Composition de partitions musicales

Dominique Fober, Yann Orlarey, Stéphane Letz

COMPOSITION DE PARTITIONS MUSICALES

D. Fober, Y. Orlarey, S. Letz
Grame - Centre national de création musicale
{fober, orlarey, letz}@grame.fr

RÉSUMÉ

Basé sur le format de notation musicale GUIDO, nous avons développé un ensemble d'outils pour la composition de partitions musicales. Il s'agit d'opérations de transformation de haut niveau, prenant des partitions en entrée pour en générer de nouvelles en sortie. Ces opérations peuvent par exemple s'appliquer au domaine temporel (e.g. couper le début ou la fin d'une partition) ou concerner la structure de la partition (mise en séquence, en parallèle). La définition d'opérateurs applicables au niveau de la notation permet d'englober l'expression d'idées musicales et leur composition dans une même métaphore. Cela soulève toutefois un certain nombres de problèmes liés à la cohérence de la notation. Cet article donne un aperçu du format de notation musicale GUIDO, puis il présente les opérations de composition de partitions, les problèmes qui se posent pour la cohérence de la notation et les solutions proposées.

1. INTRODUCTION

Le format de notation musicale GUIDO [GMN] [1] [2] a été défini par H. Hoos et K. Hamel il y a plus de 10 ans. Il est très proche du format adopté par Lilypond [3] [4] mais il est apparu antérieurement. Le format GMN est un langage textuel de représentation de partitions musicales. Il est basé sur un formalisme simple mais puissant, se concentrant sur les grands concepts musicaux (en opposition aux caractéristiques graphiques). L'approche développée par GUIDO repose sur l'idée *d'adéquation*, qui invite à noter simplement les concepts musicaux simples, et à ne recourir à des notations complexes que pour les notions musicales complexes.

Basée sur le format GMN, la librairie GUIDO [5, 6] fournit un puissant moteur de mise en page de partitions, qui se différencie notamment des approches de type *compilateur* [3, 7] par sa capacité à être embarquée dans une application indépendante, et par la rapidité et l'efficacité du moteur de rendu, qui le rendent utilisable dans un contexte *temps réel* pour des partitions simples.

Basées sur la combinaison du langage et du moteur de rendu GUIDO, des opérations de composition de partitions ont été développées : opérations de transformation rythmique, de hauteur, de sélection temporelle ou événementielle, de mise en séquence ou en parallèle, etc.

La notation musicale reste l'une des représentations les plus utilisées par les musiciens. La définition d'opérateurs de composition de partitions constitue un moyen homogène de concilier écriture et transformation tout en restant à un haut niveau de description symbolique de la musique. De plus, la conception de ces opérateurs permet d'utiliser des partitions tant comme cible que comme argument de ces opérations, renforçant la métaphore de la notation comme support d'idées musicales aussi bien que d'opérations de composition.

Toutefois, ces opérations de composition de partitions musicales soulèvent un certain nombre de problèmes liés à la cohérence de la notation. Comme élément de réponse à ces problèmes, nous proposons une typologie simple des éléments de notation musicale ainsi qu'un ensemble de règles de composition basées sur cette typologie.

Cet article donnera quelques rappels sur le format de notation GUIDO, les sections suivantes présenteront les opérations de composition de partitions, les problèmes soulevés par ces opérations et les solutions proposées. Nous proposerons ensuite une extension du langage GMN pour prendre en compte la réversibilité des opérations.

2. LE FORMAT DE NOTATION GUIDO

2.1. Concepts de base

Le format de base de GUIDO recouvre les notes, silences, altérations, voix indépendantes, et les concepts les plus courants de la notation musicale comme les clefs, métriques, armures, articulations, liaisons, etc. Les notes sont représentées par leur nom `(a b c d e f g h)`, une altération optionnelle ('#' et '&' pour dièse et bémol), un numéro d'octave optionnel et une durée optionnelle.
La durée est spécifiée par l'une des formes suivantes :

```
'*'enum'/'denom dotting
'*'enum dotting
'/'denom dotting
```

où $enum$ et $denom$ sont des entiers positifs et $dotting$ est soit vide, '.', ou '..', avec la même sémantique que dans la notation musicale. 1 est la valeur par défaut si $enum$ ou $denom$ sont omis. La durée exprime une fraction de ronde.

