IN213

Langage de programmation

Agathe BEUCHER

26/04/2024

Sommaire

1 Introduction

1.1 Tout savoir sur les partitions

Une partition est constitué de nombreux éléments chacun important à l'interprétation de la musique :

— **La portée** : Elle est composée de 5 lignes et de 4 espaces.Parfois certaines notes graves ou aigües doivent être écrites hors de la portée de 5 lignes, on ajoute alors des bout de lignes supplémentaires autour de ces notes

— **La clé** : Très importante en musique, elle indique aux notes où se disposer sur la portée. On en retrouve deux principales :

1. La clé de sol : Indique que le sol s'ecrit sur la deuxième ligne de la protée en partant du bas, utilisé par la guitare, le violon, ou encore la flûte.
2. La clé de fa (4ème ligne) : indique que la note sur la quatrième ligne est un fa, utilisé par le violoncelle ou encore la basse.

On retrouve encore la clé d'Ut, beaucoup moins utilisée.

— **Les notes** : il existe sept notes naturelles (do ré mi fa sol la si) (ou encore A B C D E F G) qui représentent un octave.

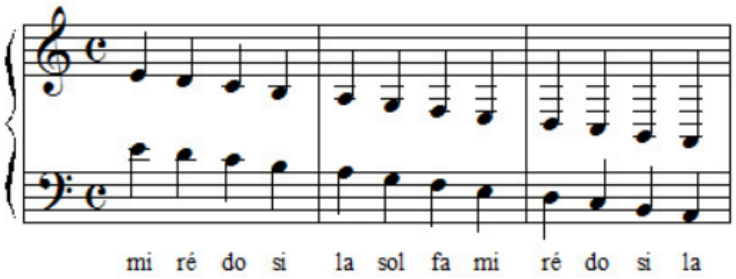


FIGURE 1 – Octave

— **La valeur des notes** : La valeur est ce qu'on appelle la durée des notes : elles dépendant graphiquement de si la note est pleine/vide/avec un queue à crochet/sans queue :

— la ronde : elle dure 4 temps, elle est vide et sans queue
— la blanche : elle dure 2 temps, elle est vide avec queue
— la noire : elle dure un temps, elle est pleine et droite
— la croche : elle dure un demi temps, elle est plein, droite avec un crochet

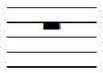
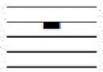
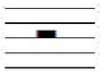
Figures de notes	Symboles	Figures de silences	Symboles	Valeurs
La ronde		La pause		4 temps
La blanche pointée		La pause		3 temps
La blanche		La demi-pause		2 temps
La noire		Le soupir		1 temps
La croche		Le demi-soupir		1/2 temps
La double-croche		Le huitième de soupir		1/4 temps

FIGURE 2 – Valeur des notes

— **Signature de temps** : Ce sont deux chiffres en début de portée qui permettent de savoir comment positionner les barre de mesure qui divisent la porté en **mesure**. Des mesures 4/4 contiennent 4 noires.

— **Les pauses** : un symbole de pause différent existe pour chaque note (voire Figure 2)(pause, demi-pause, soupir...)

— **Les points et liaisons** : Elles permettent de modifier la durée d'une note : le point après une note augmente la durée de la note de la moitié de se valeur, la liaison entre deux notes ajoute la valeur de la seconde note à la première.

— **L'armure** : Altérations constitutives d'une tonalité, écrites immédiatement après la clef, affectant toutes les notes de même nom, quelle que soit leur octave, et dont l'effet se prolonge pendant toute la durée du morceau. (dièze et bémol)

