Visualisation musicale d'un CSP

Jérémie Vautard, Arnaud Lallouet

Visualisation musicale d'un CSP

Article Jeune Chercheur JFPC'06

Jérémie Vautard **Arnaud Lallouet**

Université d'Orléans — LIFO
BP6759, F-45067 Orléans
{Jeremie.Vautard|Arnaud.Lallouet}@univ-orleans.fr

Résumé

Cet article décrit plusieurs méthodes d'auralisation afin de réaliser des pièces musicales à partir de la trace d'éxécution d'un CSP au format xml GenTra4CP. Ces méthodes sont essentiellement basées sur des associations entre évènements de trace et évènements musicaux. Plusieurs associations sont proposées ici, ainsi que les phénomènes d'éxécution (parcours d'arbre, arc-consistance, ...) qu'il est possible de reconnaître dans la musique ainsi créée. On dresse enfin la liste des intérêts possible d'une telle auralisation tant pour un compositeur que pour un programmeur de CSP. L'article est accompagné d'exemples sonores aux formats Midi et MP3.

Abstract

This article describes several ways to automatically create music from a GenTra4CP-formatted CSP execution trace. These ways are essentially based on associations between music and trace events. We present some of these ways here, with a description of which particularities of the trace it is possible to hear through the music generated. Finally, the interests, both for a musician and for a programmer, are listed. This article comes with some examples in MIDI and MP3 format.

1 Introduction

L'utilisation de problèmes de résolution de contraintes en musique n'est pas nouvelle. Les éditeurs de logiciels de musique assistée par ordinateur sont actuellement intéréssés par cette méthode de programmation, ainsi que par les méthodes de recherche locale, pour inclure à leurs logiciels de puissants outils d'aide à la composition. Par exemple, Sony CSL Paris est à l'origine de nombreux documents étudiant l'utilisation de la programmation par contraintes en harmonisation automatique[1] [6, 7].

Plus récemment, Charlotte Truchet a présenté en 2004 dans [9] un grand nombre de CSPs modélisant des problèmes posés par des compositeurs contemporains. Il s'agit de proposer une musique solution du problème, ou du moins résolvant le plus de contraintes posées, afin de fournir une matière de base au compositeur. Ces CSPs décrivent un large panel de problèmes relatifs à l'harmonisation et au rythme d'une partition, à la construction d'une mélodie, ou encore à la jouabilité d'une partition par un instrumentiste. Un ordinateur est donc ensuite capable de proposer des partitions répondant le plus possible aux contraintes fixées par un compositeur.

Cependant, cet article propose une approche différente : on utilise la trace d'éxecution de la résolution d'un CSP quelconque pour générer automatiquement de la musique. Cette approche est apparentée à certains logiciels permettant de réaliser automatiquement des pièces musicales. Ces logiciels sont basés sur un générateur de nombres qui seront associés à un évènement musical. Ce générateur est soit pseudo-aléatoire, auquel cas la musique, bien que pouvant être mélodique, n'a en général aucune structure, soit fractal, dont l'intérêt est de retrouver, dans la musique générée, l'autosimilarité qui caractérise généralement les fractales. Les associations entre les évènements musicaux et les nombres produits sont variées, et vont de la simple assignation de note (les nombres produits sont ramenés à des entiers appartenant à un certain intervalle, puis une note, dont la hauteur est fixée par le nombre obtenu, est posée dans la musique), à des algorithmes plus complexes, basés sur des notions avancées

[1] Création de plusieurs portées satisfaisant entre elles des règles d'harmonie définies.

de solfège.

Le projet CAITLIN de Paul Vickers [10], est assez proche de ce que nous faisons ici : il s'agit d'une extention du logiciel Turbo Pascal de Borland qui permet d'ajouter à un programme Pascal les instructions nécessaires pour faire de la musique en rapport avec son éxécution. Ainsi, les boucles et les conditionelles sont chacunes représentées par un son bien précis, identifiable entre tous. Le but de ce projet est d'aider les programmeurs novices a déboguer à l'oreille leurs programmes.

2 Outils et notions utilisées

2.1 Coté contraintes

Un CSP est un ensemble de relations logiques appelées contraintes portant sur un ensemble fini de variables ayant un ensemble fini de valeurs possibles, appelé domaine. Un solveur de contrainte est un programme exhibant les solutions d'un CSP, une solution consistant en une affectation de toutes les variables à des valeurs contenues dans leur domaine initial et satisfaisant toutes les contraintes.

Un format de trace d'éxécution en XML, nommé Generic Trace Format for Constraint Programming ou GenTra4CP [5], a été défini à l'occasion du projet OADymPPaC en Juillet 2004. Il définit avec précision les différents évènements rencontrés lors de la résolution d'un CSP devant être rapportés par un solveur générant cette trace, tout en étant suffisament souple afin de permettre à une grande variété de solveurs de générer une trace correspondant à ce format. GenTra4CP est public et précisément décrit.

Un solveur de contraintes manipule des variables, et maintient leur domaine courant. Chaque fois qu'une variable est créée, la trace l'indique par une balise <`new-variable`> ayant en attributs un identifiant unique de la variable dans la trace (`vident`), le nom donné (`vname`), s'il y a lieu, par l'utilisateur à cette variable, et le nom interne donné à la variable par le solveur (`vinternal`). La balise <`variable`>, ayant les mêmes attributs, sera utilisée chaque fois qu'il sera nécessaire de faire référence à une variable dans la trace. Ces deux balises contiennent une balise <`vardomain`> représentant un domaine de variable sous forme d'une union d'intervalles, éventuellement réduits à une valeur.

L'acquisition de la connaissance d'une nouvelle contrainte par le solveur est signalée dans la trace par une balise <`new-constraint`>, ayant pour attribut l'identifiant unique `cident` et le nom interne au solveur `cinternal` de cette contrainte. Quand une étape de la réduction aboutit à une réduction des domaines suffisament importante pour que toutes les affectations possibles des variables satisfassent une contrainte, une balise <`solved`> apparaît dans la trace, avec l'identifiant de la contrainte concernée comme attribut.