Si omises, les valeurs optionnelles sont les dernières utilisée pour la note précédente de la séquence.

Les accords sont décrits par des notes entre accolades séparées par des virgules, i.e. `{c, e, g}`

2.2. Tags GUIDO

Les tags sont utilisés pour donner des informations musicales supplémentaires, comme les liaisons, clés, ar-

mures, etc. Un tag a l'une des formes suivantes :

```
\tagname
\tagname<param-list>
```

où `param-list` est une liste de chaînes de caractères ou de nombres, séparés par des virgules. Un tag peut avoir une étendue temporelle et s'appliquer à un ensemble de notes (par ex. les liaisons) ; la forme correspondante est :

```
\tagname(note-series)
\tagname<param-list>(note-series)
```

Le code GMN suivant illustre la concision de la notation ; la figure 1 donne le rendu effectué par le moteur GUIDO.

```
[ \meter<"4/4"> \key<-2> c d e& f/8 g ]
```

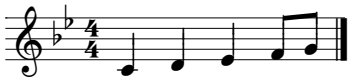

Figure 1. Un exemple GMN simple

2.3. Séquences et segments

Une séquence de notes a la forme `[tagged-notes]` où `tagged-notes` est une série de notes, tags, tags temporels séparés par des espaces. Une séquence de notes représente une seule voix. Les segments de notes représentent plusieurs voix ; ils sont notés par `{seq-list}` où `seq-list` est une liste de séquences de notes séparées par des virgules, comme dans l'exemple suivant (figure 2) :

```
{ [ e g f ], [ a e a ] }
```

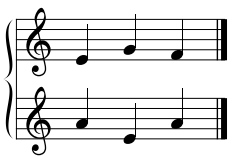

Figure 2. Exemple à plusieurs voix

2.4. GUIDO Avancé

Le format GUIDO Avancé inclus des tags supplémentaires et un contrôle plus fin de la mise en page. Il définit notamment des paramètres tels que dx et dy pour ajuster le positionnement des éléments de la partition. Il autorise le formatage de notes et de silences, les assignations à des portées, etc. Un exemple de code est donné ci-dessous avec le rendu correspondant (figure 3).

```
{
 [
  \barFormat<"system">
  \staff<1> \stemsUp \meter<"2/4">
  \intens<"p", dx=1hs,dy=-7hs>
  \beam(g2/32 e/16 c*3/32) c/8
  \beam(\noteFormat<dx=-0.9hs>(a1/16) c2 f)
  \beam(g/32 d/16 h1*3/32) d2/8
  \beam(h1/16 d2 g)],
 [\staff<1>\stemsDown g1/8 e
  f/16 \noteFormat<dx=0.8hs>(g) f a a/8 e
  f/16 g f e],
 [\staff<2> \meter<"2/4">
  \stemsUp a0 f h c1],
 [\staff<2> \stemsDown c0 d g {d, a}]
}
```

