figure 9 correspond aux cas (c) et (d) de la figure 3, et le troisième arbre au cas (e).

Gravure. Des langages textuels comme Lilypond ou Guido permettent d'écrire de manière symbolique des suites d'événements musicaux avec des durées. La procédure décrite sommairement ci-dessus pourrait donc être utile dans le cadre d'une conversion de code dans ces formats textuels vers des partitions XML.

Des informations sur les tuples peuvent être contenues dans le code Lilypond ou Guido, qui devront être prise en compte pour la création des arbres de division. Une approche pour traiter ces contraintes est de les utiliser pour modifier la grammaire formelle utilisée dans le parsing.

Transcription. La procédure générique ci-dessus, qui utilise les arbres de ligature et tuples, pourra aussi être utilisée comme sous-tâche en aval d'une procédure de transcription par étapes.

Supposons que les premières étapes de transcription permettent d'obtenir des suites d'événements aux durées arbitraires (*e.g.* provenant d'un enregistrement MIDI de performance, ou de descripteurs extraits d'un enregistrement audio). Une étape de quantification rythmique retournera une suite d'événements aux durées quantifiées (en *quarter-length* comme ci-dessus) ou bien directement des structures semblables aux arbres de divisions ci-dessus dans le cas d'une procédure comme [14]. Nous pouvons alors procéder comme ci-dessus pour la production d'une partitions XML à partir de ces données.

6. CONCLUSION

Nous avons proposé et développé un modèle abstrait capable de représenter l'information relative au rythme d'une partition musicale. Il repose sur une structure de données unique, les arbres (au sens de la théorie des graphes), utilisée pour produire des *arbres de ligature* et des *arbres de tuple*, à partir d'encodages XML de partitions. Ce modèle a été mis à profit pour valider ces encodages, au cœur de la plateforme Neuma. Nous avons également discuté les possibilités (en cours de développement) de génération de notation à partir d'arbres existants, pour des tâches liées au rendu pour langages musicaux textuels ou la transcription.

Cette première avancée ouvre des perspectives intéressantes de travail et de collaboration avec la communauté musicologique. Tout d'abord, il serait sans doute utile de proposer une visualisation des arbres créés, dans Neuma mais aussi peut-être par l'intermédiaire d'un outil externe. Cela favoriserait l'adoption de ce modèle, pour les analystes comme pour les développeurs d'encodages.

Des extensions théoriques de ce travail de modélisation sont envisageables, pour valider par exemple d'autres aspects d'une partition que le seul rythme (accords, texte...). Le modèle pourrait également être étendu pour représenter l'ensemble d'une partition, avec des conteneurs pour voix, portées et systèmes.

Enfin, nous poursuivons l'objectif d'élaborer un outil de comparaison de partitions, dans lequel s'inscrit l'effort de modélisation présenté ici. Cela nécessitera d'une part l'élaboration d'algorithmes de comparaisons d'arbres appropriés et d'autre part le développement d'un retour visuel sur les partitions.

7. RÉFÉRENCES

[1] C. Agon, K. Haddad, and G. Assayag. Representation and rendering of rhythm structures. In *Proceedings Second International Conference on WEB Delivering of Music (CW'02)*, pages 109–113, 2002. IEEE Computer Society.

[2] C. Antila, J. Treviño, and G. Weaver. A hierarchic diff algorithm for collaborative music document editing. In *Third International Conference on Technologies for Music Notation and Representation (TENOR)*, 2017.

[3] J. Bresson, C. Agon, and G. Assayag. Openmusic : visual programming environment for music composition, analysis and research. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 743–746. ACM, 2011.

[4] M. S. Cuthbert and C. Ariza. Music21 : A Toolkit for Computer-Aided Musicology and Symbolic Music Data. pages 637–642, 2010.

[5] A. L. Danhauser. *Théorie de la musique*. H. Lemoine, 1872.

[6] M. Good. *MusicXML for Notation and Analysis*, pages 113–124. W. B. Hewlett and E. Selfridge-Field, MIT Press, 2001.

[7] E. Gould. *Behind Bars*. Faber Music, 2011.

[8] F. Jacquemard, A. Ycart, and M. Sakai. Generating equivalent rhythmic notations based on rhythm tree languages. In *TENOR* 2017.

[9] M. Laurson. Patchwork : A visual programming language and some musical applications. Technical report, Sibelius Academy, Helsinki, 1996.

[10] Music Encoding Initiative. `http://music-encoding.org`, 2015.

[11] P. Nauert. A theory of complexity to constrain the approximation of arbitrary sequences of timepoints. *Perspectives of New Music*, 32(2) :226–263, 1994.

[12] M. Pawlik and N. Augsten. Tree edit distance : Robust and memory-efficient. *Information Systems*, 56 :157–173, 2016.

[13] P. Rolland. The Music Encoding Initiative (MEI). In *Proc. Intl. Conf. on Musical Applications Using XML*, pages 55–59, 2002.

[14] A. Ycart, F. Jacquemard, J. Bresson, and S. Staworko. A Supervised Approach for Rhythm Transcription Based on Tree Series Enumeration. In *Proceedings of the 42nd International Computer Music Conference (ICMC)*, 2016.

[15] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6) :1245–1262, 1989.