Lors de la résolution du CSP, le solveur réduit le domaine des variables en détectant et retirant des valeurs rendant une contrainte fausse quelles que soient les valeurs prises par les autres variables dans leurs domaines actuels. Une telle réduction se répercute dans la trace par l'apparition d'une balise <`reduce`> ayant en attributs les identifiants de la variable concernée et de la contrainte à l'origine de cette réduction. Cette balise contient aussi l'ensemble des valeurs retirées à la variable ainsi que son nouveau domaine. Il peut arriver qu'une variable se retrouve, suite à une réduction, avec un domaine vide. Cet évènement s'appelle un échec et est enregistré dans la trace par une balise <`failure`> ayant pour argument un identifiant unique de noeud `nident`.

Une fois que le solveur a effectué toutes les réductions possibles, il aboutit, selon que les variables sont toutes instanciées ou non, à une solution ou à un point de choix. L'arrivée à ces états correspond à l'apparition dans la trace, respectivement d'une balise <`choice-point`>, ou d'une balise <`solution`> ayant pour attribut un identifiant unique de noeud `nident`. Ces balises contiennent en outre une description générale de l'état actuel, c'est-à-dire la liste des domaines courants des variables. Cet état actuel est représenté dans une balise <`state`>.

A l'arrivée à un point de choix, une des variables est instanciée à chaque valeur de son domaine courant, créant autant de sous problèmes à traiter.

La découverte d'une solution ou d'un échec est immédiatement suivie d'un retour en arrière, ou backtracking, au point de choix le plus proche : toutes les variables retrouvent le domaine qu'elles avaient lors de ce point de choix, et la variable instanciée a maintenant la valeur suivante. Cet évènement est signalé par une balise <`back-to`>, ayant pour attribut l'identifiant du noeud correspondant au point de choix `node` auquel on revient, ainsi que l'identifiant du noeud `node-before` correspondant à la solution ou à l'échec ayant entrainé de retour. La description de l'état du point de choix sur lequel on est revenu est aussi rappelée.

Le format GenTra4CP permet aussi de décrire le fonctionnement interne d'un solveur en prévoyant par exemple de rapporter des évènements de gestion interne des contraintes. Cependant, ces évènements n'ont pas été retenus dans cet article où l'on se focalise plus sur la structure du CSP lui-même que sur le fonctionnement interne du solveur, supposé parfaitement correct. Ludovic Langevine a créé une extension du logiciel GNU-Prolog [2] appelée Codeine [4], permettant

de produire une trace d'éxécution de ce format lors de la résolution d'un CSP via le solveur de contraintes sur les domaines finis intégré à GNU-Prolog.

2.2 Côté musical

2.2.1 Quelques notions de solfège...

L'objectif de cette partie n'est pas de faire un cours de théorie de la musique, ni même une introduction au solfège, ce qui pourrait prendre un ouvrage tout entier. Il sera donc expliqué ici uniquement les notions utilisées dans cet article, de manière très sommaire et incomplète, mais cependant suffisante pour une bonne compréhension.

La durée des notes est indiquée en termes de ronde, blanche, noire, croche, et double-croche, chaque appelation désiignant une note de durée deux fois moindre que la précédente. Bien que ce ne soit pas une règle absolue en musique, nous considèrerons dans cet article que la noire a une durée de 1 temps.

Une gamme peut être considérée comme composée d'une note fondamentale (appelée Tonique) et de 7 intervalles, définissant un ensemble de 7 notes[2]. On distingue différents types de gammes, différenciés par les intervalles décrivant la gamme. Par exemple, dans une gamme majeure, en partant de la Tonique, les notes sont espacées de 1, 1, 1/2, 1, 1, 1, et 1/2 ton. Dans une gamme mineure par contre, les notes sont espacées de 1, 1/2, 1, 1, 1/2, 1, et 1 ton.

Le mode indique le type de gamme utilisé. On ne parlera ici que des modes mineur et majeur, les plus utilisés, mais il en existe bien plus.

L'armure définit l'ensemble des notes que comportera une portée. Cet ensemble de notes décrit deux gammes : il y a exactement une gamme mineure et une gamme majeure pour une armure donnée, en fonction de la tonique que l'on choisit. L'armure comporte entre 0 et 7 dièses ou bémols. Par exemple, une armure vide correspond aux notes Do, Ré, Mi, Fa, Sol, La, et Si, correspondant aux gammes de Do Majeur et La Mineur, tandis qu'une armure composée d'un bémol n'a plus de Si, mais un Si bémol, ce qui correspond aux gammes de Ré mineur ou Fa majeur.

Un accord désigne trois notes ou plus jouées ensemble. Par abus de langage, on peut aussi parler d'accord pour deux notes jouées ensemble. On rencontre par exemple très souvent des accords composés de la première, troisième, et qinquième note d'une gamme. Ces accords sont appelés accords parfaits et identifient le mode (majeur/mineur) dans lequel on se trouve.

[2]Il existe aussi des gammes de 5 ou 8 notes, mais elles sont moins courantes.

2.2.2 Le format MIDI

Le format MIDI est un flux binaire et nous avons préféré utiliser le format "MIDI assembly code" pouvant être compilé en un fichier MIDI par un logiciel gratuit du nom de Midi File Assembler [3]. Un fichier MIDI contient un flux d'informations au format du même nom. Ce format ne décrit pas directement l'onde sonore à reproduire, comme c'est le cas dans les formats audio tels que le WAV ou le MP3, mais plutôt l'ensemble des évènements musicaux élémentaires composant une partition, comme le début et la fin d'une note, la variation du volume ou encore la variation de fréquence (pitch bend). A la lecture du fichier, l'ordinateur ou le synthétiseur se charge alors d'interpréter ces informations avec une banque[3] d'instruments.

