Programmation par contraintes relationnelles pour l'analyse et la composition musicales

Sascha Van Cauwelaert

Université Catholique de Louvain
ICTEAM/PLDC

04 Novembre 2011

Plan de l'exposé

- Programmation par contraintes relationnelles
- Représentation d'une partition par une relation
- Représentation d'une transformation par une relation
- Exemple
- GeLisp : interface évolutive

Introduction

- « One of the difficulties [...] was the need for a more structured representation of musical data. Handling the parameters independently [...] made it rather difficult for parameters to interact. In future a rich, hierarchically organized data structured for notes (a bundle of information containing many parameters like pitch, duration, loudness, [...] etc ...) should help to create even more « intelligent » rules. »

Johannes Kretz, in OM composer's book vol. I

Plan de l'exposé

- **Programmation par contraintes relationnelles**
- Représentation d'une partition par une relation
- Représentation d'une transformation par une relation
- Exemple
- GeLisp : interface évolutive

Programmation par contraintes relationnelles

- Module de Gecode, en développement actuellement
  - Gecode : librairie C++ permettant la programmation par contraintes
- Développé dans le groupe de Peter Van Roy
  - Peter Van Roy : professeur à l'Université Catholique de Louvain
- Auteur : Gustavo Gutiérrez

Programmation par contraintes relationnelles

- Relations en tant que variables
  - Relation d'arité n : ensemble de n-tuples d'entiers
  - Valeur d'une variable = une relation
  - Domaine d'une variable = ensemble fini de relations
  - Etend variables ensemblistes, déjà bien développées

$$\{\langle 1, 2\rangle, \langle 2, 3\rangle, \langle 1, 1\rangle\}$$

1	2
2	3
1	1

Programmation par contraintes relationnelles

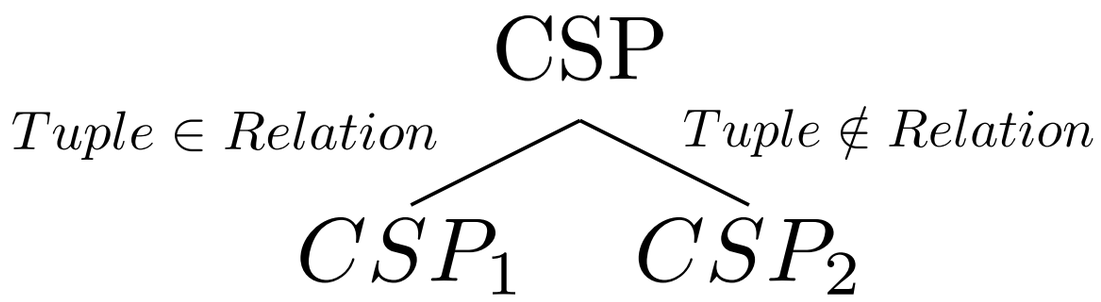
- Relation «ground»
  - Relation non variable (ensemble fixé de n-tuples)
  - De manière interne, représentée par un SROBDD (Shared Reduced Ordered Binary Decision Diagram)

- Domaine d'une variable relationnelle
  - similaire aux ensembles d'entiers : approximé par deux bornes (i.e. deux relations ground)
  - *glb* : borne inférieure, qui contient tous les tuples obligatoirement présents dans la relation
  - *lub* : borne supérieure, qui contient tous les tuples potentiellement présents dans la relation
  - $glb \subseteq lub$; la variable est assignée à une valeur lorsque $glb = lub$

Programmation par contraintes relationnelles

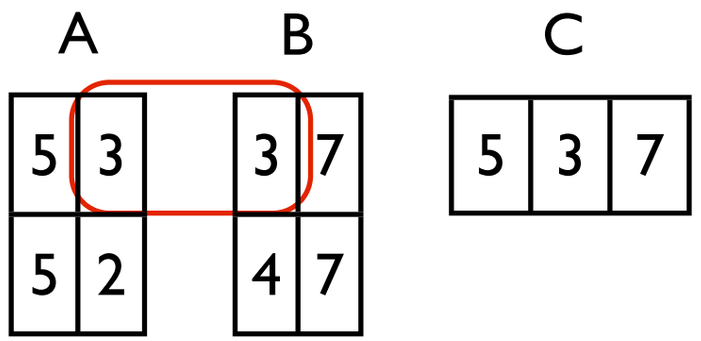
- Contraintes
  - Algèbre relationnelle ($\bowtie, \pi, ...$) et théorie des ensembles ($\cup, \cap, ...$)
  - $\bowtie$ = jointure relationnelle , π = projection relationnelle

