Du temps écrit au temps produit en informatique musicale

Jean-Louis Giavitto
CNRS & IRCAM – UPMC – INRIA/MuTant

1 Introduction

Plusieurs analogies peuvent s'établir entre la notion de partition et de programme (cf. Table 1). Ces objets correspondent tous deux à la spécification hors temps d'un processus qui se déroulera plus tard. Ici « hors temps » veut dire que le temps de la composition de la partition ou de l'écriture du programme n'est pas celui de l'interprétation de la musique composée ou de l'exécution du programme. Le compositeur, ou le programmeur, définit à l'avance un enchainement d'événements qui se dévoile peu à peu lors de la réalisation dans le temps de l'objet défini. Et ces événements peuvent souvent se décomposer en événements plus simples.

Que ce soit pour la musique ou pour le calcul, les événements à produire sont spécifiés (écrits) dans un langage permettant d'exprimer leurs enchaînements temporels à partir de trois relations temporelles fondamentales : la succession, la simultanéité et la durée. Dans les deux cas, ce langage introduit une distinction entre le sens et la dénotation de ce qui est défini.

Partition	Programme
phase de composition	phase de spécification
phase d'interprétation	phase d'exécution
une écriture / sa dénotation	une expression / son évaluation
un compositeur	un programmeur
un (des) interprète(s)	une (des) machine(s)
…	…

Figure 1. Analogie entre partition et programme informatique

Cependant, malgré ces convergences, le statut d'une partition et celui d'un programme diffèrent. Si partition et programme sont tous deux écrits par un humain pour être ensuite interprété[1], la partition musicale est interprétée par des humains et le programme par des machines. En conséquence, les relations attendues entre la musique écrite et ses interprétations divergent de celles voulues entre un programme et son exécution. Guerino Mazzola parle de la partition comme d'un geste surgelé qu'il faut dégeler lors de l'interprétation. Mais ce réchauffement ne peut se comparer à l'exécution d'un programme informatique : c'est un processus ouvert qui demande une création de l'interprète qui échappe aux spécifications de la partition afin, comme l'exprime Theodor Adorno, de passer d'un état solide à l'état liquide de la musique. Cet espace de liberté et de création, pose des problèmes particulièrement aigus à l'informatique musicale dans le cadre de la *musique mixte*, où doivent coopérer instrumentistes humains et processus informatiques.

[1] En informatique, l'interprète d'un langage de programmation L est un programme ayant pour tâche d'analyser, de traduire et d'exécuter un programme écrit en L.

Si la partition n'est pas un programme, rapprocher les deux notions est cependant potentiellement fertile : on parle par exemple de partition algorithmique en musique et d'orchestration de services en informatique. Confronter les problématiques de la partition à celles des programmes n'a, pour l'informaticien ici, de visés ni normatives ni démonstratives, uniquement heuristiques. La musique occidentale a développé depuis longtemps une écriture permettant de définir finement des relations temporelles impliquant date, durée et séquencement entre des objets musicaux. Les concepts développés pour cette écriture du temps suggèrent de nouveaux outils adaptés à l'expression des scénarios temporels que doit gérer une application informatique en interaction avec des humains.

Dans la suite de ce chapitre, nous esquisserons des parallélismes possibles entre partition et programme du point de vue de l'organisation de structures temporelles, en passant en revue les constituants du temps et leurs relations, le style des expressions temporelles, les formes du temps et enfin sa pluralité. La réalisation de ces structures temporelles ne se fait pas dans le vide. Elle nécessite la coordination entre plusieurs agents qui, dans le cas de la musique mixte, mélangent humains et machines. Nous verrons alors comment la notion d'interaction musicale et d'anticipation permet de développer une réelle coordination lors de l'interprétation entre le musicien et l'ordinateur.

2 Définir des relations temporelles

Traditionnellement le temps est constitué d'instants, de moments, qui sont ses objets élémentaires, ses "atomes". Les instants du temps sont structurés par trois relations :

- la *succession* (avant ou après),
- la *simultanéité* (en même temps),
- la *durée* (depuis/jusqu'à).

Chacune de ces relations s'oppose aux deux autres. Ainsi, ce qui survient ni avant et ni après, et qui ne dure pas, doit se produire en même temps : la simultanéité est donc la négation à la fois de la succession et de la durée. Par ailleurs, la succession s'oppose à la durée, puisque « tant que ça dure, on n'est pas passé à autre chose » ; *etc.* La figure 2 résume ces relations et leurs oppositions.

Cette présentation du temps va nous permettre de classer les différents modèles du temps qui apparaissent dans les langages de programmation[2].

Alan Turing est un des premiers à proposer, en 1936, une formalisation de la notion de calcul. Le temps est présent parce que les opérations élémentaires du calcul s'enchaînent séquentiellement et parce qu'un calcul doit se terminer. Dans les mêmes années, Alonzo Church propose une autre définition du calcul (qui sera prouvée par Turing équivalente à la sienne) fondée sur un modèle très abstrait mais très contraint de la notion de fonction (le λ-calcul). Là aussi le temps apparaît comme une dépendance entre des calculs élémentaires : pour calculer (1+2)*3, il faut d'abord calculer (1+2). Dans ces deux modèles de ce qu'est un calcul, les relations temporelles n'apparaissent qu'à travers la notion de succession : les instants sont logiques, leur durée n'a pas d'importance et la succession des instants est fortement reliée à la notion de causalité et de déduction logique.

L'idée de succession est rendue par la notion de relation d'ordre. Dans un modèle de calcul séquentiel, la relation de succession entre les actions élémentaires est un *ordre total* : on sait toujours si une action se produit avant ou après une autre. C'est le cas dans les langages de programmation séquentielle, comme le langage `C` ou `Java`. Les structures de contrôle (les itérations, la condition, l'appel de fonction, etc.) servent à définir de manière concise une relation d'ordre total entre les très nombreuses instructions élémentaires d'un programme.

[2] voir aussi le chapitre de Gérard Berry dans ce livre

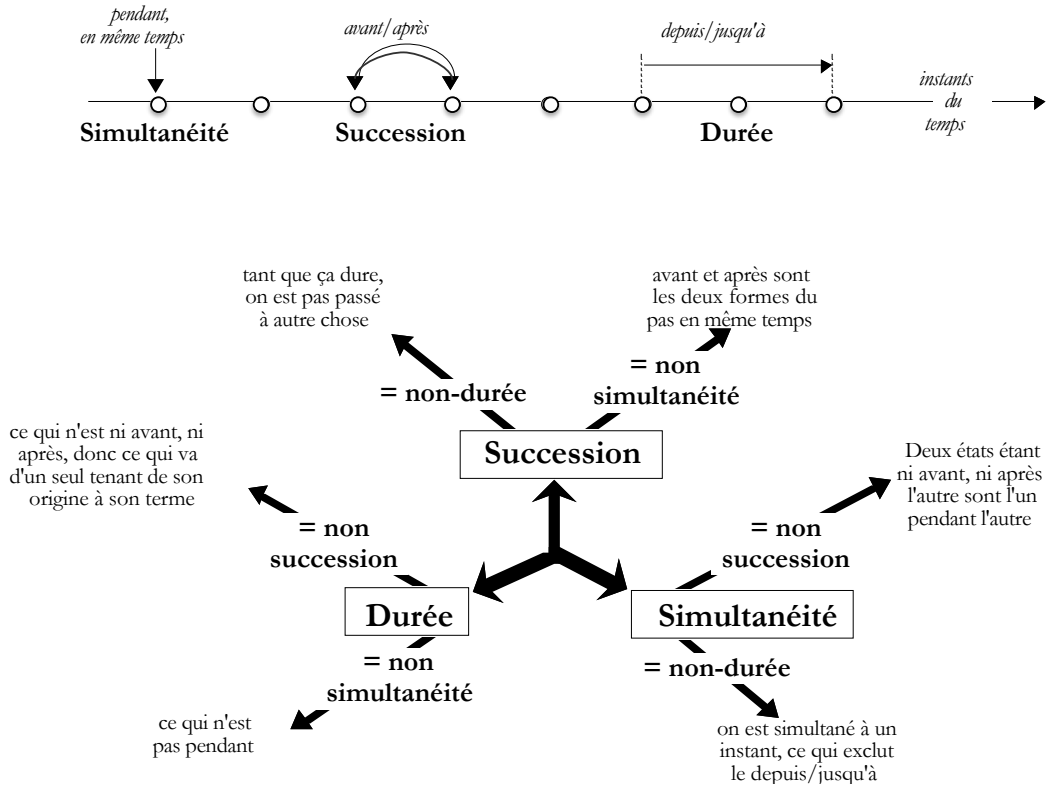

Figure 2. Les trois concepts qui traditionnellement définissent le temps : succession durée et simultanéité, et leurs oppositions.

On a cependant vite eu besoin de définir des successions *partielles* : le calcul de l'expression (1+2) et celui de (3+4) doit par exemple précéder le calcul de la multiplication dans (1+2)*(3+4), mais ces deux sous-expressions peuvent se calculer dans n'importe quel ordre. Dans le λ-calcul cet ordre partiel est défini implicitement par ces relations de dépendances. Cependant, sa mise en œuvre sur un calculateur séquentiel rajoute des contraintes pour obtenir un ordre total : on parle de stratégie d'évaluation.

Les *algèbres de processus* augmentent les langages précédents en introduisant des constructions permettant de définir *explicitement* des *ordres partiels*. On peut ainsi exprimer quelque chose comme :

```
(a | b) ; c
```

pour indiquer que les instructions `a` et `b` sont effectuées *concurremment*, c'est-à-dire dans un ordre indéterminé (opérateur « | »), mais doivent précéder l'exécution de l'instruction `c` (opérateur « `;` »).

Si deux événements sont dans un ordre indéterminé, cela laisse la liberté qu'ils adviennent simultanément. Mais l'utilisation d'un ordre partiel ne permet pas vraiment de dire que deux événements sont simultanés. Ils ne sont ni avant, ni après, ni « l'un pendant l'autre » ; ils sont incomparables. L'informatique distribuée offre de nombreux exemples de temps discrets partiels : les événements locaux à deux ordinateurs ne sont pas comparables (mais lors d'une communication entre ces deux ordinateurs, on sait que l'envoi d'un message précède sa réception !). On parle de *modèle asynchrone du temps* quand la relation de succession des instants est partielle et que deux événements ne peuvent se produire simultanément.

Il faut attendre les *langages à parallélisme de données* et les *langages synchrones* pour pouvoir exprimer que deux calculs ont lieu en même temps. Dans un langage de programmation parallèle, on veut pouvoir exprimer que des calculs ont lieu simultanément (mais en employant des ressources différentes) afin d'aller plus vite. Beaucoup de langages parallèles se contentent d'exprimer de la concurrence : deux calculs sont indépendants, ils peuvent donc être réalisés dans n'importe quel

ordre et éventuellement en même temps si l'on dispose des ressources adéquates. Les langages à parallélisme de données permettent d'exprimer qu'un calcul s'effectue simultanément sur tous les éléments d'une structure de données. Dans cette approche, le parallélisme est vu comme une propriété opérationnelle des calculs*, i.e.* qui ne porte pas sur ce qui est calculé mais sur la manière de réaliser le calcul.