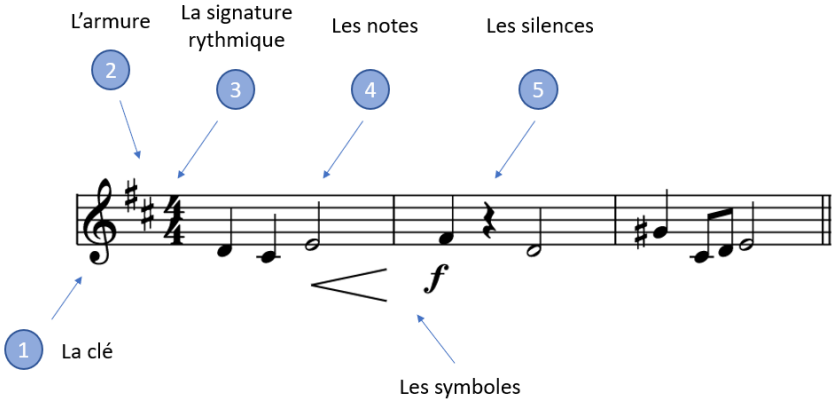


FIGURE 3 – Elements d'une partition

On retrouve donc 8 éléments principaux constitutifs d'une partition dont il faudra tenir compte.

2 Cahier des charges

2.1 Portée du projet

Mon projet consiste à développer un langage dédié à la représentation de partitions musicales. Ce langage doit permettre de décrire des notes, des modificateurs de hauteur et de rythme, ainsi que des structures musicales complexes. Il doit également offrir des fonctionnalités pour factoriser des parties de morceaux et pour générer du son à partir de la représentation musicale.

2.2 Fonctionnalités

— **Description des notes** : Le langage doit permettre la représentation précise des hauteurs, des durées et des caractéristiques des notes musicales.

— **Modificateurs de hauteur et de rythme** : Il doit être possible d'appliquer des altérations à la hauteur et à la durée des notes, telles que les altérations accidentelles, les nuances dynamiques, etc.

— **Structures musicales complexes** : Le langage doit prendre en charge la gestion de structures telles que les mesures, les motifs répétés, les transitions, les variations, etc.

— **Factorisation des parties de morceaux** : Des mécanismes doivent être mis en place pour permettre la réutilisation et la manipulation efficaces de parties de morceaux, facilitant ainsi la composition et l'arrangement.

— **Génération de son** : Le langage doit offrir des fonctionnalités pour générer du son à partir de la représentation musicale, permettant aux utilisateurs d'entendre le résultat de leurs compositions.

2.3 Contraintes techniques

Le langage doit être conçu pour être facilement interprétable par des logiciels de composition musicale et des synthétiseurs. Il doit être extensible afin de permettre l'ajout de nouvelles fonctionnalités à l'avenir. La syntaxe et la sémantique du langage doivent être clairement définies pour assurer sa cohérence et sa facilité d'utilisation.

3 Analyseur lexical : *musiclexeur.mll*

Un lexer est un programme qui transforme une séquence de caractères en une séquence de "tokens". On veut créer un langage minimaliste, constitué des éléments suivants :

— Le titre

— Le compositeur

— Le tempo

— la signature rythmique

— L'armure (*optionnel*)

3.1 Définition des termes

On utilise dans la syntaxe du langage certains termes et notation qu'il convient de définir dans un premier temps :

1. **Signature de temps** : Indication du nombre de temps par mesure et de la valeur de la note. (voir Tout savoir sur les partitions (c.f. 1.1)
2. **Note** : Représentation d'une note musicale comprenant la **hauteur** (pitch), **l'octave**, une **altération accidentelle** optionnelle et la **durée**.
3. **Silence** : Pause dans la musique, indiquée par un caractère spécial ("_") suivi d'une durée.
4. **Barre de mesure** : Marqueur de division dans la partition musicale. .

3.2 Synthaxe du langage

La syntaxe du langage est définie en utilisant des règles de définition pour chaque élément musical identifiable :

— Pitch : Défini comme une lettre de A à G, représentant la hauteur musicale.

— Digit : Chiffre de 0 à 9, utilisé pour représenter l'octave et la durée.

— Octave : Un ou plusieurs chiffres représentant l'octave de la note.

— Durée : Durée de la note, composée d'un ou plusieurs chiffres, éventuellement suivis d'une décimale pour une précision supplémentaire.