- Stratégie de recherche
  - «Naïve» pour le moment : une branche de l'arbre de recherche inclue un tuple, l'autre l'exclue

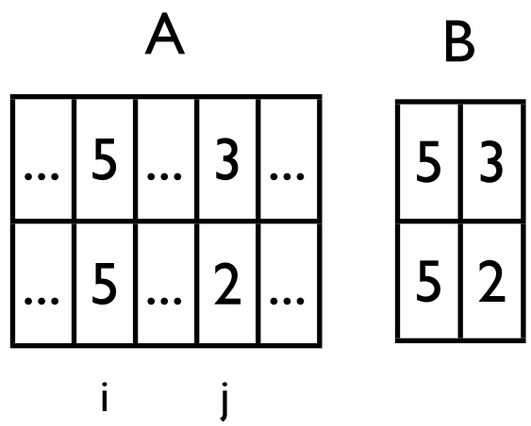


Deux contraintes importantes : jointure et projection relationnelles

- Jointure $\bowtie$ $\quad A \bowtie_1 B = C$



- Projection π $\quad \pi_{i,j}(A) = B$



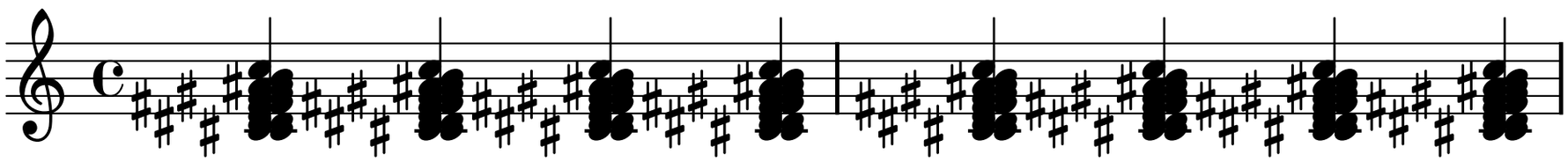
Plan de l'exposé

- Programmation par contraintes relationnelles
- **Représentation d'une partition par une relation**
- Représentation d'une transformation par une relation
- Exemple
- GeLisp : interface évolutive

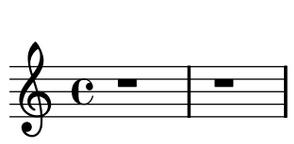
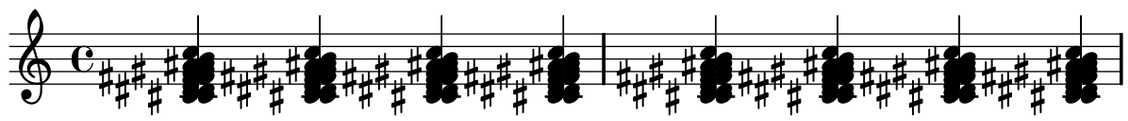
Représentation d'une partition par une relation

- Une note = un tuple
  - ex : $\langle pitch, onset, duration, velocity, channel \rangle$
- Une relation = ensemble quelconque de notes
  - ex : relation ground totale = une partition musicale où toutes les notes possibles sont jouées

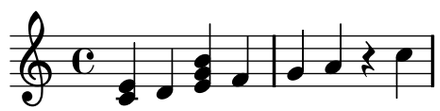
$$P = \{\langle pitch, onset \rangle\}$$



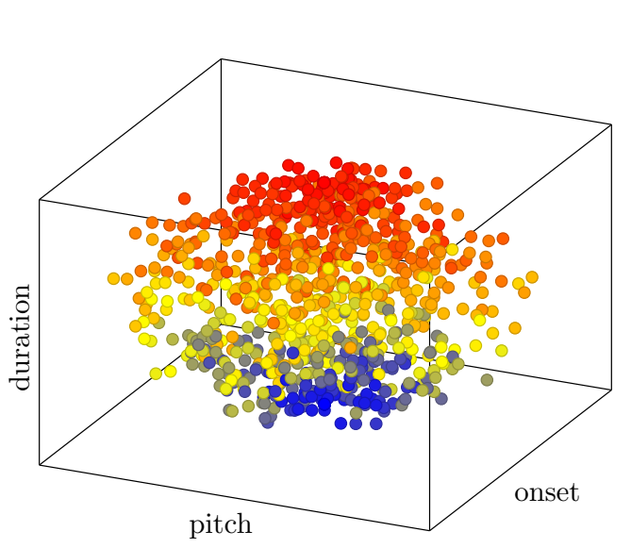
Représentation d'une partition par une relation