Techniquement, le format MIDI permet d'envoyer des évènements musicaux sur 16 canaux séparés. Les principaux évènements sont :

- *NOTE ON(Hauteur, Volume)* : commence à jouer une note à la hauteur et au volume indiqués. La note ne sera relachée qu'au prochain NOTE ON ou NOTE OFF de même hauteur rencontré.
- *NOTE OFF(Hauteur)* : Relache une note préalablement posée par un NOTE ON.
- *PROGRAM (Instrument)* : Change l'instrument associé à une piste. Tous les prochains NOTE ON rencontrés seront joués par l'instrument dont le numéro est passé en paramètre.

Les paramètres hauteur, volume et instrument sont des entiers compris entre 0 et 127. La hauteur d'une note varie de un demi ton à chaque unité. Le format MIDI permet ainsi naturellement de décrire les musiques dont les notes sont sur l'échelle chromatique[4], c'est-à-dire toutes les musiques occidentales, par opposition par exemple à certaines musiques moyen-orientales dont les notes peuvent varier d'un quart de ton, voire moins.

Par convention, le canal 10 est destiné à la description des percussions. Ainsi le paramètre hauteur des évènement NOTE ON posés sur ce canal désignent non pas une hauteur de note mais un instrument percussif (grosse caisse, cymbale, tom etc...) tandis que PROGRAM sert à désigner le jeu de batterie utilisé (jeu standard, jeu de chambre, jeu puissant etc...)

Là aussi, cette liste est loin d'être exhaustive (il existe aussi des évènements décrivant précisément la variation de hauteur ou de volume d'une note au cours du temps), mais cependant, les évènements décrits ici sont suffisants pour générer plus que l'essentiel d'une

[3]Ensemble d'enregistrements de sons joués par divers instruments et permettant de recréer artificiellement une mélodie jouée par l'un de ces instruments.

[4]Ensemble de notes toutes séparées d'un demi-ton entre elles.

musique.

3 CSP Singer

Le logiciel CSP Singer est une implémentation du concept suivant : un ensemble d'associations entre les évènements de résolution de contraintes et des évènements musicaux permet de générer de la musique à partir de la trace d'exécution de la résolution d'un CSP fournie dans le format GenTra4CP. La musique est produite dans le format Midi Assembly, qu'il faut ensuite convertir en un fichier au format Midi. Son utilisation est fort simple : à l'ouverture, une boite de dialogue demande de selectionner le fichier de trace XML à auraliser. Puis, après une première analyse du document, il informe de la profondeur maximale de l'arbre de recherche et du nombre de balises que le document XML contient et dresse la liste des points de choix et retours en arrière rencontrés, ainsi que le numéro de balise qui leur correspond :

```
(...) Choice point at : 867
(...) BackTracking at : 2508
```

Cette information permet de savoir quelle partie de la trace on transforme en musique en choisissant les numéros de balises sur lesquelles le programme commencera et terminera la génération de la musique. Il demande cette information immédiatement après :

```
Commencer sur quelle balise ? 0
Finir sur quelle balise ? 5000
```

Ceci est très pratique lors de l'auralisation de gros problèmes, dont la trace d'éxécution contient plusieurs millions de balises et dont l'auralisation complète engendrerait une musique pouvant durer plusieurs journées. Il est ensuite demandé si toutes les variables doivent être utilisées, ceci afin soit de se focaliser sur le sort de certaintes d'entre elles dans une optique de compréhention de l'éxécution du CSP, voire de débogage, soit d'éliminer des variables dont le domaine initial est trop petit pour générer quelque chose de musicalement intéressant (une variable dont le domaine reste inclus dans $0, 1$ engendre, selon les cas, la mélodie d'une sirène de pompier, ou du code morse.)

```
Le probleme possede 15 variables.
Toutes les utiliser ? (O/N)n
Variable _#2 Intervenant 9 fois.
valeur min : 0   valeur max : 268435455
Correspond à la variable : (Variable interne)
Garder ? (O/N)o
```

Ici, le code après Variable est la représentation interne de la variable. On trouve ensuite le nombre de fois où elle est rencontrée dans la trace, les valeurs extrêmes de son domaine de définition, et enfin, le nom donné par l'utilisateur de Codeine à cette variable.

Ce choix de variable peut se révéler très intéressant pour relever du premier coup d'oreille le positionnement des évènements les plus importants pour les seules variables sélectionnées. On peut ainsi, par exemple, en ne retenant qu'une variable, retracer immédiatement tout son chemin dans l'arbre de recherche, les évènements de réduction et de description du domaine de variable utilisés pour produire la musique ne correspondant qu'à cette variable là. Le style, qui définira le type de musique générée, est ensuite demandé. Chaque style engendre un type de musique différent, en rapport avec son nom. On définit ensuite l'emplacement et le nom du fichier MIDI Assembly généré. Enfin, s'il y a lieu, quelques réglages complémentaires, dépendantes du style choisi, sont demandées à l'utilisateur, et le fichier est enfin généré.

Ce logiciel a à l'heure actuelle été utilisé uniquement avec des traces générées par Codeine, cependant le choix des évènements utilisés pour générer la musique réduident l'importance du solveur utilisé. Cependant, la forme générale de l'arbre de recherche est aussi donné par la stratégie de recherche employée, et il est donc probable que la résolution d'un même CSP par deux solveurs utilisant des stratégies de recherche différentes engrendre deux musiques totalement différentes.

Voici une description complète des associations entre évènements musicaux et évènements de trace de chaque style que contient actuellement le programme :

Style "Philip Glass". Historiquement, ce style découle directement du tout premier essai d'associations réalisé, enrichi et rendu jouable[5]. Il a été nommé ainsi après coup, en raison des similitudes avec certaines musiques de ce compositeur contemporain.