Les langages synchrones (voir le chapitre de Gérard Berry dans cet ouvrage) utilisés pour la programmation des systèmes temps réels permettent aussi d'exprimer que plusieurs actions peuvent avoir lieu au même instant. Fondés sur un modèle totalement ordonné du temps on peut exprimer dans ces langages la simultanéité de deux événements, comme par exemple : « à midi, faire telle action », expression impossible dans les langages séquentiels classiques. On peut noter que, pour assurer la simultanéité des calculs, il faut faire l'hypothèse que les calculs ont lieu avec une durée logique nulle (hypothèse de synchronisation forte). Dans la pratique, il suffit qu'un calcul commencé se termine avant l'occurrence de tout autre événement significatif.

Succession et simultanéité sont aussi au cœur de l'organisation temporelle du matériau musical : par exemple, simultanéité des sons qui composent un accord ou des voix qui forment une polyphonie, successions des notes de la mélodie, règles du contrepoint qui organisent la superposition de lignes mélodiques et qui portent donc à la fois sur la succession et la simultanéité. Ces deux dimensions correspondent aux dimensions horizontale et verticale de la notation musicale[3]. Bernd Alois Zimmermann souligne que ce sont les deux formes temporelles de l'intervalle, cette capacité que nous avons à percevoir la distance qui sépare deux sons [Zimmermann, 1957]. L'intervalle est perceptible aussi bien verticalement (simultanéité) qu'horizontalement (succession) et cela n'est possible qu'à l'intérieur du temps.

Cependant dans une partition la succession se définit principalement de manière *extensionnelle* contrairement à la succession des instructions informatiques (nous reviendrons sur ce point dans la section 6). Une autre différence est que la musique utilise la notion de durée de manière essentielle, alors que son traitement reste très élémentaire dans les langages de programmation.

3 Une durée irréductible aux instants

En informatique, les événements correspondant à l'exécution d'un calcul élémentaire sont considérés comme atomiques et discrets : il n'est par exemple pas possible d'évaluer « la moitié d'une instruction ». Par suite, il est possible d'indexer par des entiers la succession des événements de calcul. Cette indexation ne représente que la succession logique d'événements qui sont « logiquement instantanés » et rien n'impose que la « quantité de temps[4] » qui s'écoule entre deux instants repérés par des entiers successifs soit la même quels que soient ces entiers.

Les langages de programmation peinent à parler de l'écoulement et de la durée. Comme il est nécessaire d'exprimer des contraintes de durée (par exemple annuler une transaction sur le web quand aucune confirmation n'a été envoyée dans l'heure), on s'en tire en réduisant cet intervalle de temps à deux instants : celui de son début et celui de sa fin. Pour mesurer une durée, par exemple pour les comparer, on suppose alors qu'il existe un événement périodique, les tics d'une horloge, qu'on compte soigneusement. Un tel codage subordonne la notion de durée à celle de succession mesurable d'événements instantanés. Il ne permet pas de diviser arbitrairement une

[3] Pour Claudy Malherbe, une part importante du développement de la musique occidentale est due à sa capacité à agréger des événements successifs en une simultanéité, opérant un renversement des éléments de l'axe horizontal du mélodique dans celui vertical de l'harmonique [Malherbe, 2009].

[4] L'expression « quantité de temps » est quelque peu paradoxale : en effet, pour mesurer une quantité de temps, il faudrait un *autre* temps pour mesurer l'écoulement du premier. Il est donc impossible de vérifier dans l'absolu "l'isotropie" du temps. Nous l'employons donc ici dans un sens intuitif pour l'informaticien qui l'associe à une certaine quantité de travail exécutée par une tâche entre deux instants qui sont alors qualifiés de *logiques* et qui ne correspondent pas à deux *dates* mesuré sur une horloge physique extérieure mesurant l'écoulement du temps.

durée (on est limité par la période de base de l'horloge[5]). Et il ne permet pas de définir simplement l'écoulement d'une durée, sa manière de *fluer*, en fonction de l'écoulement d'une autre durée. Nous reviendrons dans la section 6 sur l'idée qu'une durée donnée peut, en musique, s'écouler de manière variable mais jusqu'à là, nous assimilerons la durée à un bloc de temps donné, un intervalle.

Si les langages de programmation parlent peu de la durée, plusieurs formalismes ont été développés par les informaticiens pour raisonner sur des relations temporelles qui incluent la durée. Ces formalismes sont utilisés pour faire de la vérification de systèmes temps réel, ou bien pour faire du raisonnement dans des domaines comme les bases de données, le traitement automatique des langues naturelles, la planification, ou encore la simulation. Par exemple, le formalisme des intervalles de Allen introduit treize relations possibles entre des intervalles (précède, succède, intersecte, contient… *cf.* figure 3), la notion d'instant étant absente. Ces relations vérifient de nombreuses propriétés qui permettent de les manipuler simplement, par exemple elles sont mutuellement exclusives.

relation	illustration	interprétation
A < B, B > A	A B	A précède B
A *m* B, B *mi* A	A B	A rejoint B
A *o* B, B *oi* A	A B	A intersecte B
A *s* B, B *si* A	A B	A et B débutent ensembles
A *d* B, B *di* A	A B	A pendant B
A *f* B, B *fi* A	B A	A et B finissent ensembles
A = B	A B	A égale B

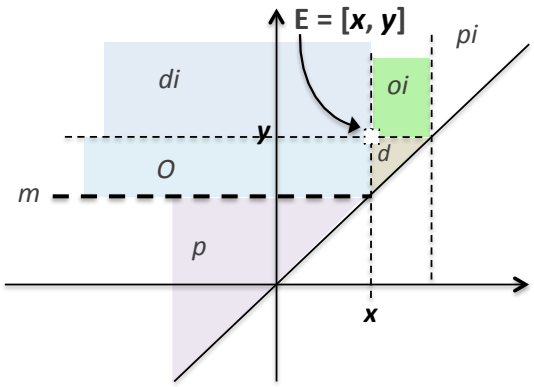

Figure 3. Les treize relations de Allen entre deux intervalles A et B. Les relations sont abrégées ainsi : *m* pour *meet*, *o* pour *overlap*, *s* pour *start*, *d* pour *during* et *f* pour *finish* ; un *i* dans le nom indiquant la relation inverse. Le diagramme de droite propose une représentation géométrique de ces relations : un intervalle E débutant en *x* et s'achevant en *y* est figuré par un point dans un plan et induit des régions qui correspondent aux intervalles satisfaisant une certaine relation de Allen avec E. Par exemple, les intervalles qui sont pendant E sont ceux correspondant aux points dans le triangle rectangle délimité par E et la première diagonale.

Puisqu'il est malcommode de coder une durée par un comptage des instants, il est tentant de construire à l'inverse les instants à partir de la durée. Par exemple, dans la théorie des intervalles de Allen, on pourrait définir les instants comme les intervalles de durée zéro (donc réduits à un point). Il faut pour cela affaiblir considérablement les propriétés vérifiées par les relations sur les intervalles, par exemple ces relations ne sont plus mutuellement exclusives.

En fait, essayer de réduire une notion à l'autre et rendre homogène les notions d'intervalle et de durée, amène des difficultés épineuses [Vila, 2005]. Par exemple, une vitesse ne peut être définie comme la dérivée d'une position que sur un intervalle : même si elle a une valeur en chaque instant, il faut « plus qu'un instant » pour pouvoir la calculer. Un autre exemple classique

[5] Il y a là un point délicat : bien sur un ordinateur exécute des opérations discrètes et donc, *in fine*, son implantation se réfère à une horloge discrète et à un quantum de temps insécable. Mais nous parlons ici du modèle du temps présenté par un langage de programmation à un programmeur, et ceux-ci n'offrent pas de notion de durée qui s'écoule. Cette abstraction est offerte « en dehors du langage » par le système d'exploitation de l'ordinateur qui offre un service de « réveil à une date donnée » construit à partir des quantum de temps de l'horloge matérielle.

de telles difficultés est la définition de la valeur instantanée d'un prédicat qui porte sur un état : imaginons par exemple une lampe qui est allumée pendant une période p_1 qui s'achève en un instant i puis éteinte sur la période p_2 qui débute en i (*cf.* figure 4). Le problème est de déterminer l'état de la lampe à l'instant i. Le problème est de nature logique : si les intervalles p_1 et p_2 sont fermés, la lumière est à la fois allumée et éteinte, ce qui est inconsistant. Si les intervalles sont ouverts, il y a un moment du temps, l'instant i, où l'état de la lumière n'est pas défini. Et les solutions mixtes (intervalle fermé d'un côté et ouvert de l'autre) brisent la symétrie d'un intervalle et sont jugées artificielles.

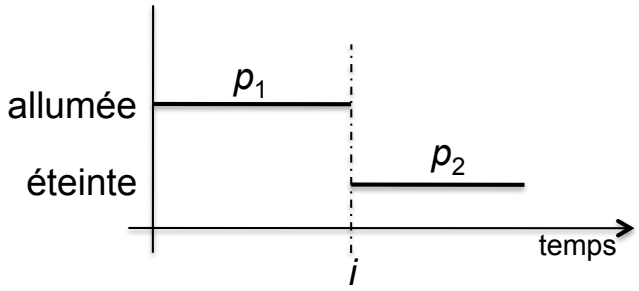

Figure 4. Le problème de la définition d'une valeur d'état sur un instant.

Ne pas réduire les instants à des intervalles de durée nulle, et ne pas définir les intervalles comme un comptage des instants, permet de distinguer naturellement les propriétés qui portent sur les instants des propriétés qui portent sur les intervalles, ce qui évite les difficultés précédentes. Cette irréductibilité des intervalles aux instants, et donc de la durée au séquencement de la succession, évoque la distinction faite par Pierre Boulez entre temps lisse et temps strié, que l'on voit ici comme une durée qui s'écoule continument *versus* des événements qu'on compte :

> Dans le temps lisse, on occupe le temps sans le compter ; dans le temps strié, on compte le temps pour l'occuper.
>
> [Boulez, 1963]

4 Du sens à la dénotation

Nous venons de présenter, très succinctement, ce qui fait « la matière » des relations temporelles : la simultanéité, la durée, la succession. Mais cette présentation ne met pas en évidence qu'il y a plusieurs manières de définir ces relations. Une manière d'aborder cette distinction entre la définition et la chose définie, est de suivre Gottlob Frege, qui introduit les notions de *sens* et de *dénotation* dans un article de 1892 [Fre92]. Il les illustre ainsi : un même objet, par exemple la planète Venus, peut être désigné de différentes manières, par exemple par les expressions « l'étoile du matin » et « l'étoile du soir ». Ces expressions ont la même dénotation, *i.e.* elles réfèrent au même objet, mais correspondent à des significations différentes ; Frege parle de « modes de donation ». Le sens est donc ce qui nous permet, partant d'une expression (un calcul ou une partition) de retrouver sa dénotation (le résultat du calcul, le son produit). Il porte sur la fabrication du résultat plutôt que sur le résultat lui-même.