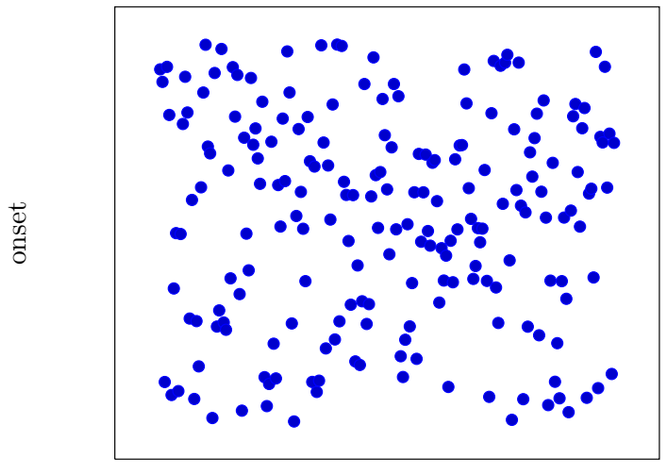
- Une relation = ensemble quelconque de notes
  - Une variable permet de représenter un ensemble quelconque de notes, précisément représentées par des tuples
    - glb : 
    - lub : 

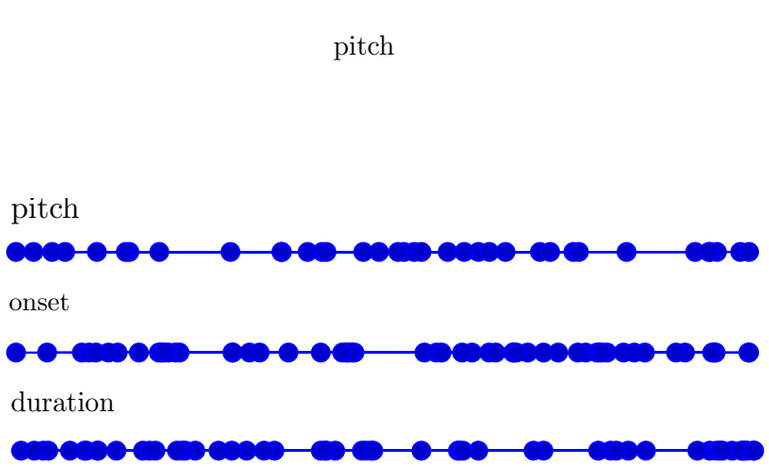
$$P = \{\langle pitch, onset \rangle\}$$

  - Chaque note est représentée par un tuple binaire
  - La variable, une fois assignée, est une partition fixée
    - ex : 

Représentation d'une partition par une relation







Alors que l'expressivité augmente, le nombre d'éléments possibles explose de manière exponentielle (Effet du «fléau de la dimension»)

Difficulté combinatoire

- Explosion combinatoire lors de la résolution à maitriser
  - Le nombre de tuples possibles au sein d'une relation d'arité m, pour un ensemble S de valeurs potentielles pour chacune des composantes, est de $\#S^m$
  - Pour une variable relationnelle, le nombre total de valeurs potentielles est une double exponentielle : $2^{(\#S)^m}$
  - Pour n relations comme celle-là combinées, le total est de $2^{n*(\#S)^m}$
- Pistes de résolution
  - (1) SROBDDs permettent parfois une représentation très efficace en mémoire des relations
  - (2) Certaines contraintes peuvent élaguer énormément l'arbre de recherche

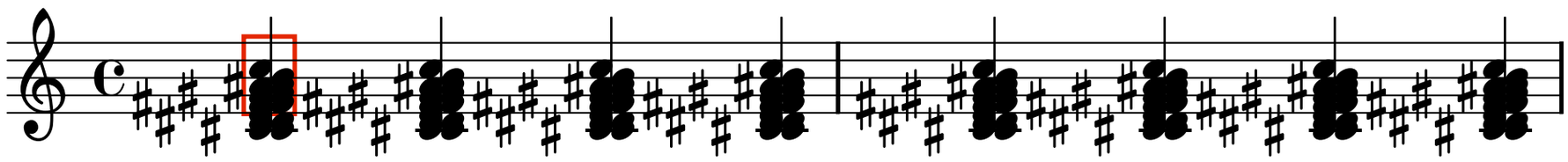
Avantages des relations

- Expressivité simplifiée et naturelle
  - On impose des contraintes sur des structures musicales plutôt que sur des paramètres entiers isolés
- Représentation plus précise
  - Notes représentées par des tuples
- Nombre de notes non fixé a priori
  - Cardinalité non-fixée a priori
- Conséquence : nouveaux problèmes musicaux et contraintes
  - Problèmes globaux impliquant plusieurs paramètres sonores