— Silence : Représenté par un caractère spécial suivi d'une durée.

— Space : Caractères d'espacement comme l'espace ou la tabulation.

— Time Signature : Indiqué par deux chiffres séparés par une barre oblique, représentant le nombre de temps par mesure et la valeur de la note.

— Accidental : Optionnel, représenté par un bémol ou un dièse.

3.3 Règle de tokenisation

Les règles de tokenisation définissent comment les caractères d'entrée sont interprétés pour identifier les éléments musicaux dans la partition :

— **Title, Composer, Tempo, Key Signature** : Ces règles identifient les métadonnées de la partition musicale, telles que le titre, le compositeur, le tempo et la signature de la clé.

— **Time Signature** : Identifie la signature de temps dans la partition.

— **Note** : Représente une note musicale, comprenant le pitch, l'octave, l'accidental et la durée.

— **Silence** : Représente une pause dans la musique, avec une durée spécifiée.

— **Bar** : Identifie une barre de mesure.

— **Newline** : Ignore les caractères de nouvelle ligne.

— **EOF** : Indique la fin de fichier.

— **Unexpected Character** : Gère les caractères inattendus dans la partition.

4 Analyseur synthaxique : *musicparser.mly*

Ces tokens sont utilisés par un parser pour construire une structure de données : On y définit un ensemble de règle de grammaire :

1. **Règle main** : définit la structure principale de la partition musicale. Elle indique que la partition musicale est composée d'une séquence de notes, suivie de la fin du fichier (EOF).Cela signifie que le parseur doit analyser une structure musicale suivie de la fin du fichier.

```
main:
  | music EOF { $1 }
```

2. **Règle notes** : définit la syntaxe des séquences de notes dans la partition musicale. Une séquence de notes peut commencer par une note individuelle, suivie éventuellement d'une barre et d'autres notes (note BAR notes), ou simplement d'une note individuelle (note).

```
notes:
  | note { [$1] }
  | note BAR notes { $1 :: $3 }
  | note notes { $1 :: $2 }
```

3. **Règle note** : définit la syntaxe d'une note individuelle dans la partition musicale. Une note peut être une note avec sa durée (NOTE) ou un silence avec sa durée (SILENCE).

```
note:
  | NOTE { ... }
  | SILENCE { ... }
```

4. **Règle music** : définit la structure globale de la partition musicale, qui commence par un titre, suivi du nom d'un compositeur, un tempo, une signature de temps, une éventuelle signature de clé et une séquence de notes.

```
music:
  | TITLE COMPOSER TEMPO TIME_SIGNATURE key_signature_opt notes
  { ... }
```

5 AST partition : *musicast.ml*

Le fichier musicast.ml fournit des outils pour traduire des partitions musicales et les convertir en documents XML compatibles avec le format MusicXML :

5.1 MusicXML

MusicXML est un format de fichier XML qui est conçu pour représenter des partitions musicales de manière interchangeable entre différents logiciels musicaux.
MusicXML est devenu un format standard pour l'échange de partitions musicales entre différents logiciels de notation musicale, ce qui permet aux utilisateurs de partager et de travailler sur des partitions avec une variété de programmes.
Il permet également de représenter une grande variété d'éléments musicaux, notamment les notes, les accords, les paroles, les nuances, les indications de tempo, les doigtés, les articulations, les modifications de clé, les changements de mesure... Il est basé sur XML, donc chaque élément musical est représenté par des balises XML avec des attributs et des valeurs associées. Il est intéressant car de nombreux logiciels de notation musicale prennent en charge l'importation et l'exportation de fichiers MusicXML, en particulier Musescore, que 'lon va utiliser ici pour générer la partition en pdf.
MusicXML est donc un format polyvalent et largement utilisé pour représenter des partitions musicales de manière standardisée.