4.1 Du style déclaratif et du style impératif

Cette distinction est évidente pour un informaticien qui vérifie tous les jours qu'un même résultat peut être obtenu de plusieurs manières différentes, correspondant à des algorithmes différents. Ainsi, obtenir le triple d'une valeur x peut se faire en sommant x trois fois, ou bien par une multiplication entière par 3. Cette distinction est également importante en musique, où le même processus sonore peut être exprimé de différentes façons : il suffit de penser par exemple aux multiples manières de noter le même rythme, ou bien aux différents doigtés permettant d'obtenir la même note sur un instrument.

Mais cette distinction intervient aussi à un autre niveau, celui du langage de programmation et du « style d'expression » qu'il induit. La plupart des langages de programmation sont équivalents au sens où ils permettent de calculer les mêmes fonctions et de répondre au même problème : on dit qu'ils sont Turing-complet. Pourtant, toute personne qui a pratiqué plusieurs langages sait bien que chacun présente des spécificités permettant d'exprimer certains calculs plus simplement ou, au contraire, plus difficilement que d'autres. On parle d'*expressivité* mais ce concept, bien que ressenti intimement par tout programmeur, n'est pas formalisé.

On oppose souvent deux styles de langages : le style *déclaratif* et le style *impératif*. Le style déclaratif privilégie la spécification de ce qui doit être calculé plutôt que le comment. Le style impératif, on dit aussi procédural, se focalise sur l'état du programme, les étapes du calcul et les instructions pour passer d'un état au suivant. On associe souvent le style déclaratif aux langages fonctionnelles ou logiques, proche des notations mathématiques et sans effet de bords (*i.e.*, sans notion d'état global modifié par une instruction). La distinction déclaratif/procédural est aussi pertinente en musique où le compositeur doit décider entre noter ce que fait chaque musicien ou bien noter le résultat attendu (et laisser le musicien se débrouiller). C'est particulièrement vrai en musique mixte où il faut choisir entre noter les paramètres des actions électroniques des logiciels utilisés (style procédural) ou bien essayer de définir le son attendu (style déclaratif). La figure 5 est exemple de partition dans un style « procédural » : le compositeur a indiqué très précisément les gestes du musicien sur un violoncelle, la pièce reposant largement sur des modes de jeu très diversifiés, passant de la note jouée au bruit : frottements de l'archet ou de la main sur les cordes, sons harmoniques, coups répartis sur diverses parties de l'instrument…

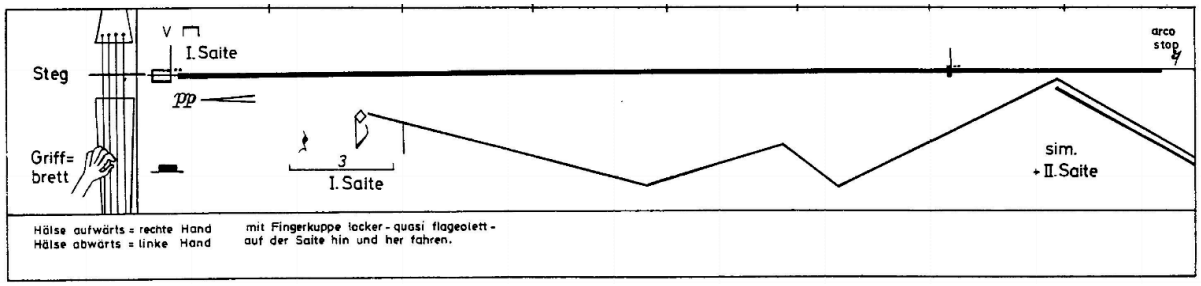

Figure 5. Un fragment de la partition de *Pression* de Helmut Lachenmann.

4.2 Du style modale et du style fonctionnel des définitions temporelles

Différents styles d'expression existent aussi pour la définition des relations temporelles et il faut introduire les deux types de succession d'événements temporels distingués par John Mc Taggart [McTaggart, 1908] : les *A-series* et les *B-séries*. Chacun de ces types formule la notion de succession d'une certaine manière :

- Le style des *A-series* fait appel à des prédicats unaires caractérisant un événement comme étant passé, présent ou futur. Cette manière de décrire la succession des instants se réfère à une succession d'instants « qui passent », la succession étant une transformation continue des instants qui sont d'abord futurs, puis présents puis passés.
- Le style des *B-séries* caractérise les événements relativement les uns aux autres à partir de dates ou de prédicats binaires comme *avant* ou *après* : il n'y a pas de notion de présent et les instants du temps apparaissent *a priori* comme tous égaux, à l'instar des points d'un espace.

John Mc Taggart argumente que la notion de *A-series* est contradictoire : un même événement sera futur puis présent puis passé alors que chacune de ces propriétés exclue les deux autres. Il attaque aussi la notion de *B-series* : puisque la relation entre deux instants sera toujours la même (l'instant de ma naissance précèdera toujours l'instant de ma mort), cette notion est incapable de

rendre compte du changement. Il en tire la conclusion que le temps est une construction humaine qui n'a pas de réalité.

Arthur N. Prior répondra dans les années 1950 aux arguments de Mc Taggart en développant une théorie logique bien fondée permettant de parler des relations entre les instants du temps dans le stricte style des *A-series*. La motivation initiale de Prior est de développer un cadre théorique qui préserve le libre-arbitre et l'idée d'un futur ouvert. Cela n'est pas possible si on adopte le point de vue des *B-series* où les instants futurs existent de la même manière que les instants passés et sont tout aussi déterminés. Il fallait donc montrer, contrairement à ce qu'avance Mc Taggart, que la notion de *A-series* n'est pas contradictoire. Le cadre théorique que Prior développe utilise des opérateurs *modaux* : ces opérateurs s'appliquent à une formule logique pour contextualiser la valeur de vérité de la formule. Par exemple « *il sera toujours le cas que* F » (ici l'opérateur modal est « *il sera toujours le cas que …* ») est *vrai maintenant* si F est *vrai à tout instant ultérieur*. Remarquons que même si nous faisons appel à une description de type *B-series* pour expliquer l'opérateur modal — en faisant une référence explicite à des instants ultérieurs —, l'opérateur modal lui ne les explicite pas. Les logiques modales, qui utilisent des opérateurs modaux, ne sont plus *vérifonctionelles* : la valeur de vérité d'une formule logique ne dépend plus uniquement de la valeur de vérités de ses sous-formules, mais du contexte où cette formule apparaît (et pour les logiques modales, ce contexte est le moment présent). Il existe d'autres modalités que les modalités temporelles comme les modalités épistémiques (relatives à la connaissance), déontiques (morales), doxastiques (portant sur les croyances), etc.

Les logiques temporelles modales, initiées par Arthur Prior pour des raisons philosophiques, ont trouvé des applications très pratiques en informatique. Elles permettent en effet d'exprimer efficacement des propriétés temporelles que doit satisfaire un programme (par exemple qu'un ascenseur n'est pas en mouvement avec les portes ouvertes) et que l'on veut vérifier *a priori*, c'est-à-dire sans exécuter le programme dans toutes les situations possibles (tous les cas de trajets que l'ascenseur peut faire). Le style des *A-series* est aussi celui adopté par des langages pour le temps réel comme Lustre ou Lucid Synchrone : dans ces langages on peut écrire

```
p when q
```

pour indiquer que le calcul de `p` est simultané avec les instants où `q` s'évalue à vrai, et cela « tout le temps ». Si une telle construction permet d'exprimer « *le 1er mars à 13h faire* `p` », une assertion typique des *B-series*, il faut pour cela disposer d'un signal qui est vrai le 1er mars à 13 heures. La construction d'un tel signal est possible si l'on dispose d'un signal horloge qui permet de dater chaque événement, ce qui repousse le problème en dehors du langage.

La logique « classique » permet d'exprimer directement la succession des événements dans le style des *B-series* : il suffit d'augmenter chaque prédicat par une variable temporelle *t* explicitant la date à laquelle ce prédicat est évalué. Ainsi, le prédicat binaire « *x* aime *y* » devient un prédicat ternaire « à la date *t*, *x* aime *y* » et la valeur de vérité de « *x* aime *y* » est une fonction de *t*.

On dispose donc de cadres logiques cohérents aussi bien pour parler de la succession à la manière des *A-series* que des *B-series*. Mais reconnaître qu'il y a deux manières d'aborder la succession nous éclaire sur le traitement du temps lors de la composition (*resp.* le codage) et lors de la performance (*resp.* l'évaluation du code).

Dans un des canons qui composent son *Offrande Musicale*, Johann-Sébastian Bach compose une ligne mélodique qui est un palindrome : dans la seconde partie, les notes jouées sont celles de la première partie mais à l'envers. Nous prenons pour exemple cette forme musicale[6] car elle implique « de renverser le temps » ce qui montre bien que le temps écrit par le compositeur n'est pas celui qui s'écoule pendant qu'il compose et qui lui ne peut pas se renverser. Le compositeur peut penser aux instants du temps qu'il écrit, *hors temps*, c'est-à-dire distinct du temps qui s'écoule au moment de la composition, comme des points de l'espace, de même nature, sans se référer à

[6] qui n'est pas rare : Mozart, Haydn, Bartók, Stravinsky, Webern… nombreux sont les compositeurs qui ont écrit des palindromes musicaux

des instants qui seraient passés, présents ou futurs. En fait, on peut définir cette organisation en palindrome comme un ensemble de contraintes dans le style des *B-series* où la variable temporelle apparaît explicitement ; et on peut résoudre ces contraintes indépendamment du temps qui s'écoule. Par contre, définir de manière modale un palindrome est d'une complication extrême. Il semble donc que le style des *B-series* soit bien adapté à l'expression des contraintes temporelles dans une partition lors de la composition.

> (…) C'est là où l'écriture sort du temps, où l'écriture nous permet de sortir du temps linéaire. Lorsque j'écris je peux à tout moment rater quelque chose, l'effacer et tout recommencer jusqu'à ce que cela me semble présentable. Je peux anticiper sur des choses très lointaines et rappeler des éléments qui semblaient incompatibles entre eux mais qui s'unissent dans une forme commune.
>
> [Manoury, 2012]

Cette situation n'est pas celle de la performance : lors d'une performance, une note jouée est une note passée, fausse ou juste. L'expérience de la performance est celle d'un temps irréversible, avec le passé qui fait mémoire, un présent au sein duquel, dans le vif de l'interprétation, le musicien fabrique des événements sonores, et un futur anticipé mais non encore advenu. C'est le *temps réel* de la performance qui s'oppose au hors temps de la composition. Il reste à vérifier que les logiques modales permettront d'exprimer les contraintes de la performance aussi efficacement qu'elles permettent d'exprimer des contraintes sur l'exécution des programmes.