Récupération d'une partie de partition

$$SubScore = Score \cap (Rel \times \mathcal{U}_m)$$

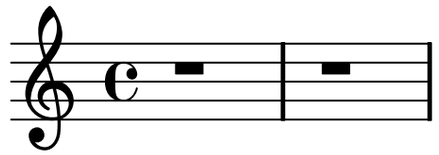
- $Score$: partition initiale
- Rel : relation contenant un ou plusieurs tuples respectant une propriété donnée
- $\mathcal{U}_m$: Univers d'arité m, i.e. ensemble de tous les tuples d'arité m possibles



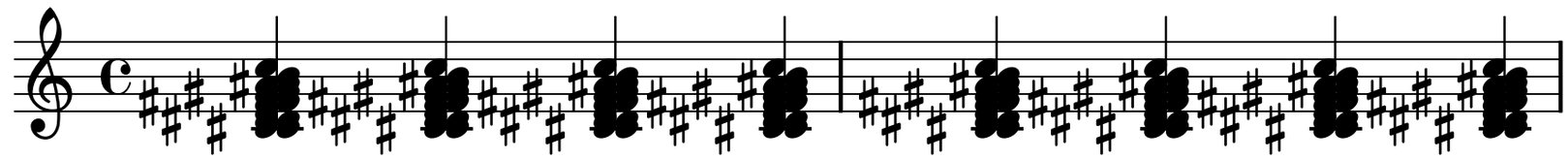
- Les notes peuvent être récupérées par la contrainte qu'elles soient présentes ou non dans la solution finale
- On peut alors les contraindre à l'avance sans savoir si elle font partie de la partition finale
- Solution à l'«Inaccessible Score Context Problem» de Torsten Anders (ex: on sait à l'avance quelles sont les notes simultanées sans savoir si elles seront jouées ou non)

Exemple de contrainte simple

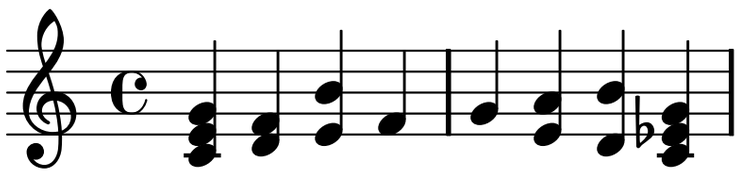
glb :



lub :



$$\bigwedge_{i=0}^{7} \#(r_{score} \bowtie \langle i \rangle) \leq 3$$



⋈	…	…
	60	I
	64	I
	67	I
I	71	I
	…	…

Plan de l'exposé

- Programmation par contraintes relationnelles
- Représentation d'une partition par une relation
- **Représentation d'une transformation par une relation**
- Exemple
- GeLisp : interface évolutive

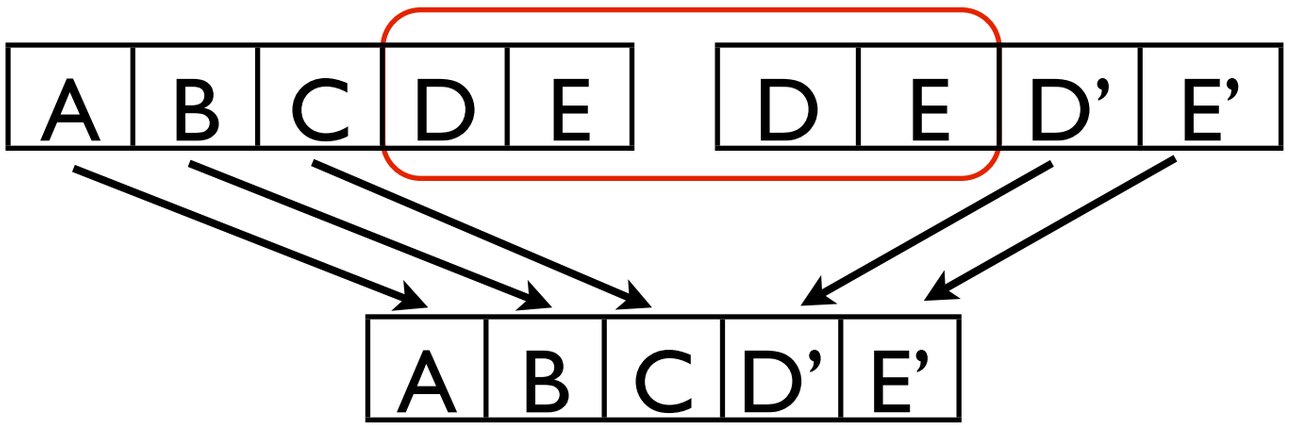
Représentation d'une transformation par une relation