Cet exemple illustre une seule mesure contenant une note de Do (C) à l'octave 4, avec une durée de quatre temps et un type de note de noire (quarter note) :

Exemple de synthaxe XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 3.1
Partwise//EN" "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.1">
  <part-list>
    <score-part id="P1">
      <part-name>MusicXML Part</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">

      <attributes>
        <divisions>4</divisions>
        <key>
          <fifths>0</fifths>
          <mode>major</mode>
        </key>
        <time>
```

```
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>quarter</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

5.2 Module de manipulation

On reprend et définit plusieurs types et fonctions pour représenter et manipuler des notes de musique, des mesures et des morceaux de musique :

```
type note =
  | Silence of float
  | Note of {
    pitch : string;
    octave : int;
    accidental : string;
    duration : float;
    tie : string option;
  }
type measure = note list
type music = {
  title : string;
  composer : string;
  tempo : string;
  beats : int;
  beat_type : int;
  key_signature : string option;
  notes : note list;
}
```

On définit ensuite des fonctions de conversion pour traduire notre langage vers le langage de musicXML :

— key_signature_to_altered_notes : Convertit une signature de clé (par exemple, "C#" ou "Fb") en une liste des notes modifiées, soit renvoie la liste des notes affectées par les altérations.

— string_of_measure : Convertit une mesure en une chaîne de caractères.

— **tempo_string_to_float** : Convertit une indication de tempo en valeur numérique de bpm :

```
| "largo" -> 40.0
| "adagio" -> 60.0
| "andante" -> 80.0
| "moderato" -> 100.0
| "allegro" -> 120.0
| "presto" -> 140.0
| _ -> failwith "Unsupported tempo"
```

— **duration_to_type** : Convertit une durée en type de note.

```
| 4.0 -> "whole"    (* ronde *)
| 2.0 -> "half"     (* blanche *)
| 1.0 -> "quarter"  (* noire *)
| 0.5 -> "eighth" (* croche *)
| 0.25 -> "sixteenth" (* double croche *)
| _ -> failwith "Unsupported duration"
```

— **notes_to_measures** : Convertit une liste de notes en une séquence de mesures. Elle prend trois arguments : le nombre de battements par mesure (`beats`) et le type de battement (`beat_type`) donnés par la signature rythmique, et une liste de notes (notes). Elle renvoie une liste de mesures, où chaque mesure est représenté par une liste de notes. La fonction commence par calculer la durée maximale d'une mesure (`max_duration`) en multipliant le nombre de battements par la durée du type de battement. Ensuite, elle définit une fonction auxiliaire récursive aux qui prend quatre arguments : une liste de mesures accumulées (measures), une mesure en cours de construction (measure), la durée totale de la mesure en cours (measure_duration) et la liste des notes restantes à traiter.

— **note_to_xml** : Convertit une note en un élément XML. Elle prend en entrée une signature de clé (`key_signature`) et une note, qui peut être soit un silence (`Silence duration`), soit une note musicale (`Note pitch; octave; accidental; duration; tie`), et renvoie une chaîne de caractères représentant la note en XML.

— **measure_to_xml** : Convertit une mesure en un élément XML.

— **music_to_xml** : Convertit une partition musicale complète en un document XML.

5.3 *main.ml*

Ce script OCaml est utilisé pour analyser un fichier de musique, le convertir en XML, puis convertir ce fichier XML en PDF :

- **parse** ouvre un fichier, crée un tampon lexical à partir de ce fichier, puis tente de l'analyser avec parse_with_error
- **parse_with_error** analyse un tampon lexical (lexbuf). Si l'analyse réussit, elle affiche le résultat analysé et le renvoie. Si l'analyse échoue, elle affiche un message d'erreur et termine le programme avec un code de sortie de -1.
- **convert_to_pdf** tente de convertir un fichier XML en PDF en utilisant la commande `"mscore"`. Si la conversion réussit, elle affiche un message de succès. Si la conversion échoue, elle affiche un message d'échec.