> (…) le temps de l'écoute et celui du compositeur au travail n'est pas le même. Pour le dire vite, l'ordre des événements dans l'écoute n'est pas le même que celui de la création. (…) Pour donner un exemple très simple : quand j'ai une idée musicale, je peux désirer ne pas l'exposer telle quelle, mais préférer la faire précéder d'une introduction. Cela veut dire que je vais déduire l'introduction de l'idée première. Et pourtant, cela ne sera pas entendu comme une déduction, mais comme une préparation. Le temps de l'écoute est parfois à rebours du temps de la composition.
>
> [Manoury, 2012]

Donnons encore un exemple pour illustrer la différence *A-series*/*B-series* ou encore *temps réel*/*hors temps* et la difficulté qui en découle pour mettre dans le temps les structures qui ont été pensées hors temps. La figure 6 est une esquisse de Christofer Trapani. Le compositeur, inspirée par les canons rythmiques de Colon Nancarrow, veut superposer une phrase musicale jouée par un clarinettiste et le playback de cette même phrase jouée plus vite. Sur la partition présentée, la séquence à enregistrer est indiquée par les marqueurs ① et ② qui repèrent des notes dans la partition. Le playback, à la vitesse 5/3, est matérialisé par la flèche rouge. Le début du playback n'est pas explicitement spécifié mais il faut satisfaire la contrainte suivante : le musicien et le playback arrivent au même instant au point Ⓐ. Si on raisonne en termes de tête d'enregistrement et de tête de lecture sur une bande magnétique, la tête d'enregistrement démarre en ① et poursuit l'enregistrement jusqu'en ② alors que la tête de lecture démarre plus tard, avance plus vite et rejoint la tête d'enregistrement exactement en ②.

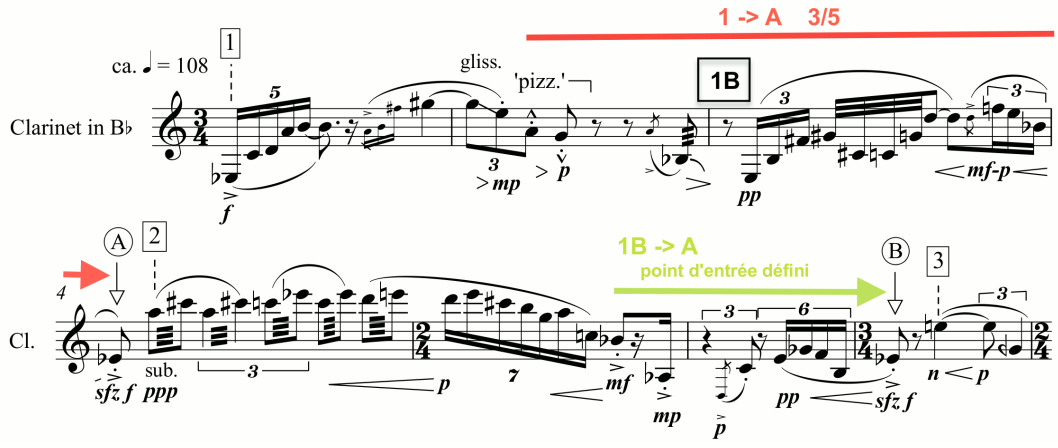

Figure 6. Extrait d'une esquisse de Christopher Trapani

On voit sur cet exemple que le compositeur peut facilement définir le comportement désiré, en explicitant les dates des événements dans la partition. Par contre, la mise en œuvre de ce dispositif lors de la performance gère de manière bien différente la date d'un événement passé, qui est connue, et la date d'un événement à venir. On peut, dans la partition, rechercher la note qui correspond au début théorique du playback. Lors de la performance, dès que cette note est jouée, le playback démarre. Ensuite, à chaque note produite par le clarinettiste, il faut recalculer une estimation du temps restant jusqu'au point Ⓐ afin d'en déduire la vitesse du playback. Cette vitesse, fixe si on raisonne au niveau de la partition, va varier pendant la performance au gré des fluctuations du tempo du musicien. Ces estimations successives de l'arrivée du clarinettiste au point Ⓐ sont nécessaires car c'est un événement à venir.

5 Un temps stratifié

Lors de l'exécution d'un programme *P*, il faut réaliser dans le temps les structures qui ont été spécifiées dans *P*. Cette exécution fait appel à plusieurs « couches de temps » : celles des circuits matériels qui composent le calculateur, celles des instructions élémentaires qui peut se décrire par l'avancement du compteur d'instructions, et puis des couches de temps plus logiques qui correspondent à l'organisation des différentes échelles de temps liées à l'application et déjà présentes dans *P* (*cf.* figure 7). Ces différentes strates temporelles logiques existent aussi dans la musique. Elles sont particulièrement explicites dans une pièce comme *Gruppen* de Karlheinz Stockhausen, qui met en jeu trois ensembles instrumentaux, chacun avec son chef d'orchestre, jouant des voies distinctes hormis certains passages où un seul processus musical passe d'un orchestre à l'autre.

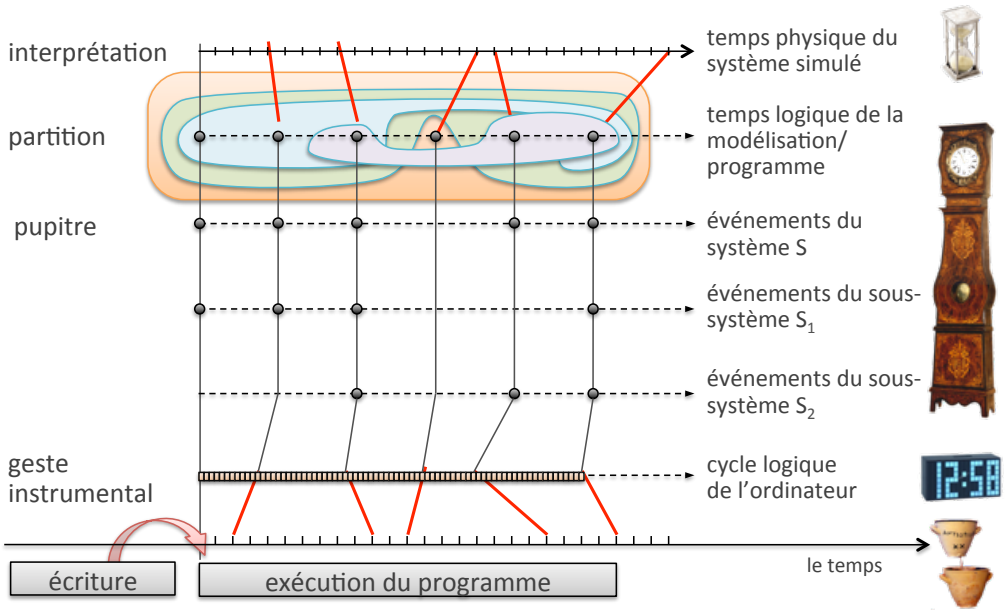

Figure 7. Diverses échelles de temps lors d'une simulation : certaines échelles de temps correspondent à des événements induits par la réalisation du calcul, d'autres correspondent à la structure temporelle de l'application. La musique doit aussi gérer des échelles de temps différentes, allant de la production du son (geste instrumental, de l'ordre de la centième de seconde) à la structure d'une pièce (de l'ordre de l'heure). De plus, dans l'architecture d'une pièce, un compositeur est amené à gérer des couches sonores qui ont leur propre temporalité.

Revenons aux différentes échelles temporelles mises en œuvre lors de la réalisation d'un calcul, par exemple la synthèse par modèle physique d'une corde vibrante qui produira des ondes sonores. Il faut résoudre numériquement un ensemble d'équations modélisant l'évolution d'un

système mécanique dans le temps. Le temps intervient alors une première fois, mais c'est un temps simulé, qui n'a pas nécessairement de lien avec le temps physique qui s'écoule lors de la résolution numérique. Cependant, on peut aussi vouloir produire cette synthèse sonore en temps réel, par exemple pour entendre le son en même temps que l'on agit sur les paramètres du système modélisé dans le cas d'un instrument virtuel joué « en live ». Pour cela il faut que la résolution numérique se fasse suffisamment rapidement pour qu'on puisse disposer de la description de l'onde sonore à temps pour une restitution interactive. Et cela ne suffit pas : il faut non seulement disposer des résultats de la simulation à temps, (et donc calculer suffisamment vite) mais il faut aussi les délivrer au système de rendu sonore (la carte son d'un ordinateur) au bon moment, ni en avance, ni en retard. Le temps du système simulé est le même que le temps réel, celui utilisé pour faire le calcul (*cf.* figure 8.a). Notons que dans cet exemple, il faut gérer non seulement le temps au niveau de la corde qui vibre, mais aussi un temps plus macroscopique, correspondant aux événements de pincement de la corde qui changent les conditions initiales du système, et qui peuvent correspondre à des interactions immédiates avec un utilisateur.

La figure 8.a reprend la figure 7 en faisant coïncider le temps spécifié dans la partition (*resp.* le programme) avec le temps de la performance (*resp.* le temps du calcul). Il existe cependant une autre manière de « tordre » le diagramme de la figure 7, le long d'un axe vertical, afin de faire coïncider la phase d'écriture et celle de production, *cf.* figure 8.b. Ce cas de figure correspond aux musiques improvisées ou bien à la pratique du *live coding*, mais n'a pas vraiment d'équivalent en informatique où l'on considère généralement que le temps de l'écriture d'un programme est distinct de son exécution. Certains langages de programmation sont interprétés, ce qui permet d'enchaîner rapidement les phases de codage et de test. Mais ces tests ne constituent pas la finalité du codage : la plupart du temps le programme obtenu est ensuite exécuté pour répondre aux besoins d'une application, sans que l'on revienne sur l'écriture du programme. À l'opposé de cette distinction, le *live coding* réunit dans le même temps improvisation et composition algorithmique [Sorensen, 2010]. Le processus d'écriture est rendu visible généralement en projetant l'écran d'un ordinateur à destination du public. L'idée est que la programmation du processus sonore prend place en temps réel et en interaction avec la production sonore. Cet usage, qui perturbe des catégories peu questionnées, pousse l'informaticien à la réflexion.

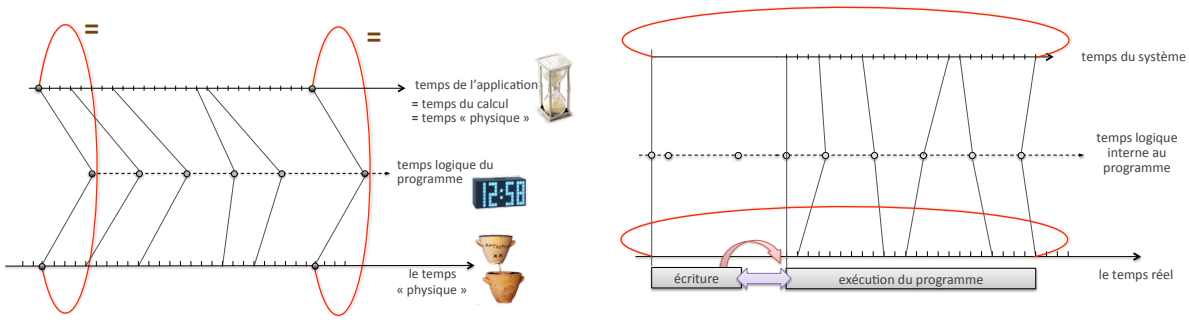

Figure 8. Réalisation dans le temps réel des structures temporelles spécifiées hors temps lors de la composition.

6 La forme du temps, œuvre ouverte et programme non-déterministe

6.1 Temps linéaire, temps ramifié

Succession et simultanéité sont compatibles avec plusieurs formes globales du temps, *cf.* figure 9. On connaît le temps cyclique, par exemple celui de l'éternel retour nietzschéen, qui

échappe au temps linéaire[7]. On est peut être moins habitué à la représentation des ordres partiels (introduit à la section 2) où certains instants sont incomparables. Le graphe de la succession n'est alors plus une droite ou un cercle, mais bifurque vers des instants incomparables, avant de se rejoindre en des instants ultérieurs[8].