- Transformation générique de partition :

$$\{\langle param_{init_1}, ..., param_{init_n}, param_{final_1}, ..., param_{final_n}\rangle\}$$

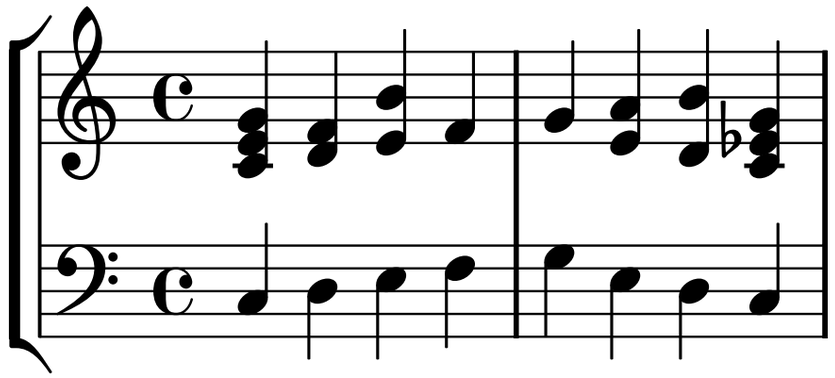
- Application de la transformation :

$$Partition' = \pi_{param_{nottrans}, param_{final}}(Partition \bowtie T)$$



Représentation d'une transformation par une relation

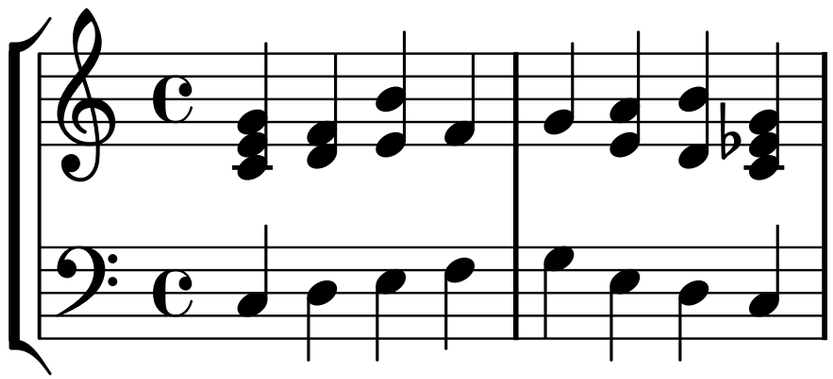
- Exemple : génération à partir d'une ligne de basse



- Fonction : cas particulier d'une relation
  - Fonction : une structure musicale ne possède qu'une seule transformée (image)
  - Relation : une structure musicale peut posséder plusieurs images
- Une transformation musicale peut aussi être variable ...

Transformation généralisée : lien musical

- Transformation : cas particulier d'une relation vue de manière unidirectionnelle
- De manière générale, une relation permet de lier deux structures musicales
- Plus généralement encore, n structures musicales ...