6 AST son : sonast.ml

Pour jouer de la musique à partir d'une structure de données représentant une partition musicale, on définit les fonctions :

— **play_sound** joue un son de fréquence f pendant une durée l, avec la commance `"play"`

— **silence** crée un silence d'une durée l, avec la commande `"sleep"`

— **beats_to_ms** convertit une durée en battements en une durée en millisecondes, en fonction du tempo en battements par minute.

— **play_note** joue une note de musique. Si la note est un silence, elle crée un silence de la durée appropriée. Si la note est une note de musique, elle calcule la fréquence de la note en fonction de sa hauteur, de son octave et de la signature de clé de la musique, puis elle joue le son de cette fréquence pendant la durée appropriée.

— **play_bar** est une fonction qui joue une mesure de musique. Elle joue chaque note de la mesure l'une après l'autre.

— **play_music]** est une fonction qui joue une musique entière. Elle joue chaque note de la musique l'une après l'autre.

De la même manière que pour la partition, un fichier *son.ml* est responsable de créer, d'analyser un tampon lexical, et d'appeler les fonctions précédentes pour créer un son.

7 Exemple

7.1 Mon langage

C'est parti pour créer notre propre partition de musique !

7.1.1 Pré-requis : header

Chaque script dois commencer par un header sous ce format composé d'un "Title :", "Composer :", "Tempo :", "nmb/nmb". Le champ d'altération "Key_signature" est optionnel.

ATTENTION! La casse compte!

```
1   Title:"Frère Jacques"
2   Composer:"Jean-Philippe Rameau"
3   Tempo:"allegro"
4   4/4
5   Key_signature:"-1"
```

FIGURE 4 – Synthaxe de header

7.1.2 La musique

Une fois que que ce header est bien configuré, il convient de coder vos notes! Chaque note est entourée de "Bar"="|", avec sa durée et son octave comme ceci :

$$|\,\text{A4-1.0}\,|\,[1]$$

Ici est codé une noire, le **La 440Hz** (octave 4). On peut donc les enchaîner pour créer la partition. Les barres de mesures s'ont ajoutés automatiquement en fonction de la durée des notes et du la signature rythmique. [2]

```
7   C5-1.0|D5-1.0|E5-1.0|C5-1.0|
8   C5-1.0|D5-1.0|E5-1.0|C5-1.0|
9   E5-1.0|F5-1.0|G5-2.0|
10  E5-1.0|F5-1.0|G5-2.0|
11  G5-0.5|A5-0.5|G5-0.5|F5-0.5|E5-1.0|C5-1.0|
12  G5-0.5|A5-0.5|G5-0.5|F5-0.5|E5-1.0|C5-1.0|
13  C5-1.0|G4-1.0|C5-2.0|
14  C5-1.0|G4-1.0|C5-2.0|
```

FIGURE 5 – Exemple de code de notes

Pour écouter votre musique, consultez le README github...

1. La Bar en début et en fin de partition est négligeable.
2. Dans le format MusicXML, une nouvelle ligne est généralement représentée par une nouvelle balise <system>. Cependant, le format MusicXML ne permet pas de contrôler directement le nombre de mesures par ligne. C'est généralement le logiciel de notation musicale, dans notre cas MuseScore qui gère automatiquement la mise en page, y compris le nombre de mesures par ligne. Si je veux contrôler le nombre de mesures par ligne, je devrais le faire dans le logiciel de notation musicale après avoir importé le fichier MusicXML...