Voici la liste des associations réalisées :

- Le tempo est fonction de la profondeur dans l'arbre de recherche (plus on est profond, plus le tempo est élevé) ;
- Chaque backtracking provoque un changement du mode (mineur/majeur) ;
- La description d'un état (techniquement, la lecture d'une balise <`state`>) modifie l'armure, sans changer le mode ;
- Chaque description d'un domaine de variable entraine une montée au piano durant exactement un temps. Chaque note correspond à une valeur possible de la variable. Si le domaine est trop étendu, cette montée est remplacée par un accord

[5] La partition engendrée pourrait être physiquement jouée par des interprètes humains.

de quinte [6] ;
- La découverte d'une solution est marquée par un coup de cymbale crash ;
- La découverte d'un échec est marquée par un coup de caisse claire ;
- Chaque nouvelle contrainte provoque un accord de clarinette qui prend fin au changement de gamme ou de tempo suivant.

Ce style demande de choisir un tempo de base de la musique, exprimé en pulsations par minute, ainsi qu'un modificateur de tempo qui, multiplié par la profondeur dans l'arbre de recherche et additionné au tempo de base, donnera le tempo réel de la mélodie. Une valeur raisonable pour le tempo de base se situe entre 60 et 100, tandis que le modificateur dépend de la profondeur maximale de l'arbre de recherche, et devrait être réglé de sorte à ce que le tempo réel de dépasse pas 150, sans quoi la mélodie serait injouable par un instrumentiste.

De par ses origines de test, ce style génère des musiques sonnant juste mais très épurées. Il est cependant possible d'en extraire quelques informations, comme la structure générale de l'arbre de recherche (en prètant attention aux variations de tempo) ou la taille des domaines de variables (rapidité des montées au piano). Il serait cependant très difficile d'identifier précisément chaque valeur d'un domaine de variable : en effet, comme le souligne également Paul Vickers dans [10], l'identification rapide et exacte de la hauteur des notes est beaucoup moins naturelle pour un non-musicuen que d'autres évènements, par exemple le rythme. Les accords à la clarinette permettent aussi de savoir à quel endroit une contrainte vient d'être définie.

Style "Hard Rock". Ce style est le premier réellement bâti dans l'optique d'obtenir un morceau de musique, joué par un orchestre rock composé d'un batteur, d'un guitariste lead, d'un guitarise rythmique et d'un bassiste. On obtient donc un morceau très typé, où les évènements de trace jouent des rôles plus discrets dans la partition. La batterie est créée à partir de motifs prédéfinis.

Voici la liste des associations réalisées :
- La profondeur dans l'arbre de recherche détermine le tempo et le motif joué par le batteur. Plus on est profond dans l'arbre de recherche, plus celui-ci est violent.
- La rencontre d'un noeud dans l'arbre de recherche (point de choix ou backtracking) entraine un changement de gamme. La nouvelle gamme est calculée en fonction de l'identifiant du noeud rencontré. Généralement, l'identifiant de noeud est incrémenté à chaque noeud créé, ce qui fait que le retour à un noeud déjà existant entraîne un changement radical de gamme, tandis que la découverte d'un autre noeud se contente de transposer d'un demi ton la gamme utilisée modulo l'octave.
- les descriptions des domaines de variables sont responsables de la mélodie jouée par la guitare lead. Chaque variable est décrite sur exactement un temps et la hauteur des notes jouées correspond à une borne d'un intervalle constituant le domaine de la variable en question. Il y a donc plus de notes jouées sur un temps si les valeurs du domaine forment plusieurs intervalles.
- La réduction du domaine d'une variable entraîne une descente de double-croches de la guitare basse. La note de départ de cette descente est fixée par le nombre de valeurs retirées au domaine : plus ce nombre est grand, plus la note de départ est haute.
- La guitare rythmique est calculée en fonction de la hauteur des notes jouées par la guitare lead et d'une autre valeur incrémentée à chaque calcul.

Ce style demande aussi de choisir un tempo de base, et un modificateur de tempo, selon le même principe que le style précédent. Le tempo de base devrait être compris entre 150 et 170. Etant donné que la batterie joue déjà quelque chose de différent en fonction de la profondeur dans l'arbre de recherche, il n'est pas nécessaire de fournir un grand modificateur de tempo (on peut même le mettre à 0). Dans tous les cas, le tempo final ne devrait pas dépasser 220 pour rester écoutable.

Bien que ce style n'y soit pas destiné, on retrouve une certaine quantité d'informations en écoutant les musiques produites. La plus flagrante est sans doute de savoir si la consistance d'arc a été utilisée pour résoudre le problème. En effet, pour toute résolution où seule la consistance de borne est appliquée, tous les domaines seront décrits par un seul intervalle et donc, la guitare lead ne jouera que des croches. Par contre, là où la consistance d'arc intervient, la guitare pourra jouer des notes plus rapides. On retrouve aussi, via la modification du tempo et de la batterie, la forme générale de l'arbre de recherche. Les moments où une réduction est effectuée sont aussi identifiés grâce à la basse.

Par exemple, concentrons-nous sur le second morceau fourni en exemple (`2 - Hard rock - 4 queens.ogg`). Il débute relativement calmement, mais son tempo ainsi que la violence du motif de batterie augmente tous les quatre temps. Cela permet de deviner que l'on s'enfonce dès le début très profondément

[6] Intervalle de sept demi tons entre deux notes. Le terme "Accord" est ici utilisé de manière abusive, car seules deux notes sont jouées en même temps.