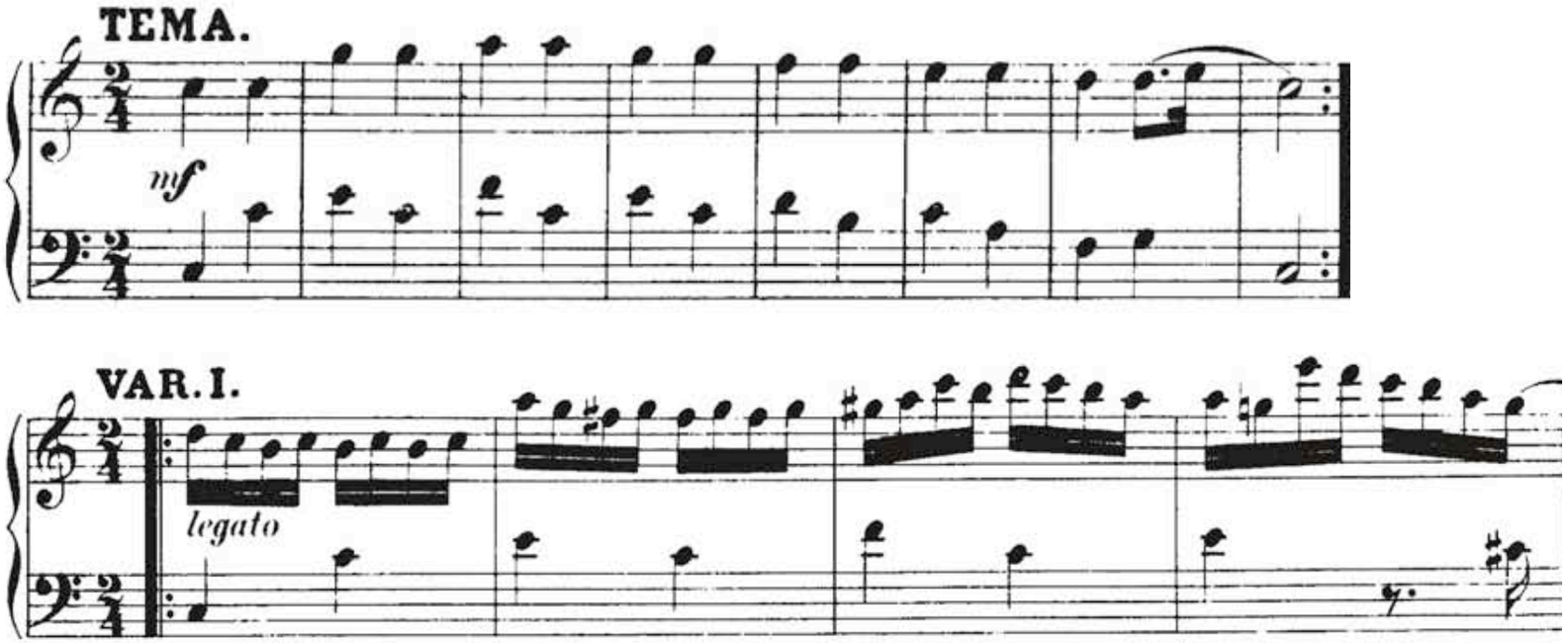


Récupération de transformations existantes : problème ouvert, lié au machine learning

- Exemple de transformation : variation à partir d'un thème
- Définir la transformation du thème en une variation est loin d'être trivial
  - Quels sont les paramètres musicaux à prendre en compte ?
  - Sur combien de notes s'applique une transformation ?
  - La transformation doit-elle être fonctionnelle ?

Récupération de transformations existantes : problème ouvert, lié au machine learning

- Quels sont les paramètres musicaux à prendre en compte ?
- Sur combien de notes s'applique une transformation ?
- La transformation doit-elle être fonctionnelle ?



Plan de l'exposé

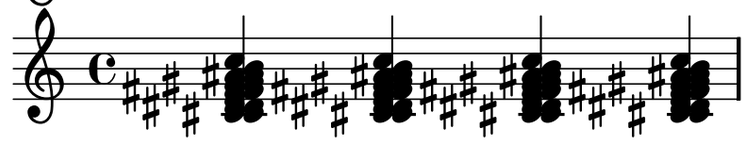
- Programmation par contraintes relationnelles
- Représentation d'une partition par une relation
- Représentation d'une transformation par une relation
- **Exemple**
- GeLisp : interface évolutive

Utilisation de relations ground

- Permettent de travailler à partir d'informations précalculées
  - Information qui n'est plus à calculer durant la recherche et qui peut être utilisée dans la résolution de problèmes
- Ex : Ensemble d'accords : $Chords = \{\langle pitch, index \rangle\}$
  - Do majeur : $\langle 60, i \rangle$; $\langle 64, i \rangle$; $\langle 67, i \rangle$
  - Ré majeur : $\langle 62, j \rangle$; $\langle 66, j \rangle$; $\langle 69, j \rangle$; $i \neq j$
  - ...
- Trouver ce type de relations fait partie de la recherche en cours

Exemple complet

- Contrainte
  - Imposer pour une partition quelconque (valeurs quelconques pour les durées, pitch, onsets, ... des notes et nombre de notes non fixé), on entend au maximum un et un seul accord d'un ensemble d'accords donné à chaque moment

- Exemple simple d'utilisation :
  - glb : 
  - lub : 

Exemple complet

- Relations fixées (ground)
  - Ensemble d'accords : $Chords = \{\langle pitch, index \rangle\}$
    - Do majeur : $\langle 60, i \rangle$; Ré majeur : $\langle 62, j \rangle$; ...
      $\langle 64, i \rangle$ $\langle 66, j \rangle$
      $\langle 67, i \rangle$ $\langle 69, j \rangle$