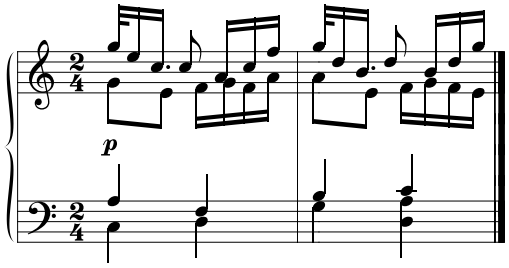

Figure 3. Exemple GUIDO Avancé

3. COMPOSITION DE PARTITIONS

3.1. Operations

Des opérations de composition de partitions ont été implémentées sous forme de librairie indépendante (GuidoAR). Elles sont décrites par la table 1. Ces opérations sont disponibles sous forme d'API, d'utilitaires pour la ligne de commande, ou encore dans l'environnement graphique d'une application indépendants : GUIDOCalculus. La plupart de ces opérations prennent une partition et une valeur en entrée pour produire une nouvelle partition en sortie. La valeur passée en entrée peut-être donnée sous forme de partition : par exemple, l'opération `top` coupe les voix d'une partition après un nombre de voix n ; en utilisant une partition à la place de ce paramètre, c'est le nombre de voix de cette partition qui sera pris comme valeur. Ainsi, toutes les opérations peuvent se décrire de manière homogène au niveau symbolique de la notation.

Ce design permet également d'exprimer des séries de transformations en ligne de commande, comme le pipeline de partitions à travers de opérateurs e.g.

```
head s1 s2 | par s2 | transpose "[ c ]"
```

3.2. Les problèmes de notation

Dans les faits, les fonctions de composition de partition opèrent sur une représentation en mémoire de la notation musicale. Toutefois, nous allons illustrer les problèmes liés avec la représentation textuelle qui est équivalente à la représentation mémoire.

Prenons l'exemple de l'opération `tail` appliquée à la partition suivante :

```
[\clef<"f"> c d e c]
```

Une coupure *brute* de la partition après 2 notes donne le code `[e c]` où l'information de clef a disparu, pouvant conduire à un rendu non souhaité (figure 4).

Voici un autre exemple avec l'opération `seq` : la mise en séquence de `[\clef<"g"> c d]`

operation	args	description
seq	$s1$ $s2$	met $s1$ et $s2$ en séquence
par	$s1$ $s2$	met $s1$ et $s2$ en parallèle
rpar	$s1$ $s2$	met $s1$ et $s2$ en parallèle, alignés à droite
top	$s1$ $[n \mid s2]$	prend les n premières voix de $s1$; quand on utilise une partition $s2$ comme paramètre, n est le nombre de voix de $s2$
bottom	$s1$ $[n \mid s2]$	coupe les n première voix de $s1$; quand on utilise une partition $s2$ comme paramètre, n est le nombre de voix de $s2$
head	$s1$ $[d \mid s2]$	prend le début de $s1$ jusqu'à la date d ; quand on utilise une partition $s2$ comme paramètre, d est la durée de $s2$
evhead	$s1$ $[n \mid s2]$	prend les n premiers événements $s1$; quand on utilise une partition $s2$ comme paramètre, n est le nombre d'événements de $s2$
tail	$s1$ $[d \mid s2]$	coupe le début de $s1$ jusqu'à la date d ; quand on utilise une partition $s2$ comme paramètre, d est la durée de $s2$
evtail	$s1$ $[n \mid s2]$	coupe les n premiers événements de $s1$; quand on utilise une partition $s2$ comme paramètre, n est le nombre d'événements de $s2$
transpose	$s1$ $[i \mid s2]$	transpose $s1$ d'un intervale i ; quand on utilise une partition $s2$ comme paramètre, i est calculé comme la différence entre la première note de la première voix de $s1$ et $s2$
duration	$s1$ $[d \mid r \mid s2]$	étire $s1$ à une durée d ou selon un facteur r ; quand on utilise une partition $s2$ comme paramètre, d est la durée de $s2$
applypitch	$s1$ $s2$	applique les hauteurs de $s1$ à $s2$ en boucle
applyrythm	$s1$ $s2$	applique le rythme de $s1$ à $s2$ en boucle

Table 1. Opérations sur les partitions

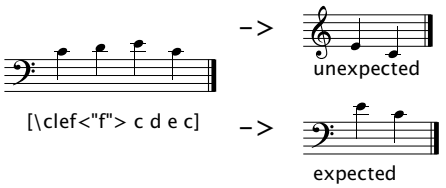

Figure 4. Cohérence de l'opération `tail`

et `[\clef<"g"> e c]`
donnera `[\clef<"g"> c d \clef<"g"> e c ]`
où la clef est répétée inutilement (figure 5) ce qui rend la lecture plus confuse.