Les représentations évoquées ci-dessus, linéaire totale, linéaire partielle ou cyclique, correspondent à des conceptions du temps qui relèvent plutôt de la perspective des *B-series* : les instants passés, présents et futurs sont également réels et de même nature. Si l'on veut figurer un temps qui relèverait purement des *A-series* et qui préserverait l'indétermination du futur, il faut représenter un arbre où un instant est singularisé : l'instant présent. Les branches alternatives partant du présent correspondent aux futurs possibles d'un monde ouvert. L'instant présent se déplace, en suivant un chemin. Cette ligne de temps dans l'arbre représente « ce qui a été ». Toutes les autres branches en arrière de l'instant présent représentent « ce qui aurait pu être », et toutes les ramifications partant de l'instant représentent « ce qui pourrait être ». Parmi ces branches potentielles, une seule s'actualisera.

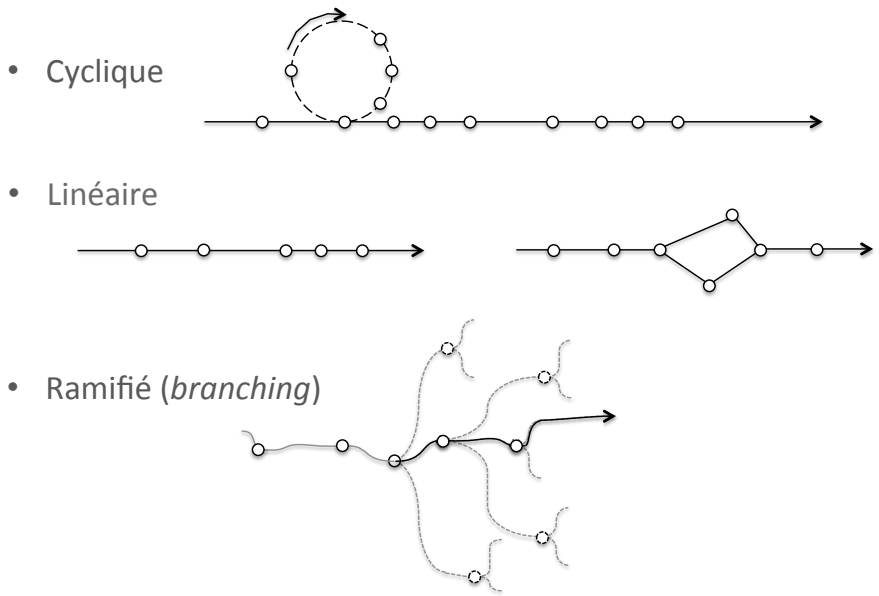

Figure 9. Les formes du temps.

Le temps ramifié permet de rendre compte correctement des programmes *non-déterministes*. On regarde souvent un programme qui s'exécute comme un système déterministe : les données en entrée déterminent complètement le calcul à effectuer et à chaque instant, il est possible de prévoir, en fonction de l'état actuel, quel sera l'état suivant du programme. Cependant les informaticiens développent depuis longtemps des programmes non-déterministes, et ce pour plusieurs raisons : un programme peut utiliser des nombres aléatoires, par exemple parce qu'il simule un processus stochastique ou parce que l'algorithme est probabiliste[9] ; ou encore le programme n'effectue pas la transformation d'une entrée en une sortie déterminée, mais réagit en

[7] En musique, *Zyklus* de Karlheinz Stockhausen est une pièce qui peut illustrer le temps cyclique. Composé en 1959, sa forme, circulaire et sans spécification d'un point de départ, est matérialisée par une partition dont la reliure en spirale ne distingue pas de première page ni de dernière page. L'interprète peut débuter où il le souhaite et doit jouer la partition, de gauche à droite ou bien de droite à gauche, jusqu'à retrouver son point de départ.

[8] Par exemple, l'envoi de deux messages à deux machines isolées (qui ne partagent pas d'horloge commune) précède leur réception, sans qu'on puisse en dire plus sur l'ordre relatif de ces réceptions.

[9] Par exemple, l'un des algorithmes les plus efficaces pour tester si un nombre *n* est premier, est probabiliste : il donne avec une faible probabilité une réponse erronée, mais on peut demander à ce que cette probabilité soit aussi petite que voulue (au prix d'une exécution plus longue). On peut par exemple contraindre cette probabilité à être plus faible que la probabilité qu'un rayon cosmique vienne perturber l'ordinateur et rendre faux le résultat d'un calcul déterministe « correct ».

continu aux sollicitations aléatoires d'un environnement ouvert (c'est le cas des programmes communiquant via un réseau, ou des programmes qui interagissent avec des utilisateurs).

Linéaires, cycliques ou ramifiées, ces représentations sont des modèles étudiés dans diverses variantes des logiques temporelles. On peut citer par exemple LTL (*linear time logic*) qui correspond à un temps linéaire et CTL (*Computation tree logic*) à un temps ramifié. Ces modèles nous permettent aussi de réfléchir à la structure d'une pièce musicale écrite.

6.2 Œuvre ouverte et définition en intension

Une partition se présente habituellement de manière linéaire et *extensionelle* : chaque note jouée est indiquée explicitement dans l'ordre de sa succession. En mathématique, un ensemble S est défini en extension quand on donne la liste de ces éléments, comme par exemple $S = \{0, 1, 2, 3, 4, 5\}$. Cependant, une partition peut inclure des indications de reprise permettant la répétition de fragments plus ou moins longs (barre de reprise 𝄆 et 𝄇, *da capo*, signe de renvoi). Ces indications permettent d'échapper au déterminisme de la succession des notes indiquées par la partition, mais de manière relativement restreinte car ces indications n'impliquent que peu sinon aucune décision de la part de l'interprète. Ce sont surtout des commodités de notation permettant de « factoriser » certains passages.

Dans les *formes ouvertes*, les interprètes ont une bien plus grande liberté pour organiser eux-mêmes certaines séquences musicales plus ou moins préétablies. De telles partitions correspondent à un *processus génératif* car elles permettent de générer plusieurs œuvres. Nous qualifions ces partitions d'*intensionnelles*. En mathématique, un ensemble S est défini en intension, on dit aussi en compréhension, quand on caractérise les éléments de S par une propriété, par exemple : $S = \{n \mid n \in \mathcal{N} \text{ et } n \leq 5\}$. Un programme est un ensemble d'instructions et c'est un ensemble défini en intension. En effet, un ordinateur moderne exécute typiquement de l'ordre du milliard d'opérations à la seconde, et donc, si l'on écrivait explicitement toutes les instructions exécutées par un programme, il faudrait en écrire plus d'un milliard pour obtenir un calcul qui dure à peine une seconde. Mais les langages de programmation, grâce à la notion de boucle, de fonction et de récursion, permettent d'écrire des programme qui sont bien plus courts qu'un milliard de lignes et qui nécessitent beaucoup plus d'une seconde pour s'exécuter.

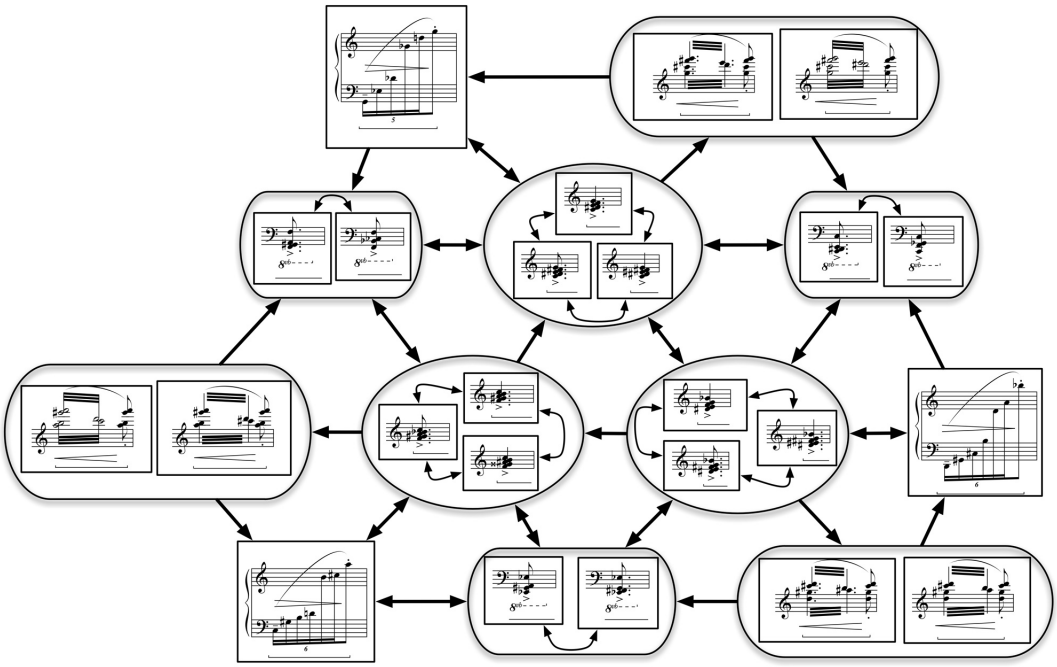

Figure 10. La partition ouverte de « Observing Squirrels », la première étude des *Piano Etudes* de Jason Freeman. Il serait difficile de définir cette notation sous la forme habituelle d'une partition en utilisant les indications traditionnelles de reprises. La définition sous forme de graphe est bien plus commode, si ce n'est pour l'interprète, au moins pour le compositeur. Le lecteur peut créer sa propre version de l'étude à partir de la page web
http://turbulence.org/spotlight/pianoetudes/net.jasonfreeman.pianoetudes.PianoEtudes/wordpress/

La problématique est la même pour les œuvres ouvertes : l'ensemble des possibles est généralement trop grand pour être noté explicitement et la partition dénote alors un processus qui définit chacune des variantes possibles. La figure 10 montre une partition ouverte de Jason Freeman sous la forme d'un graphe : chaque sommet correspond à un choix ou bien à une répétition possible de fragments élémentaires. Les arcs partant d'un sommet sont autant d'alternatives pour la poursuite de la performance. On peut facilement transformer ce graphe en un arbre correspondant à la représentation ramifiée des futurs possibles de la figure 9 : une fois le sommet de départ fixé, il suffit de « déplier » le graphe en parcourant tous les chemins possibles[10]. Chaque parcours correspond à une branche qui descend dans l'arbre.

Dans cet exemple d'œuvre ouverte, les fragments musicaux qui doivent se succéder sont en petit nombre et les règles de succession sont explicitement définies par les liens du graphe. Ce n'est pas toujours possible : par exemple, les fragments musicaux peuvent être trop nombreux pour être énumérés[11], et les enchaînements admissibles peuvent être décrit intensionnellement par des contraintes[12]. De ce point de vue, les partitions usuelles sont des graphes linéaires extensionnels, les « petites » œuvres ouvertes des graphes non-linéaires encore définis en extension et les « grandes » œuvres ouvertes des graphes non-linéaires définis en intension. Les qualificatifs de « petit » et « grand » sont bien sûr à prendre au sens de la complexité du processus génératif.