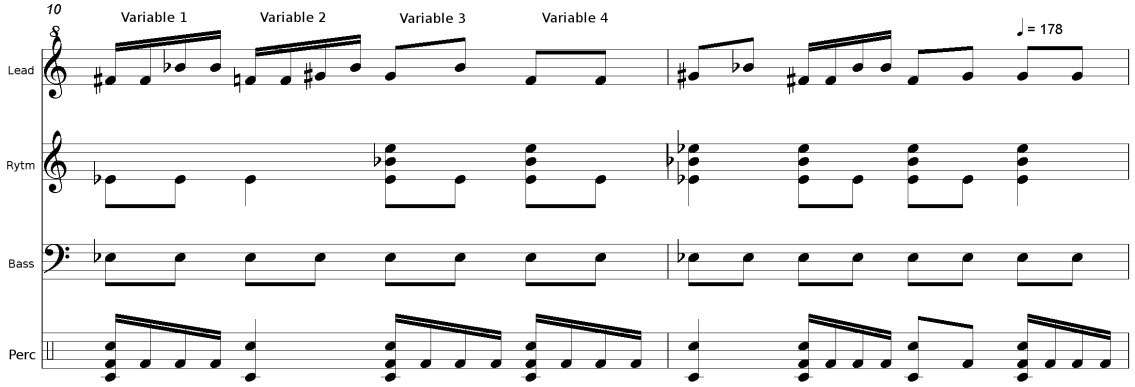

FIG. 1 – Dans cet extrait de partition d'une musique dans le style Hard Rock, la guitare lead décrit à chaque temps une variable. On peut ici voir que les variables 1 et 2 ont leur domaine découpé en deux intervalles disjoints, alors que les variables 3 et 4 possèdent un domaine réduit à un unique intervalle.

dans l'arbre de recherche (et effectivement : cet arbre de recherche possède un tronc de 5 noeuds). L'état actuel des variables étant décrit à chaque noeud, l'écart de quatre temps entre chaque augmentation de tempo permet de dire que le problème se compose de quatre variables, et qu'aucune réduction n'est effectuée. Cette dernière affirmation est renforcée par le fait que l'écart entre deux croches jouées par la guitare lead est toujours d'une quarte, indiquant que le domaine de chaque variable est toujours constitué de 4 valeurs consécutives. Au bout de 12 secondes, la mélodie change radicalement. Le fait d'entendre deux notes consécutives de hauteur égale à la guitare permet de dire que certaines variables ont été assignées, et le fait d'entendre certains passages en double-croche permet de dire que le domaine de certaines variables a été divisé en deux intervalles disjoints, et donc que l'arc-consistance est utilisée dans la résolution de ce CSP. Durant toute cette partie du morceau, l'alternance des trois motifs de batterie les plus violents permet d'extrapoler que les branches du sous-arbre de l'arbre de recherche que l'on est en train d'explorer sont courtes. Enfin, la musique revient graduellement à un rythme plus calme, au fur et à mesure des backtracking finaux où l'on remonte vers la racine de l'arbre.

Style "Debug". L'idée de ce style est venue à la lecture de la thèse de Paul Vickers [10] traitant de l'auralisation de programmes Pascal. Il procède à des associations simples de sons bien identifiables, appelés *earcons*, à chaque évènement d'éxécution. Ce style décrit la trace d'éxécution de manière analogue, en associant à chaque évènement une série bien particulière de notes.

Voici la liste de ces associations :

- La définition d'une nouvelle variable (balise <`new-variable`>) produit un arpège majeur montant au glockenspiel ;
- la définition d'une nouvelle contrainte (<`new-constraint`>) produit un arpège descendant mineur au basson ;
- la résolution d'une contrainte est décrite par un arpège majeur montant aux cuivres ;
- la découverte d'une solution est décrite par une suite d'accords de quarte aux cuivres (son de "victoire") ;
- la découverte d'un échec est décrite par une mélodie descendante aux cuivres(son de "catastrophe") ;
- la réduction du domaine d'une variable est indiquée par une descente de gamme très rapide au basson ;
- Une onde sinusoidale, dont la fréquence est déterminée par la profondeur dans l'arbre de recherche, est jouée pendant tout le morceau ;
- A chaque description complète d'une variable, un piano joue une double croche dont la hauteur est définie par la taille de son domaine.

Le but de ce style n'était certes pas de produire de la musique agréable, mais plutot d'essayer de donner le plus d'informations possibles au sujet de la trace. Ce but n'est ici que partiellement atteint : en effet, l'auralisation de tous les évènements d'exécution par un son précis se prête mieux aux langages impératifs, dont l'exécution a une forme beaucoup plus linéraire. Outre les informations directement données par les évènements musicaux (nouvelle variable, nouvelle contrainte, résolution d'une contrainte, réduction, solution et échec), on a aussi une information sur l'évolution de la taille des domaines des variables : plus les notes jouées au piano sont basses, plus le

domaine des variables est réduit. Ainsi, il est possible d'entendre où se sont effectuées les réductions les plus importantes.

Style "New Age". Ce style, basé au départ sur le style Philip Glass, a subi d'importantes modifications et les musiques qu'il génère sont calmes, ressemblant au style d'où il tire son nom. On y trouve deux lignes mélodiques, à la harpe et aux choeurs, et un accompagnement aux violons. Là encore, c'est un style musical avant tout.

Les associations effectuées sont toutefois assez proches de celles du style Philip Glass. En voici la liste :

- Le tempo est fonction de la profondeur dans l'arbre de recherche ;
- Chaque description d'un domaine de variable entraine une montée à la harpe durant exactement un temps. Chaque note correspond à une valeur possible de la variable. Si le domaine est trop étendu, cette montée est remplacée par un accord de quinte, tout comme dans le style Philip Glass. Les restrictions dans ce style sont cependant moindres, la harpe permettant un enchainement de notes bien plus rapide ;
- Chaque description de valeurs enlevées d'un domaine de variable entraîne une ou plusieurs notes de choeurs ;
- Les accords de violons sont aussi dus aux évènements de nouvelle contrainte, de la même manière que la clarinette dans le style Philip Glass ;
- La découverte d'une solution est représentée par une quinte jouée au "Brillance"[7].