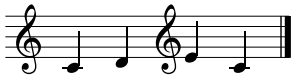

Figure 5. Mise en séquence *brute*

Certaines opérations peuvent également donner des résultats syntactiquement incorrects. Considérons le code suivant :

`[g \slur(f e) c]`

le partage de la partition en 2 après `f` donnerai

a) `[g \slur(f]` et **b)** `[e) c]`

i.e. comprenant des tags temporels inachevés. Nous utiliserons les termes tags à *fin-ouverte* pour faire référence au cas a) et tag à *début-ouvert* pour le cas b).

Ces quelques exemples illustrent une partie du problème et il y a beaucoup d'autres cas où la cohérence de la notation doit être préservée lors d'opérations sur les partitions.

3.3. Etendue temporelle des éléments

Nous proposons de traiter le problème en définissant une typologie des éléments de notation en fonction de leur étendue temporelle, et en spécifiant des règles de cohérence basées sur ces types.

Le format GMN fait une distinction entre les tags de position (comme `\clef` ou `\meter`) et les tags temporels (`\slur`, `\beam`, etc.) :

- les tags temporels ont une durée explicite : la durée des notes qu'ils contiennent,
- les tags de position sont de simples marques de notations à une position donnée.

Cette distinction n'est toutefois pas suffisante pour couvrir le problème : beaucoup de tags de position ont une durée implicite, qui s'étend généralement jusqu'à un signe de de notation similaire ou jusqu'à la fin de la partition. C'est le cas par exemple de la notation de l'intensité.

La table 2 présente une typologie simple des éléments de notation musicale, principalement fondée sur leur étendue temporelle. Nous définissons également la notion *de tag courant* : pour un type de tag d'étendue temporelle implicite donné (clef, meter...), il s'agit de la dernière valeur rencontrée.

Basées sur cette typologie, des précautions sont à prendre lors des opérations suivantes :

- calcul du début d'une partition :

  1) tous les tags d'étendue temporelle explicite

étendue temporelle	description	exemple
explicite	la durée est explicite dans la notation	slur, cresc
implicite	la durée s'étend jusqu'au prochain signe similaire ou jusqu'à la fin	meter, dynamics, key
autres	contrôle de la structure	coda, da capo, repeats
-	instructions de formatage	new line, new page
-	notations diverses	breath mark, bar

Table 2. Typologie des éléments de notation.

doivent être ouverts (i.e. les tags à *début-ouvert*, voir section 3.2)

2) les tags courants d'étendue temporelle implicite doivent être rappelés,

– calcul de la fin d'une partition :

3) les tags d'étendue temporelle explicite doivent être fermés (i.e. les tags à *fin-ouverte*),

– mise en séquence de partitions :

4) les tags d'étendue temporelle implicite de la deuxième partition doivent être supprimés quand ils correspondent à un tag courant.

3.4. Cohérence de la structure

Les éléments qui relèvent de la catégorie *autres / contrôle de la structure* peuvent également donner lieu à des incohérences de notation : une barre de début de répétition sans barre de fin de répétition, un *dal segno* sans *segno*, un *da capo al fine* sans *fine*, etc. Nous introduisons de nouvelles règles pour la consistance des barres de répétitions. Nous définissons tout d'abord une barre de répétition *en suspens* comme le cas d'une barre de début de répétition sans barre de fin correspondante.

5) lors du calcul de la fin d'une partition, les barres de répétition *en suspens* doivent être fermées avec une barre de fin de répétition,

6) dans le cas de barres de début de répétition successives, seule la première sera retenue,

7) dans le cas de barres de fin de répétition successives, seule la dernière sera retenue.

Il n'y a pas de préconisation particulière pour les autres éléments de contrôle de la structure : les incohérences qui peuvent se produire sont ignorées mais ce choix préserve la réversibilité des opérations.