[10] La construction de l'arbre des successions possibles des fragments de « Observing Squirrels », se heurte à une seule difficulté réelle : la durée de la pièce n'étant pas fixée, l'arbre correspondant doit être infini. Cela ne veut pas dire qu'il existe une étude qui ne s'arrête jamais, mais plutôt que, pour toute étude, il en existe une de durée supérieure (il suffit de poursuivre en allant visiter un nœud supplémentaire).

[11] C'est le cas par exemple si ces fragments sont définis eux même par un processus génératif de niveau inférieur (une grammaire, un graphe d'alternatives, etc.).

[12] Par exemple, on peut dans tous les chemins possibles du graphe de la figure 10 ne permettre que ceux qui passent une fois et une seule par chaque fragment (chemin hamiltonien), une contrainte souvent explorée par un compositeur comme Tom Johnson.

> Je connais toutes les possibilités, mais je n'ai pas prévu la formulation concrète de toutes ces possibilités, et je sais que ce réseau peut se tisser de toutes ces façons-là : ce sont ces possibilités que j'envisage et dont je ne sais pas et ne veux pas savoir comment elles vont se concrétiser.
>
> [Boucourechliev, 1997]

Les programmes étant des objets intensionnels, il n'est pas possible de voir immédiatement les différentes étapes d'une exécution. C'est encore plus vrai pour les programmes non-déterministes. L'objectif des techniques de *model-checking* et de pallier à ce manque de connaissance en permettant de vérifier qu'une certaine propriété exprimée dans une logique temporelle est vérifiée par toutes les exécutions d'un programme. L'idée de base consiste à représenter l'arbre des différentes exécutions possibles sous une forme symbolique et compacte : dans cet arbre, chaque sommet correspond à un état du programme (le contenu possible de sa mémoire) et un état *a* est relié à un état *b* s'il existe une exécution du programme qui permet de passer de *a* à *b*. Une exécution correspond donc à un chemin dans ce graphe. Dans une certaine mesure, il est possible de vérifier ainsi des propriétés d'atteignabilité (à partir d'un état, peut-on atteindre un autre état ?), des propriétés de vivacité (est-il toujours possible de repartir d'un état ?), des propriétés de convergence (quelque soit l'état de départ, finit-on toujours sur un même état final ?), des propriétés d'exclusion (tout chemin qui passe par un état *a* ne passera pas par *b*), *etc.* Les techniques développées par les informaticiens pour comprendre le comportement d'un programme non-déterministe seront peut-être utiles un jour aux compositeurs pour contrôler leurs œuvres ouvertes : est-ce que les contraintes définies sur l'enchaînement de parties élémentaires admettent pour solution une pièce dont la durée est entre 10 et 20 minutes ? Est-ce que ces contraintes permettent de s'assurer qu'il n'y aura pas de répétitions successives du même fragment ? Est-ce que l'occurrence d'un fragment est exclusive de l'occurrence d'un autre ? Répondre à ce type de questions permet au compositeur de vérifier que son *intention* est bien préservée malgré le caractère ouvert de l'œuvre.

7 *Antescofo* : accorder le temps de l'homme et celui de la machine

> Car tout le monde a compris que faire des musiques avec les seules machines, comme cela se faisait dans les premiers temps de la musique électronique, n'a plus aucun sens. Ce qui importe avant tout, c'est de développer des systèmes de captation et d'analyse du monde extérieur. En musique, cela veut dire analyser une interprétation, des manières de jouer, et en déduire des conséquences possibles et intéressantes. Ce mécanisme de déduction fait partie de l'appareil compositionnel, il doit être intégré. Si un interprète X joue un événement A, alors la machine produira B ; s'il joue C, elle produira D, etc. B et D peuvent être des procédures extrêmement sophistiquées et influencer très bien le comportement de X. On le voit donc, l'humain entre en force dans ce dispositif.
>
> [Manoury, 2012]

À la section 3 de ce chapitre, nous avons présenté une notion de durée, irréductible aux instants, et assimilé celle-ci à la notion d'intervalle. Mais avec Henri Bergson, la notion de durée prend une toute autre dimension en opposant un *temps intérieur*, *réel*, considéré comme le temps du mouvement pur, de l'écoulement et de la durée, à un *temps objectif* concrétisé par des instants ponctuels en relation de succession mesurable extérieurement[13]. Les mots sont de Bernd Alois Zimmermann qui voit en la musique la possibilité de réconcilier ces deux notions : le *temps effectif* — le temps cosmique et mesurable des horloges — et le temps de l'expérience — celui de la conscience perceptive intérieure où, grâce à la mémoire et à l'imagination, passé, présent, et avenir se confondent[14].

Cette distinction entre ces deux sortes de temps et la possibilité de les réconcilier, de les accorder dans une *interaction musicale*, est un des fondements du système *Antescofo*. *Antescofo* est un

[13] Gérard Grisey parle lui du *squelette du temps* en se référant aux divisions temporelles du temps chronométrique, et de la *chair du temps* pour parler de l'intuition directe et de la perception du temps en relation avec le matériel sonore [Grisey, 1987].

[14] On se reportera au chapitre de Laurent Feneyrou dans ce même ouvrage.

projet qui a été initié en 2007 par Arshia Cont, pour la pièce *Of Silence* de Marco Stroppa. S'inscrivant dans la tradition du suivi de partitions[15] il en a étendu l'idée en couplant une machine d'écoute avec un moteur réactif rendant possible un véritable dialogue entre les musiciens et la machine. C'est la prise en compte des temps musicaux hétérogènes qui permet cette interaction fluide entre ordinateurs et instrumentistes. Dans le reste de ce chapitre, nous esquissons la problématique de l'interaction musicale et l'approche proposée par *Antescofo* pour réconcilier le temps intérieur de chaque musicien et celui de la machine.

7.1 Interprétation musicale et coordination des musiciens

En écoutant deux interprétations de la même œuvre musicale, on constate que si la succession des événements musicaux reste la même, leurs paramètres (le nombre de pulsations par minute, la durée des notes, leur amplitude, etc.) fluctuent. Ce sont ces fluctuations, qui ne dépendent pas du hasard mais relèvent de la volonté du musicien[16], qui donnent à chaque interprétation son caractère unique et sa couleur singulière. On peut alors supposer, comme le fait Zimmermann dans le passage ci-dessous, que le tempo et ses variations sont contrôlés par ce temps intime qui unifie nos expériences et fixe l'écoulement continu d'une durée perçu comme densité du vécu :

> Nous appellerons *durée temporelle effective* la portion de temps que nécessite une œuvre musicale pour son exécution. Cette durée n'est cependant pas dans son extension une grandeur constante dans le sens où elle resterait la même à chaque exécution. Autrement dit, les conditions toujours changeantes de l'exécution musicale, même si elles sont la plupart du temps d'ordre minimal, occasionnent des durées d'exécution variables pour une même composition, alors qu'au contraire les proportions de toutes les relations métriques, rythmiques et donc temporelles demeurent inchangées à l'intérieur de la durée temporelle (*effective*) qui, elle, varie. De cette manière, le temps inhérent à une composition sera doublement organisé : d'une part par le choix d'une certaine unité de temps (*effective*) qui a la fonction de tempo musical, d'autre part, par le choix d'une certaine unité de temps (*intérieure*) qui ordonne les rapports entre Intervalle et Temps. Les unités de temps, intérieures aussi bien qu'effectives, sont définies par la conscience intérieure du temps musical à laquelle, en ce sens, nous devons attribuer la fonction régulatrice de l'expérience et de la perception du temps en musique.
>
> [Zimmermann, 1957]

L'interprétation musicale devient d'autant plus complexe que plusieurs musiciens se produisent sur scène. Chaque musicien est alors responsable d'interpréter sa propre partition en cohérence avec celles des autres : il doit ralentir ou accélérer en des points musicalement choisis pour se synchroniser avec les autres et tolérer les erreurs de jeu avec pour objectif de produire un ensemble cohérent respectant les contraintes temporelles données par la partition d'origine. Cette synchronisation est assurée « en temps réel », soit implicitement par les musiciens entre eux dans le cas d'un petit ensemble de musique (un quatuor à cordes par exemple), soit via un chef d'orchestre quand un grand nombre de musiciens sont présents sur scène.

Cette aptitude à se synchroniser et se coordonner musicalement est une des compétences fondamentale des musiciens. On peut supposer que cette capacité implique pour les instrumentistes, de synchroniser entre eux leur *temps intérieur* à travers les événements musicaux (les notes et leurs durées) qu'ils concrétisent dans le temps objectif du métronome et des horloges. Nous utiliserons le terme de *temps relatif* pour le premier et de *temps physique* pour ce dernier. Mais peut-on parler de manière logiquement cohérente de *deux* temps et de l'*écoulement* des durées de l'un relativement à l'autre ?

---

[15] Un suiveur de partition est un logiciel permettant de localiser en temps-réel dans une partition donnée à l'avance, les événements musicaux produits par des musiciens.

[16] car ces fluctuations sont répétables par le musicien qui peut donner deux fois la même interprétation

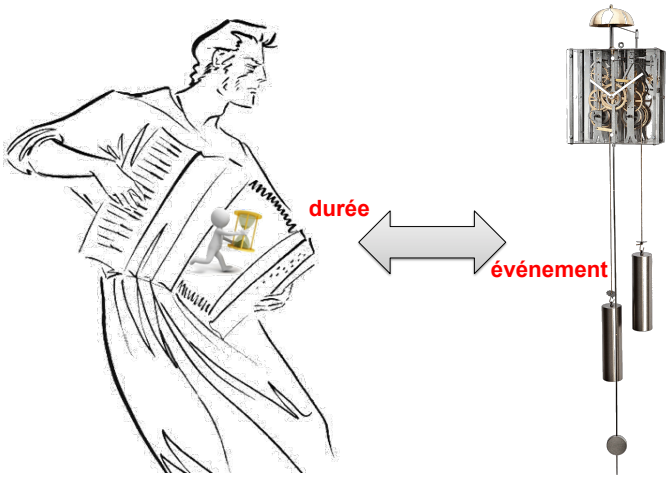

Figure 11. L'instrumentiste articule sa conscience de la durée et les événements musicaux (les notes et leurs durées, mais aussi les gestes) qui se concrétisent dans le temps effectif partagé par les autres musiciens et les horloges.

Nous proposons d'accéder à l'écoulement du temps relatif via la notion de *tempo*[17]. Le tempo est d'abord défini comme l'allure ou la vitesse d'exécution d'une œuvre musicale. L'échelle qualitative, de *larghissimo* à *vivacissimo*, a ensuite laissé la place à une définition quantitative avec l'invention du métronome. Le tempo devient alors la mesure objective du nombre d'événements musicaux à la minute. Ici, nous ferons l'hypothèse que le tempo, mesuré à chaque production d'un événement musical, peut se concevoir comme une *observation discrète* de l'*écoulement continu* du temps intérieur d'un instrumentiste relativement au temps physique. Ainsi, si une partition indique « 60 à la noire » et que l'on observe lors de l'interprétation 70 noires à la minute, on pourra dire que le temps intérieur de l'instrumentiste s'écoule 7/6 fois plus vite que le temps physique. Mais inversement, si on demande à cet interprète quel est le tempo d'un processus qui produirait 60 noires à la minute, uniquement à partir de sa conscience de l'écoulement du temps et sans qu'il se réfère à un métronome, il répondra qu'il s'écoule 6/7 fois la quantité de notes que lui-même produit quand il réalise « 60 à la noire ».