Les informations que l'on peut tirer de ce style sont à peu de choses près celles qu'on peut tirer du style Philip Glass, plus la localisation des réductions de domaines, indiquée par les choeurs. De par la nature calme des musiques générées, le tempo ne devrait pas être fixé sur une valeur trop élevée, ni même subir des modifications trop brusques. Un tempo de base entre 80 et 120 et un modificateur ne dépassant pas 5 donnent en général de bons résultats.

Style "Apocalyptica". Ce style tire son nom du quatuor de violoncellistes finlandais. Il a été créé à la base afin d'essayer de mettre en relief une information qui a été suggéré par Charlotte Truchet [8], à savoir la taille générale des domaines de variable. Dans ces musiques, un violoncelle joue dans les graves une portée rythmique, tandis que deux autres violoncelles se partagent les lignes mélodiques. La première ligne mélodique joue dans un domaine d'une octave supérieur à celui de la seconde. Le rythme joué par le premier violoncelle est constitué de notes bien plus graves que les mélodies et devient plus "énervé" au fur et à mesure que les domaines de variables sont réduits.

Les associations effectuées sont les suivantes :

- Le tempo est toujours fonction de la profondeur dans l'arbre de recherche ;
- Le rythme joué par le violoncelle rythmique est fonction de la moyenne des tailles des domaines des variables ;
- Une nouvelle variable se traduit par une trille [8] de double-croches sur la seconde portée mélodique ;
- Une nouvelle contrainte est décrite par un arpège descendant sur la seconde portée mélodique ;
- La résolution d'une contrainte se traduit par un arpège montant sur la seconde portée mélodique ;
- Chaque variable est décrite par la première ligne mélodique sur exactement un temps et la hauteur des notes jouées correspond à une borne d'un intervalle constituant le domaine de la variable en question. Il y a donc plus de notes jouées sur un temps si les valeurs du domaine forment plusieurs intervalles, tout comme dans le style hard-rock ;
- Lors d'une réduction, les différents intervalles enlevés au domaine d'une variable sont décrits par la portée mélodique grave. On entends alors deux croches dont les hauteurs sont fonction des bornes de cet intervalle.

L'information qui saute à l'oreille est, et c'était le but de de style, l'évolution de la taille moyenne des domaines des variables, grâce au rythme imprimé par le violoncelle rythmique. Les trilles et arpèges provoqués par les évènements `<new-variable>`, `<new-constraint>` et `<solved>` sont repérables lorsque seul le second violoncelle mélodique joue, mais deviennent plus difficiles à trouver lorsque les deux lignes mélodiques se trouvent mélées.

Selon le nombre de variables que comprend le CSP, les deux mélodies seront plus ou moins dissociées. Dans tous les cas on peut déterminer que la musique est en train de décrire un état général (`<state>`) quand la première ligne mélodique joue sans discontinuer, tandis que la seconde reste silencieuse. Les endroits où s'opèrent des réductions donnent par contre des passages où les deux mélodies jouent en même temps (la première décrit les domaines de variables tandis que la seconde décrit les valeurs retirées).

Prise à part, la seconde ligne mélodique est composée de motifs répétés (les arpèges et les trilles) entrecoupées de séquences plus aléatoires (les valeurs retirées lors d'une réduction). Ceci montre la possibilité de construire des mélodies plus riches, en se basant sur

[7] Instrument totalement synthétique présent dans la base d'instruments MIDI.

[8] Alternance répétée et rapide de deux notes.

des petites séquences prédéfinies, plutôt que des notes uniques comme nous l'avons fait jusqu'à présent.

Il faut noter que les musiques générées par ce style ont tendance à ne pas jouir d'une égalisation très appropriée sur beaucoup de synthétiseurs MIDI (le violoncelle rythmique a un volume souvent trop faible).

4 Observations

4.1 Comment créer un nouveau style ?

Il est nécessaire, pour créer les différents styles, de prendre en compte le nombre et la disposition des évènements présents dans une trace d'éxécution lors de l'association avec les évènements musicaux. En effet, même s'il est virtuellement possible de procéder à n'importe quelles associations, on peut d'ores et déjà être assuré d'obtenir une musique "crédible" en associant aux évènement de trace des évènements musicaux ayant, dans un morceau de musique, sensiblement la même disposition. Pour cela, il faut étudier cette disposition des évènements dans une trace d'exécution.

Il faut aussi prendre en compte la quantité d'informations données par chaque évènement : par exemple, un évènement revenant très régulièrement pourrait servir pour construire la mélodie d'une musique, qui est constituée d'un flot régulier de notes. Cependant, si cet évènement n'a aucun attribut, il n'est pas en mesure de produire d'autres informations essentielles à la construction d'une mélodie, comme la hauteur d'une note. Cette association serait donc, à priori, peu appropriée.

On considère par la suite que chaque attribut peut engendrer un nombre entier, suceptible d'être utilisé comme paramètre d'évènement musical.

Description des domaines des variables. Un domaine de variable est décrit par des blocs <`range`> inclus dans un bloc <`vardomain`>, lui même inclus dans un bloc <`variable`> ou <`reduce`>. On a donc, pour chaque domaine de variable, au moins 6 entiers à disposition. Ces entiers ont des relations fortes entre eux : on a la valeur minimum et maximum du domaine de la variable, définissant un intervalle dans lequel sont contenu la plupart des autres entiers disponibles (valeurs du domaine), à l'exception de l'entier déduit de l'identifiant de la variable.

Les descriptions des domaines de variables sont de loin les évènements les plus récurrents dans la trace. De plus, la grande quantité d'informations disponible à chaque description en fait un évènement tout trouvé pour une ligne mélodique. La génération d'une mélodie peut être effectuée de plusieurs manières. (les différents styles programmés dans CSP Singer en utilisent deux)

Ces évènements sont souvent agglomérés dans une description de l'état général (dans un bloc <`state`>), où chaque variable définie est décrite. Cependant, on trouve aussi ces évènements dans la définition d'une variable (<`new-variable`>) et dans une réduction de domaine(<`reduce`>). De manière imagée, on a donc en quelque sorte plusieurs galaxies de ces évènements, plus quelques astres isolés.