- T, qui permet de transformer la partition initiale en une partition où les notes de durée n sont remplacées par n notes consécutives de durée 1:

$$T = \{\langle X, Y, Z \rangle\} : Z \in [X, X + Y - 1] \wedge Y > 0$$

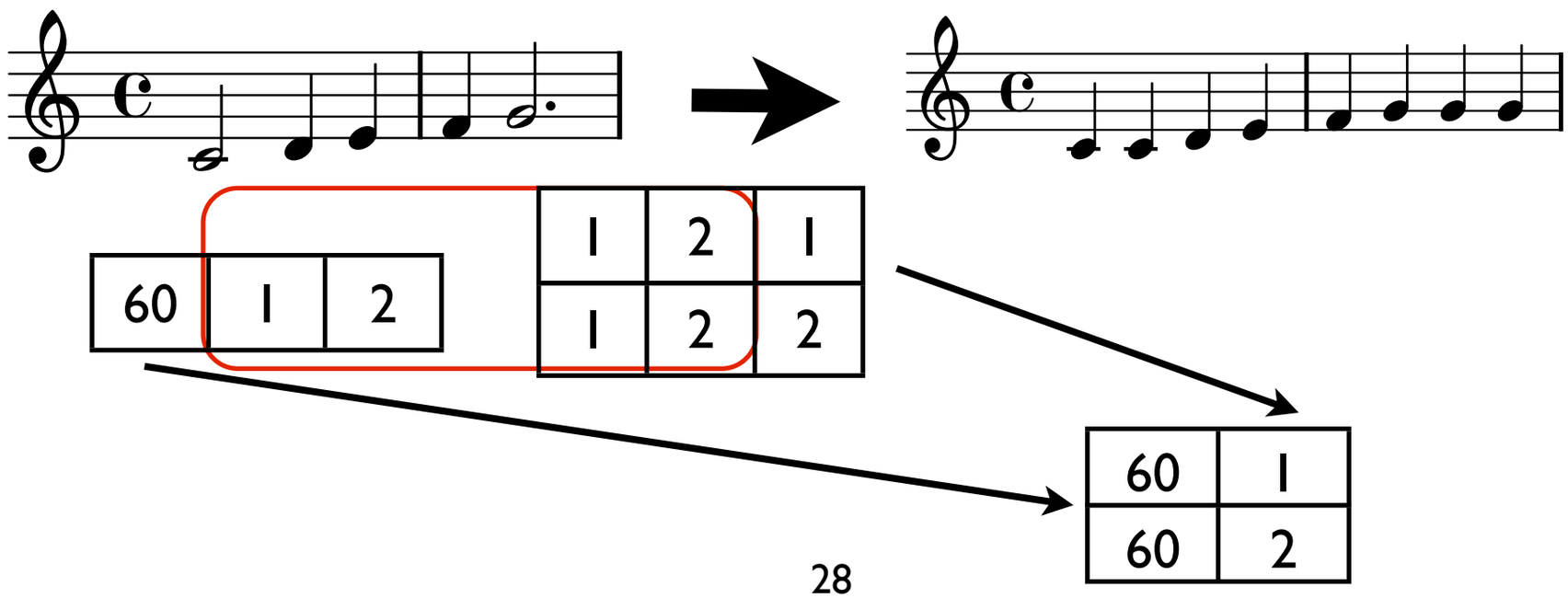
onset	durée	onset'
5	3	5
5	3	6
5	3	7

Exemple complet

- Partition initiale (variable)