Utilisons les minutes pour mesurer le temps physique, et les pulsations pour le temps relatif. En poursuivant le raisonnement précédent, on peut déduire qu'il s'écoule 1 minute par minute, ou bien 1 pulsation par pulsation. Cette conclusion, à laquelle on est amené lorsque l'on essaye de mesurer l'écoulement du temps, a souvent été invoquée comme un argument contre cette notion [Price, 1996] : si le temps s'écoule, alors il est naturel de se poser la question de sa vitesse d'écoulement. Mais cette notion requiert un « autre temps » dans lequel mesurer cette vitesse. On doit alors stopper une régression à l'infini en envisageant que le temps passe à la raison d'une minute par minute. Cette réponse ne semble pas très pertinente de premier abord. Et envisager deux temps ne résout pas le problème.

Tim Maudlin s'attaque à cet argument en faisant remarquer que pour mesurer le passage du temps, de manière analogue à la mesure de la vitesse d'un fleuve, il faut mesurer combien l'état temporel des choses a passé au bout d'un temps donné [Maudlin, 2002]. En effet, pour calculer la vitesse du flot d'une rivière, il faut observer quelle distance a parcouru une goutte d'eau : de

[17] Certaines études nie cette possibilité, en particulier parce que le tempo est une mesure discrète et qu'on ne peut l'extrapoler en une « courbe de tempo ». Par exemple [Desain, 1992] conclue d'une série d'expériences psychologiques que la notion de courbe de tempo « est une notion dangereuse, malgré un usage généralisé et une description commode, car elle renforce l'idée fausse que cette notion possède une réalité musicale ou psychologique. Il n'y a pas de courbe de tempo abstrait en musique, pas plus que dans la tête d'un instrumentiste ou d'un auditeur ». Cette conception est vivement combattue dans [Mazzola, 1994] qui voit dans les problèmes soulevés par la notion de courbe de tempo l'effet d'une définition imprécise, de la difficulté à la calculer et de son mésusage, et qui en réponse propose un cadre théorique cohérent.

combien elle s'est éloignée de la source et de combien elle s'est rapprochée de l'embouchure. De la même façon, quand passe une minute, de combien me suis-je éloigné de ma date de naissance et de combien me suis-je rapproché de la date de ma mort : la réponse est bien de une minute. La réponse n'est pas inintelligible.

Une autre image permet de se convaincre de sa pertinence, et s'applique bien à l'articulation du temps relatif et du temps physique en musique : celle des taux de change entre devises. Si l'on décide d'un ensemble bien défini de marchandises, il est possible de comparer leur coût dans diverses devises et d'établir des taux de change équitable entre celles-ci : combien d'euros pour un dollar, et vice-versa. On peut se poser dans ce cadre la question du taux de change pour la même devise : la réponse sera bien évidemment de un pour un.

Si temps relatif et temps physique sont interconvertibles, alors pourquoi parler de deux temps ? Il devrait être possible de toujours tout exprimer dans le cadre de référence du temps physique, qui, lui, est mesurable objectivement et simplement partagé. Cette idée n'est pas du tout praticable. D'abord, parce que *la musique subordonne le temps physique au temps relatif*, et non l'inverse. Les partitions sont écrites par rapport à un tempo, qui rend compte du temps relatif, et non du temps physique. Ce tempo est souvent un paramètre non explicité, par exemple dans la notation baroque où, à coté d'éléments qualifiés d'« absolus » (la hauteur de la note dans la majeure partie du répertoire occidental) d'autres sont « relatifs », c'est-à-dire laissés à la discrétion de l'interprète à l'intérieur de certaines limites (le tempo donc, mais aussi les dynamiques). Ensuite, parce que la relation entre temps relatif et temps physique n'est complètement explicitée qu'une fois que le temps s'est écoulé et que la musique a été jouée. Si temps relatif et temps physique sont interconvertibles, le « taux de change » varie dans le temps et n'est pas prévisible : à un moment donné, on n'a accès qu'aux taux passés. Enfin, parce que la prise en compte du temps relatif, dès que l'on sait mesurer ce temps à travers ses manifestations objectives et discrètes (les événements musicaux), permet l'instauration d'un vrai dialogue musical entre l'homme et la machine, en permettant à cette dernière de se synchroniser et se coordonner avec ce qui est *attendu* et *anticipé* par l'instrumentiste humain. C'est à ce dialogue que vise le projet *Antescofo*.

7.2 Le projet *Antescofo* : écoute anticipative et réaction

Doter un ordinateur des mêmes capacités de coordination et d'anticipation qu'un instrumentiste humain est un problème difficile. *Antescofo*, un système développé à l'Ircam, mobilise pour y arriver des techniques qui relèvent du traitement du signal, de l'apprentissage automatique et des systèmes temps réel. L'enjeu ? mieux comprendre la nature de l'anticipation musicale chez l'homme, mais aussi ouvrir la voie à de nouvelles formes d'interaction musicale et de création sonores. Le succès de cette analyse fine et en temps réel du discours musical a ouvert la voie à de nouvelles applications. Dans le contexte musical décrit plus haut, il ne s'agit pas simplement de produire des sons ou de les transformer à des dates préfixées à l'avance par le compositeur, mais de permettre à la machine d'interpréter sa partition en coordination et en synchronisation avec le jeu des musiciens sur scène. Un tel système informatique doit donc être capable de mener deux tâches en parallèle : écouter et réagir, *cf.* figure 8.

7.3 Une écoute anticipative

Tout commence donc par le suiveur de partitions et le difficile problème de la détection, de la reconnaissance et de l'analyse en temps réel des sons instrumentaux. À partir d'un flux d'informations audio, capté par un micro, il faut reconnaître la survenue des événements symboliques notés dans une partition. Aux fluctuations de l'interprétation, il faut ajouter les difficultés de reconnaissance dans un environnement non contrôlé (bruits de fond, erreur d'accordage, biais des dispositifs de captation), la variabilité introduite par les instruments et les différentes techniques de jeu propre à chaque musicien, ainsi que la gestion des éventuelles erreurs provenant des instrumentistes.

Antescofo s'appuie sur un système de reconnaissance probabiliste capable de détecter les événements d'une partition ainsi que certains paramètres musicaux comme le tempo en temps

réel. Le choix d'un système probabiliste est lié à l'incertitude dans le contenu d'information des signaux audio provenant des musiciens et permet de mieux gérer la tolérance du système dans un tel environnement. La partition musicale est représentée dans le système sous forme d'une chaîne d'états, dont chacun correspond à un événement de la partition (note, accord, silence…) avec ses propriétés musicales telles que sa fréquence fondamentale (hauteur) et aussi sa durée. Le rôle du système de reconnaissance est ensuite de trouver (en temps réel) le meilleur chemin sur la chaîne qui correspond au flux audio. Les techniques mises en œuvre sont très similaires à celles utilisées pour le suivi des missiles ou la reconnaissance de la parole. Cela pourtant ne suffit pas : l'identification de deux sons superposés (polyphonie) est un problème qui reste toujours à l'heure actuelle très difficile, et encore plus quand il faut faire cette reconnaissance en temps réel et sans apprentissage préalable.

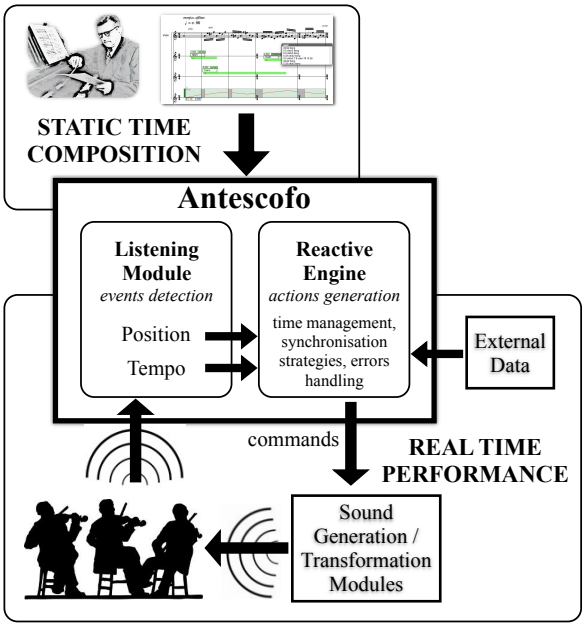

Figure 12. Architecture du système *Antescofo*

Pour faire face à ces défis, *Antescofo*, comme les êtres humains, combine plusieurs sources d'information, chacune faillible, pour arriver à un suivi fiable. Parmi les processus cognitifs impliqués dans la synchronisation musicale, le phénomène de l'anticipation musicale joue un rôle fondamental. En effet, pour pouvoir jouer ensemble, les musiciens, en plus d'une écoute élaborée, anticipent le jeu de leurs collègues pour se synchroniser avec eux au plus juste : ils synchronisent leur temps relatif. Cette anticipation est possible car *Antescofo* extrait du jeu du musicien son tempo, voir figure 9.

Pour ce faire, *Antescofo* se fonde sur un modèle dynamique de l'attente et du suivi des événements temporels par un humain [Large, 1999]. Edward Large et Mari Riess Jones font remarquer qu'un observateur est capable de repérer des rythmes stables malgré de grandes fluctuations dans les périodicités qui composent ces structures. La flexibilité temporelle d'une interprétation n'empêche pas les auditeurs, même non musiciens, d'identifier la pulsation, de reconnaître un rythme et de suivre le tempo tout en percevant les déviations à cette structure. Plusieurs modèles de représentation cognitive des structures temporelles ont été proposés, mais ils peinent à expliquer cette capacité. Les modèles fondés sur une horloge intérieure fixée sont trop sensibles à la variation des observations. Et les modèles statistiques qui proposent de représenter une durée en terme de nombre d'impulsions d'une ou plusieurs horloges aléatoires, peinent à distinguer certaines séquences rythmiques complexes. Le modèle de Large et Jones repose sur l'idée d'attention dynamique pour synchroniser la *phase* et la *fréquence* d'un oscillateur interne avec les stimuli extérieurs. Le mécanisme est similaire à celui qui intervient dans la

sympathie des horloges, un phénomène décrit par Christian Huygens alors qu'il travaillait sur une paire d'horloges en vue de résoudre le problème épineux de la détermination de la longitude en mer [Sainte-Marie, 2008]. Deux horloges à pendule qui sont légèrement déphasées se synchronisent petit à petit et trouvent une cadence commune. La synchronisation est due à la vibration du support commun qui transmet un peu d'énergie entre les deux systèmes. La « sympathie » observée par Huygens constitue un phénomène d'entrainement ou de verrouillage de fréquence caractéristique que l'on peut observer dans d'autres exemples de systèmes auto-oscillants. Plusieurs expériences ont validé les prédictions du modèle qui décrit par exemple correctement le comportement d'un musicien face à des changements de tempo.