Réductions. La réduction d'un domaine de variable est décrite dans un bloc <`reduce`>. Ce bloc contient la liste des valeurs retirées au domaine, puis une description du domaine réduit. On a donc au moins quatre entiers à disposition (identifiant de la variable, identifiant de la contrainte, et bornes du domaine), plus les bornes des différents intervalles décrivant le domaine et les valeurs retirées. Mis à part les identifiants, ces entiers ont là aussi beaucoup de relations entre eux : le domaine et l'ensemble des valeurs retirées sont disjoints, et toutes ces valeurs appartiennent à l'ancien domaine (avant réduction).

Les réductions se produisent généralement en petit nombre, voire pas du tout, en début de trace, puis elles deviennent plus nombreuses au fur et à mesure que l'on s'enfonce dans l'arbre de recherche, et ce jusqu'à la fin de la trace. Elles sont donc suffisament nombreuses pour générer un accompagnement, qui a besoin d'un flot continu d'informations pour ne pas être trop répétitif, mais cependant en moindre quantité que pour une mélodie.

Il peut être possible de générer une mélodie à part en utilisant les descriptions de domaines et de valeurs retirées. Cependant, cette ligne mélodique risque d'être silencieuse par endroits (notamment pendant une description de l'état général) si tous les autres domaines de variables sont utilisés pour générer d'autres évènements musicaux durant dans le temps.

Points de choix et backtracking. Les évènements de point de choix et de backtracking sont les seuls évènements modifiant la profondeur dans l'arbre de recherche. Ils sont décrits respectivement dans les blocs <`choice-point`> et <`back-to`> et contiennent chacun une description de l'état général (<`state`>). On peut donc les voir soit comme des évènements instantanés et précis (en considérant la description de l'état à part), soit comme des gros évènements constituant la plupart de la trace. Ces évènements laissent, mis a part la description de l'état général, deux ou trois entiers (identifiant du noeud, identifiant du noeud précédent pour le backtracking, et nouvelle valeur de la profondeur dans l'arbre de recherche).

Considérés sous leur forme ponctuelle (c'est-à-dire en traitant le bloc <`state`> à part, ces évènements sont en petit nombre et équitablement répartis dans

la trace, ce qui les rend idéaux à associer avec des évènements musicaux affectant l'ensemble de la partition, comme le changement de tempo ou d'armure.

Déclarations de nouvelles variables et contraintes. Ces derniers évènements ne possèdent pas beaucoup d'information sur eux : seulement l'identifiant de la variable ou de la contrainte déclarée. Un bloc <`new-variable`> contient aussi une description de son domaine initial, qui se compose d'un seul et unique intervalle. Etant donné que chaque identifiant est unique, ces évènements sont différents deux à deux.

La disposition et le nombre de ces évènements peut varier énormément d'un problème à un autre. On peut cependant observer certaines similitudes : d'une part, la plupart des variables, dont toutes les variables utilisateur, sont définies au tout début du problème. Puis, des variables internes peuvent être déclarées sur pratiquement toute la trace. Les déclarations de nouvelles contraintes suivent le même schéma : on trouve un grand nombre de ces déclarations en début de trace, notamment si on définit un `fd_all_different`[9] sur une liste de variables de grande taille, puis, en nombre relativement plus réduit, sur l'ensemble de la trace.

On peut donc aussi tirer de ces évènements une mélodie, mais celle-ci sera très irrégulière, et susceptible de "couper" à certains endroits. Il est possible de les utiliser pour modifier la nature d'un accompagnement : deux changements d'accompagnement coup sur coup, sans production de notes entre temps, sera perçu comme un seul dans la musique générée.

4.2 Intérêts apportés

Pour un musicien, les différents styles créés jusqu'à présent montrent qu'en utilisant différentes associations et manières de traiter les informations contenues dans les évènements de trace, il est possible de créer une grande variétés de musiques. Il est possible, en utilisant d'autres formalisations de la musique que l'harmonie classique exclusivement utilisée ici (en particulier les formalisations de la musique jazz, pratiquement impossible à décrire avec l'harmonie classique), de définir d'autres évènements musicaux, étandant ainsi grandement le domaine des évènements musicaux disponibles, et par là même les possibilités d'associations avec des évènements de trace.

De plus, la construction arborescente de la trace d'éxécution peut être très bien reproduite par la musique, ce qui donne des morceaux dont la structure est peu commune, mais néanmoins agréable. On peut ajouter aussi que les domaines de variables étant modifiés "petit à petit", et décrits entièrement de manière régulière, les informations données par les évènements correspondants sont répétitives, mais malgré tout légèrement modifiées à chaque passage, ce qui peut donner une mélodie progressive.

Enfin, il existe autant de traces d'exécution que de CSP, et ces traces sont très différentes d'un CSP à un autre, ne serait-ce qu'au niveau des domaines des variables et de la structure de l'arbre de recherche. Dans la plupart des styles programmés, la structure de l'arbre de recherche déterminera la structure du morceau créé, de part son influence sur le tempo et, plus indirectement, sur la mélodie, étant donné que les domaines des variables se réduisent au fur et à mesure que l'on s'enfonce dans l'arbre de crecherche. La matière pour générer de la musique par ce biais est donc inépuisable.

Cette matière musicale peut, selon les cas, constituer un morceau à part entière (l'exemple fourni pour le style hard-rock possède, de par la structure très spéciale du CSP des quatre reines, une sorte d'introduction et de coda déjà crées, et il ne lui manque que quelques notes à la fin pour en faire un morceau crédible), ou bien servir de base ou prendre part à une composition réalisée manuellement par la suite.