3.5. Réversibilité des opérations

Les règles définies ci-dessus résolvent la majorité des problèmes de notation mais ne permettent pas d'inverser les opérations : considérons une partition comportant une liaison qui est coupée en son milieu et rassemblée par une mise en séquences. Le résultat comportera deux liaisons (figure 6) en raison des règles **1)** et **3)** qui forcent l'ouverture des tags à *début-ouvert* et la fermeture des tags à *fin-ouverte*.

La résolution du problème requiert le support du langage GMN : nous introduisons un nouveau paramètre

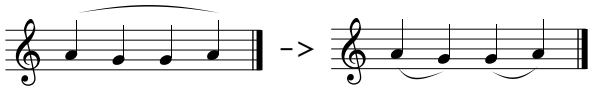

Figure 6. Une partition découpée et remise en séquence

pour les tags d'étendue temporelle explicite, conservant l'histoire du tag et indiquant un antécédent de type *fin-ouverte* et/ou *début-ouvert* . Ce paramètre est de la forme :

```
\tag<open="type">
```

où `type` est parmi [`begin`, `end`], correspondant respectivement à des antécédents *début-ouvert* et *fin-ouverte*.

Nous introduisons ensuite une nouvelle règle pour la composition de partitions, qui nécessite de définir des *tags adjacents* comme des tags placés sur la même voix et qui ne sont séparés par aucune note ou accord.

8) les tags *adjacents* similaires qui portent un paramètre `open` s'annulent mutuellement quand le premier est de type *fin-ouverte* et le second *début-ouvert*.

Ainsi, une opération rencontrant une forme de type :

```
\anytag<open="end">(f g)
\anytag<open="begin">(f e)
```

la transformera en :

```
\anytag(f g f e)
```

4. CONCLUSION

La complexité de la notation musicale repose en grande partie sur le grand nombre d'éléments de notation et sur leur statut hétérogène. La typologie que nous avons proposé (table 2) en est une simplification arbitraire, destinée à couvrir les besoins d'opérations de composition de partitions. Elle n'est pas représentative de cette complexité mais comme elle se base sur la sémantique de la notation, elle peut être appliquée à tout format de représentation de la notation musicale. Hormis les règles de réversibilité définies en 3.5 qui nécessitent le support du langage de représentation de la musique pour opérer, toutes les autres règles sont indépendantes du format GMN.

Les opérations sur les partitions peuvent se révéler très utiles dans le cas de traitements par lot (par exemple : séparation des voix d'un conducteur, extraction d'extraits, etc.). Les opérateurs présentés en table 1 permettent ce type de traitement, mais ouvrent également la voie à de nouvelles approches créatives de la musique.

5. REFERENCES

[1] H. Hoos, K. Hamel, K. Renz, and J. Kilian. The GUIDO Music Notation Format - a Novel Approach for Adequately Representing Score-level Music. In *Proceedings of the International Computer Music Conference*, pages 451–454. ICMA, 1998.

[2] H. H. Hoos and K. A. Hamel. The GUIDO Music Notation Format Specification - version 1.0, part 1 : Basic GUIDO. Technical report TI 20/97, Technische Universitat Darmstadt, 1997.

[3] Han-Wen Nienhuys and Jan Nieuwenhuizen. LilyPond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, May 2003.

[4] Han-Wen Nienhuys. Lilypond, automated music formatting and the art of shipping. In *Forum Internacional Software Livre 2006 (FISL7.0)*, 2006.

[5] C. Daudin, D. Fober, S. Letz, and Y. Orlarey. The Guido Engine - a toolbox for music scores rendering. In *Proceedings of the Linux Audio Conference 2009*, pages 105–111, 2009.

[6] D. Fober, S. Letz, and Y. Orlarey. Open source tools for music representation and notation. In *Proceedings of the first Sound and Music Computing conference - SMC'04*, pages 91–95. IRCAM, 2004.

[7] Andreas Egler Daniel Taupin, Ross Mitchell. Musixtex using tex to write polyphonic or instrumental music.