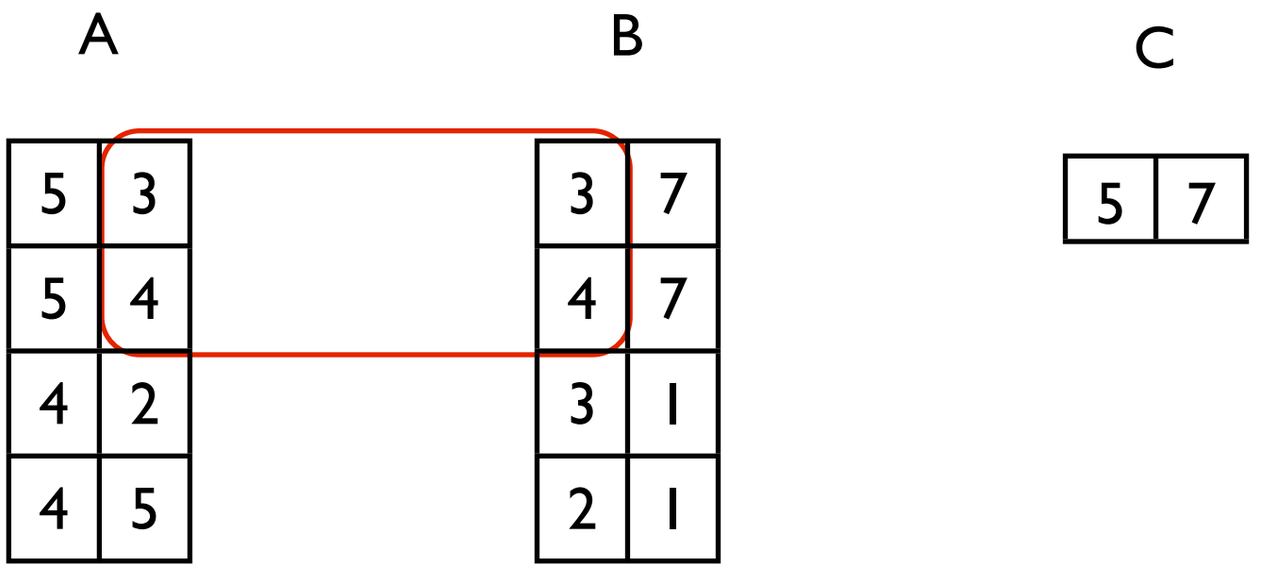
$$Partition = \{\langle pitch, onset, duration\rangle\}$$

- Partition transformée
  - Les notes de durée n sont remplacées par n notes consécutives de durée 1
  - $Partition' = \pi_{pitch,Z}(Partition \bowtie T)$

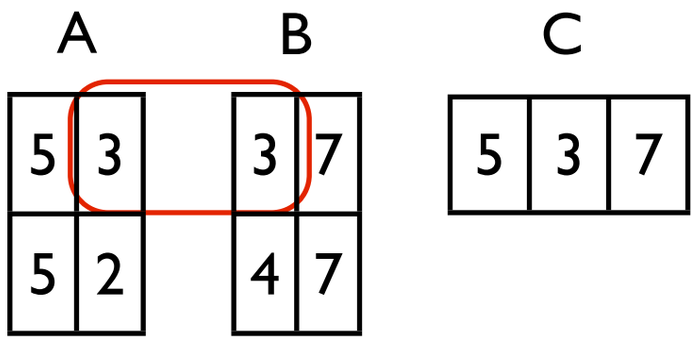


Exemple complet

- Contrainte FollowAll(A,B,C) :



- Jointure : ⋈



Exemple complet

- Expression de la contrainte

$$FollowAll(Partition', Chords, OnsetIndex)$$

$$\forall\ onset_i \in Onsets : \#(onset_i \bowtie OnsetIndex) \leq 1$$

Partition’

d	60
d	64
d	67

Chords

60	i
64	i
67	i

DémInd

d	i

Exemple complet

- Analyse si la partition est fixée et l'ensemble d'accords est variable
  - Permet de connaître quel accord (au complet et uniquement) est entendu à un moment donné, pour toute la partition
- Si ni la partition, ni l'ensemble d'accords ne sont fixés à l'avance (mais bien bornés), utilisation des 2 outils durant la recherche

Plan de l'exposé

- Programmation par contraintes relationnelles
- Représentation d'une partition par une relation
- Représentation d'une transformation par une relation
- Exemple
- **GeLisp : interface évolutive**

GeLisp : interface évolutive

- Interface pour utiliser Gecode dans le langage Common Lisp
  - Common Lisp facilement utilisable dans OpenMusic
- Utilisation de SWIG : Simple Wrapper Interface Generator
- Mise à jour très simple de GeLisp lorsque Gecode évolue
  - Gecode est en constante et rapide évolution
- Version utilisable par d'autres dans le courant du mois de décembre 2011

A venir

- Contrainte Leitmotiv
- Analyse «intelligente» et utilisée durant la recherche
- Aide à l'orchestration, au réarrangement
- Autres structures relationnelles et contraintes
  - ex : représentation compositionnelle, qui permet d'éviter de faire des répétitions explicites (mauvais pour la compression des données de manière interne)
- Outils interactifs pour les compositeurs, basés sur mes recherches

Conclusion

- Idées générales présentées aujourd'hui :
  - Variables relationnelles pour la représentation de partitions/structures musicales
  - Variables relationnelles pour la représentation de transformations/liens musicaux variables
  - Quelques contraintes sur ces variables
- Nouvelles contraintes et problèmes musicaux plus globaux
- Début des recherches présenté
- Toutes aides ou conseils pour la suite sont les bienvenus