Pour un programmeur de CSP, une des questions posées au départ était la possibilité d'obtenir des informations tangibles sur l'éxécution d'un CSP en écoutant la musique produite par la trace correspondante. Une réponse à cette question aurait pu être obtenue en faisant utiliser CSP Singer à un nombre relativement grand de programmeurs utilisant le solveur de contraintes de GNU-Prolog, sur une période suffisament grande pour prendre en main le logiciel et assimiler les différents types de musiques générées, ce qui bien sûr a été impossible à faire faute de temps et de moyens.

Cependant, l'expérimentation de CSP Singer au fur et à mesure de sa conception permet d'indiquer les informations perceptibles naturellement à l'oreille :

- La structure de l'arbre de recherche ressort très bien dans toutes les musiques générées, ce qui permet d'entendre à quel point de l'arbre de recherche on se trouve.
- On peut avoir un ordre de grandeur des bornes des domaines de variable en écoutant la mélodie, et donc d'entendre directement où se situent les réductions de domaines les plus radicales. L'application de la consistance d'arc ressort aussi très bien avec certains styles (notamment le "hard-rock").

[9] Prédicat prenant en paramètre une liste de variable et indiquant que toutes ces variables sont différentes deux à deux. Dans GNU Prolog, ceci est déclaré comme autant de contraintes binaires d'inégalité.

– Le style debug permet, de par la nature unique de chaque earcon défini, de retrouver assez précisément où a lieu tel ou tel évènement, et d'observer localement la structure de la trace. Les autres styles permettent aussi, selon les cas, de situer les réductions, nouvelles variables et nouvelles contraintes avec précision.

Cette liste n'est bien sûr pas exhaustive. L'acquisition des reflexes auditifs permettant d'identifier rapidement certains détails de plus en plus précis sur la résolution d'un CSP en écoutant la musique générée à partir de sa trace requiert une pratique régulière de l'outil CSP Singer. Seule une étude complète, sur une longue durée et portant sur un grand nombre de programmeurs permettrait donc d'identifier clairement les limites de l'écoute de traces dans le domaine du débogage. Une telle étude permettrait aussi d'identifier avec précision les informations importantes pour deboguer un CSP, et de se focaliser sur celles qui sont les moins visibles avec un débogueur purement visuel comme il en existe actuellement (par exemple Pavot [1], qui utilise le même format de trace que CSP Singer).

5 Conclusion

La génération de musique à partir de traces d'éxécutions de CSP s'avère très intéressante d'un point de vue musical. Nous n'avons vu ici qu'une infime partie des possibilités qu'ouvre ce nouveau moyen de produire des morceaux de musique. CSP Singer peut encore grandement être amélioré, notamment en formalisant d'autres aspects de l'harmonie classique (notamment la définition de gammes de gammes constituées d'un nombre de notes différent de 7), et en proposant d'autres formalisations de la musique (comme les différentes formalisations du jazz, ou celles proposées par des compositeurs contemporains).

Un autre point d'amélioration de CSP Singer à étudier porte sur la formalisation des associations entre évènements de trace et évènements musicaux. Cela permettrait à l'utilisateur de définir entièrement ses différents styles, et ainsi d'étendre les possibilités du logiciel à l'imagination de l'utilisateur. Cela permettrait aussi à l'utilisateur désireux d'utiliser le logiciel à des fins de déboguage de choisir bien plus précisément les évènements qu'il veut auraliser.

La quantité d'informations que l'on peut déduire en écoutant la musique produite par une trace d'éxécution est elle aussi très encourageante. Comme il a été dit plus haut, il est nécessaire d'étudier, sur une longue durée, l'intérêt d'une utilisation de CSP Singer à des fins de débogage. Une extention envisageable de ce logiciel serait de l'associer à un débogueur visuel, ce qui permettrait d'obtenir sur l'éxécution du CSP des informations visuelles et auditives en même temps.

Remerciements

Nous tenons à remercier Yann Lefevre pour son aide en matière de théorie de la musique et la composition des motifs de batterie utilisés dans le style Hard-rock, ainsi que Charlotte Truchet pour ses idées d'associations entre évènements de trace et musicaux.

Références

[1] Guillaume Arnaud. *Polymorphic Application for Viewer Of Traces.* http ://contraintes.inria.fr/˜ arnaud/pavot/.

[2] Daniel Diaz and Philippe Codognet. Design and implementation of the Gnu-Prolog system. *Journal of Functional and Logic Programming*, 6, 2001.

[3] Jeff Glatt. *Midi Disassembler/Assembler.* http ://www.borg.com/˜ jglatt/.

[4] Ludovic Langevine, Mireille Ducassé, and Pierre Deransart. A propagation tracer for gnu-prolog : from formal definition to efficient implementation. In Catuscia Palamidessi, editor, *Proceedings of ICLP'03, International Conference on Logic Programming*, Lecture Notes in Computer Science, Mumbai, India, December 2003. Springer-Verlag.

[5] The OADymPPaC RNTL Project. *Generic Trace for Constraint Programming*, 2004. http ://contraintes.inria.fr/OADymPPaC/Public /Trace/index.html.

[6] F. Pachet and P. Roy. Musical harmonization with constraints : A survey. *Constraints*, 6(1) :7–19, 2001.

[7] F. Pachet and P. Roy. Harmonisation automatique et programmation par contraintes. In *Du Signal au Signe Musical*, chapter 10. Hermes, 2004.

[8] Charlotte truchet. *Communication personnelle*, 2004.

[9] Charlotte Truchet. *Contraintes, recherche locale et composition assistée par ordinateur.* PhD thesis, Universite Paris 7 - Denis Diderot, 2004.

[10] Paul Vickers. *CAITLIN : Implementation of a Musical Program Auralisation System to Study the Effects on Debugging Tasks as Parformed by Novice Pascal Programmers.* PhD thesis, Loughborough University, 1999.