7.2 Partition généré

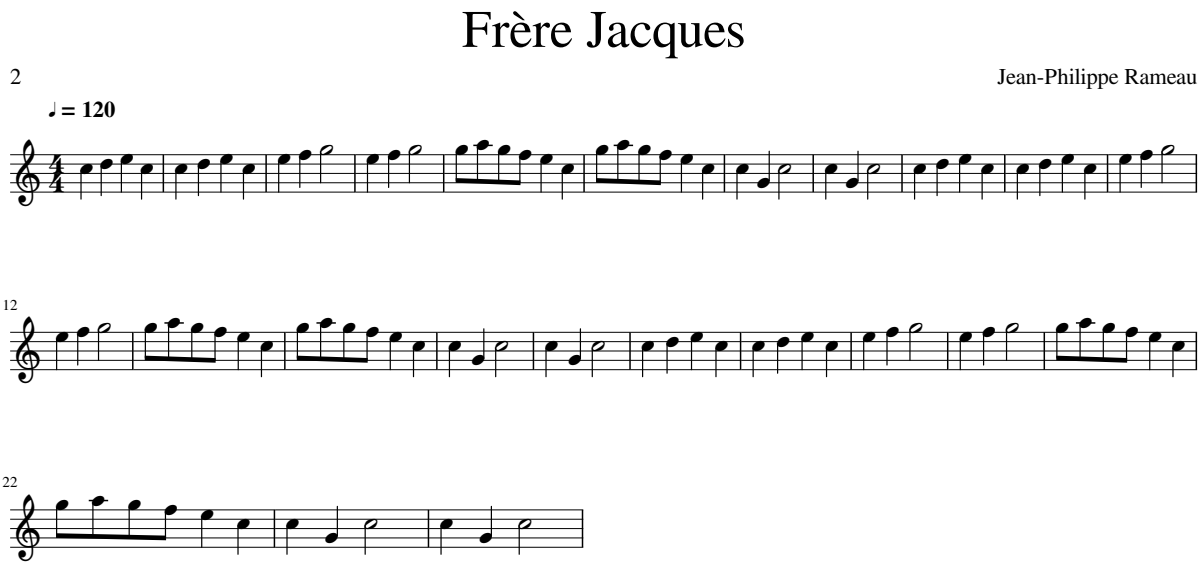


FIGURE 6 – Exemple de génération de partition

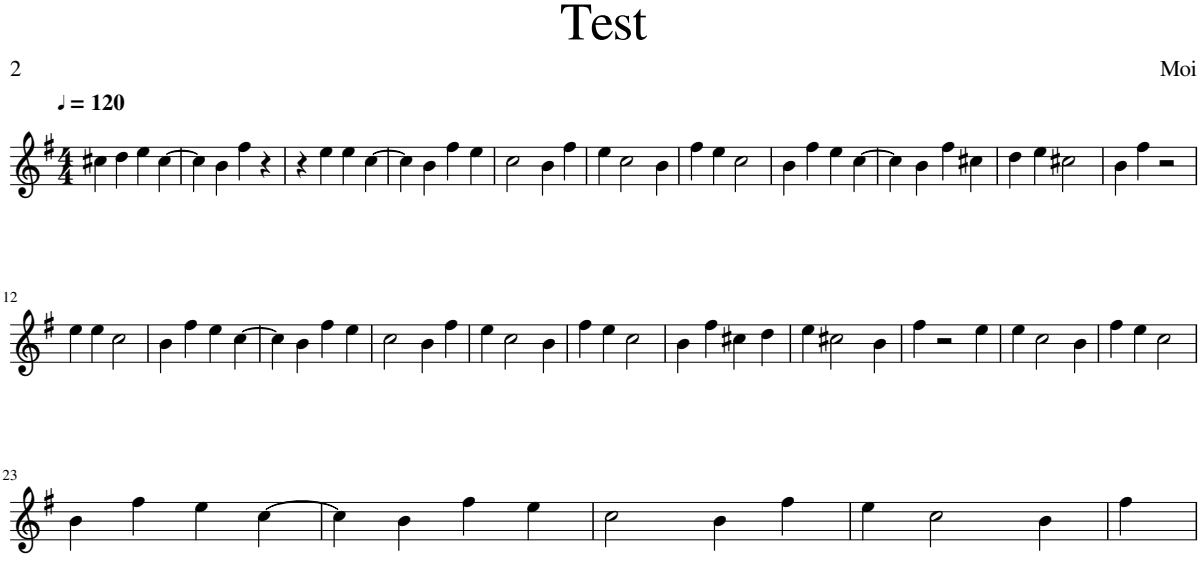


FIGURE 7 – Exemple de génération de partition avec silence et dieze