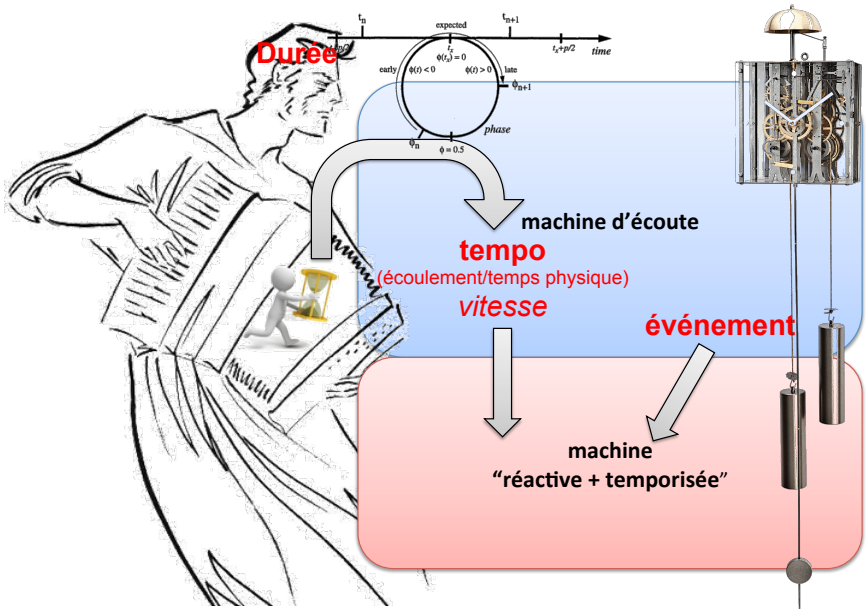

Figure 13. Outre la détection des événements musicaux (notes, accord, etc.) dans le flux audio du musicien, *Antescofo* extrait une estimation du tempo fondé sur un modèle reposant sur la « sympathie des horloges ». Tempo estimé et événements détectés sont transmis à la machine réactive qui coordonne dans le temps (chronométrique ou relatif au musicien) divers processus électroniques.

Ainsi, à coté du système d'analyse du signal audio continu, *Antescofo* est doté d'un système d'anticipation des événements fondé sur ce modèle cognitif. Les deux systèmes correspondent à des agents qui sont en compétition et collaborent à tout instant : l'agent en charge de l'anticipation aide l'agent chargé de l'analyse audio en anticipant de futurs événements, et l'agent d'analyse audio permet à l'agent d'anticipation de prévoir la date des événements futurs à partir des dates d'arrivées des événements passés et de l'estimation de la vitesse de survenue de ces événements (tempo). La reconnaissance d'événements symboliques à partir du signal audio continu se fait en fonction de la certitude de ces deux agents.

7.4 Un accompagnement réactif et temporisé

Puisque *Antescofo* permet de connaître en temps réel la position du jeu dans une partition de musique et le tempo suivi par les instrumentistes humains, il devient possible de lancer des actions correspondant à la réalisation d'une partie instrumentale électronique, en réaction au jeu des instrumentistes humains et en accord avec eux, tout en respectant la temporalité musicale décrite pour ces actions.

Pour cela, il faut permettre d'exprimer coordination et la synchronisation d'actions de nature variée : synthétiser du son, le filtrer, le transformer, le spatialiser... Cette coordination est l'objet de l'écriture musicale qui organise les matériaux sonores dans le temps et implique des événements de nature temporelle différente et hétérogène, allant par exemple de la

microstructure de la texture sonore à la macrostructure de l'architecture des différentes parties d'un opéra. La gestion de ces différentes temporalités représente un défi majeur pour l'informatique musicale, à la fois lors de l'écriture et lors de la performance en temps réel.

Antescofo offre un langage de coordination réactive synchrone assez similaire aux langages dédiés à la programmation des systèmes temps réels (comme la commande de vol des Airbus par exemple). Bien sûr, dans la machine, l'action est réalisée après la survenue de l'événement, mais il est possible de maîtriser ce décalage afin qu'il devienne négligeable. Dans le cas de la musique, le temps de réaction doit typiquement être inférieur à 15ms afin de rester imperceptible. Mais *Antescofo* est aussi un langage « temporisé », où il est possible d'attendre l'écoulement d'un délai défini en termes de temps physique, de temps relatif (à partir du tempo des musiciens humains) ou d'un temps défini par un tempo arbitraire calculé. Tout en gérant les différentes temporalités des calculs, le langage synchrone musical d'*Antescofo* permet d'organiser la temporalité de chaque action dans une partition globale, à l'image de l'écriture musicale pratiquée dans le monde instrumental.

8 En guise de conclusion provisoire

Le temps est une ressource et l'informaticien l'organise dans ses langages autour des figures de la succession, de la simultanéité et de la durée. Cette grille d'analyse reste cependant limitée. D'abord parce que ces trois notions n'épuisent pas les formes du temps en informatique — logique ou réel, événementiel ou continu, linéaire ou ramifié… — pas plus que ses usages comme par exemple la production de temps dans le temps et à temps en informatique musicale. Ensuite parce que si le temps est une ressource pour les systèmes informatiques, ceux-ci doivent de plus en plus interagir avec nous et notre temps vécu. Le calcul du temps devient alors un processus ouvert qui doit nous saisir sur le vif. Et l'informaticien se confronte alors à d'autres dimensions du temps comme le mouvement, la mémoire, l'attente, le passage, l'anticipation, le devenir…

Le projet *Antescofo* montre que ces notions ne sont pas hors de portée, ou du moins, qu'il est fertile de réfléchir dessus ; dans une certaine mesure, il est possible d'accorder le temps de l'homme et celui de la machine. Comme jamais auparavant, *Antescofo* permet de gérer la superposition de temps hétérogènes tout en permettant de concevoir de la musique sous une forme qui reste assez proche de l'écriture traditionnelle. L'effectivité de l'approche suivie a permis de valider les modèles cognitifs utilisés, comme par exemple les modèles de suivi humain de tempo. Pour autant, le projet n'en est qu'à ses débuts : l'écriture musicale, avec toute sa richesse d'expression et sa réalisation scénique, pose toujours des défis irrésolus ; la diversité des interactions musicales échappe toujours aux interactions homme-machine ; le son, et sa perception, sont toujours des continents largement inexplorés. On voit donc que le dialogue que l'informatique entretient avec la musique, dialogue ambigu, dialogue fertile, n'est pas près de cesser.

Remerciements

Antescofo a pu être développé dans l'environnement accueillant de l'IRCAM grâce à une collaboration fertile avec plusieurs compositeurs dont Marco Stroppa, Philippe Manoury, Jonathan Harvey, Gilbert Nouno et Pierre Boulez. Je voudrais remercier vivement Arshia Cont et Gérard Assayag, ainsi que tous les membres de l'équipe RepMus et du projet MuTant dans lesquels s'insère ce travail, pour m'avoir accueilli à l'IRCAM et pour les nombreuses discussions qui ont permis cette présentation. En particulier, Julia Blondeau et José Echeveste méritent une mention spéciale pour leur disponibilité et leur aide précieuse.

> Ainsi s'agit-il moins de compiler des savoirs que des affects — empreintes fugitives de vécu, de ressenti. Paroles de pure résonance, sans volonté normative ou démonstrative, et sans l'ostentatoire exhibition de « boîtes à outils » (…) pour échapper au morne soliloque.
>
> [Farabet, 2011]

9 Bibliographie

[Boucourechliev, 1997] André Boucourechliev. Propos recueillis par Costin Cazaban et Alain Lompach dans *Le Monde*, 15 novembre 1997 (à propos de l'imprévisible et de la mobilité dans ses compositions).

[Boulez, 1963] Pierre Boulez. *Penser la musique aujourd'hui.* éd. Gauthier (1963).

[Desain, 1992] Peter Desain and Henkjan Honing. *Music, Mind, and Machine: Studies in Computer Music, Music Cognition, and Artificial Intelligence*. Thesis Publication. Amsterdam (1992).

[Farabet, 2011] René Farabet. *Théâtre d'ondes, théatre d'ombres*, Champ Social Edition, (2011).

[Frege, 1892] Gottlob Frege : « Sens et dénotation » in *Écrits logiques et philosophiques*, pp. 102-126 (1892). Traduction française de Claude Imbert, Seuil (1971).

[Grisey, 1987] Gérard grisey, « *Tempus ex Machina:* A composer's reflection on musical time », *Contemporary Music Review,* Vol. 2, pp. 239-275 (1987)

[Large, 1999] Edward Large and Mari Riess Jones. « The dynamics of attending: How we track time varying events » *Psychological Review*, 106 (1), 119-159 (1999).

[Malherbe, 2009] Claudy Malherbe, « En blanc et noir: l'espace musical contemporain, altérité et cohérence » in *Musurgia*, vol. 3-4 (2009).

[Manoury, 2012] Philippe Manoury. *La musique du temps réel.* Entretiens avec Omer Corlaix et Jean-Guillaume Lebrun. Editions M. F. (2012).

[Maudlin, 2002] Tim Maudlin. « Remarks on the Passing of Time » in *Proceedings of the Aristotelian Society,* Vol. 102, pp. 259-274 (2002).

[Mazzola, 1994] Guerino Mazzola and Oliver Zahorka. « Tempo curves revisited: Hierarchies of performance fields. » *Computer Music Journal* 18.1 pp. 40-52, (1994).

[McTaggart, 1908] John Mc Taggart : « The Unreality of Time », *Mind*, n°17 : 456-473 (1908). Accessible à l'adresse http://www.ditext.com/mctaggart/time.html

[Manoury, 2012] Philippe Manoury : *La musique du temps réel.* Editions Musica falsa (2012).

[Price, 1997] Huw Price. *Time's Arrow and Archimedes' Point: New Directions for the Physics of Time*, Oxford University Press (1997).

[Queneau, 1967] Queneau, Raymond. « Un conte à votre façon » publié dans Le Nouvel Observateur. Repris dans *Oulipo. La littérature potentielle (Créations Re-créations Récreéations).* Paris: Gallimard, coll. «Folio|Essais», pp.273-276 (1973).

[Sainte-Marie, 2008] Maxime de Sainte-Marie. « Les horloges sympathiques : L'organisation sociale au rythme de la syntonisation » in *Les cahiers du LANCI*, Université du Québac à Laval (2008). Accessible à l'adresse http://www.lanci.uqam.ca/

On trouvera une vidéo montrant la synchronisation de métronomes par un phénomène de sympathie à l'adresse http://youtu.be/W1TMZASCR-I

[Sorensen, 2010] Andrew Sorensen, and Henry Gardner. « Programming with time: cyber-physical programming with Impromptu ». *ACM Sigplan Notices*, vol. 45. No. 10. ACM (2010).

[Vila, 2005] Lluis Vila. « Formal Theories of Time and Temporal Incidence » in *Handbook of Temporal Reasoning in Artificial Intelligence,* Elsevier (2005)

[Zimmermann, 1957] Bernd Alois Zimmermann. « Intervalle et temps » in *Frankfurter Allgemeine Zeitung*, 23 mai 1957. Traduction française in *Contrechamps*, n°5, novembre p. 33 (1985).