Vers un modèle du raisonnement dans les langages à objets

François Pachet
LAFORIA- Institut Blaise Pascal, Boite 169
4, Place Jussieu, 75252 Paris, Cedex 05
Tél: (33) 1 44277004
Fax: (33) 1 44277000
E-mail : pachet@laforia.ibp.fr

Résumé
Ce papier est une proposition de modèle du raisonnement dans les langages hybrides, intégrant un langage de programmation par objets avec des règles de production. Nous proposons de différencier entre deux catégories d'objets, les objet du monde *perçu* et ceux du monde *conçu*. Le raisonnement est alors considéré comme un processus qui crée ou modifie le monde conçu à partir d'observations du monde perçu. Nous nous appuyons sur un exemple, l'analyse harmonique de grilles d'accords de Jazz. Dans ce système, les objets perçus sont les objets graphiquement représentables (les notes, les accords, et dans une moindre mesure les intervalles). Les objets conçus sont les tonalités, les degrés, les formes harmoniques identifiées par l'expert. Le raisonnement est représenté par deux types de règles : des règles dites de *reconnaissance*, et des règles de *remplissage*. Ces deux types de règles sont réinterprétés dans notre cadre, pour montrer qu'ils respectent notre distinction. Nous concluons sur la validité de notre proposition comme embryon de méthode pour la construction de bases de connaissances dans les systèmes hybrides.

Mots-clés
Représentation de connaissances, programmation par objets, analyse harmonique, modélisation du raisonnement.

Introduction

Le papier de Minsky [MINS75] est aujourd'hui considéré comme à la source de la plupart des langages de représentation de connaissances. Bien que ne contenant pas de spécification précise de la notion de frame, son interprétation par le triplet frame-slot-value s'est imposée de facto dans les langages de représentation de connaissances, du précurseur KRL [BOBR& WINO85] jusqu'à des langages plus récents comme CycL [LENA&GUHA 90]. Dans ces langages, la cellule de base pour la représentation, le frame, est une entité dont la structure est décrite par un ensemble de slots, ayant chacun un certain nombre de facettes. Parmi les facettes les plus courantes, on trouve la *valeur* du slot, ainsi que des informations comme le *type*, *l'arité*, ainsi que des procédures d'accès permettant d'effectuer certains calculs lors de la modification de la valeur d'un slot. Le système LOOPS [BOBR&STEF 83] est un des premiers systèmes avoir exploré la combinaison de représentation par frames avec divers mécanismes d'inférence, dont les règles de production, et est lui-même à l'origine d'une série de systèmes de représentation appelés *systèmes hybrides*. Cet effort d'intégration a suscité une quantité gigantesque de recherche aussi bien théorique qu'appliquée (Cf. par ex. [BRAC&LEVE

85]). Nous nous intéressons ici à un aspect qui semble avoir été pour l'instant négligé, à savoir la question de la *nature du raisonnement* dans un système hybride. Bien que les langages de représentation hybrides proposent le plus souvent plusieurs *modes* d'inférences, une des seules définitions de la nature du raisonnement dans de tels systèmes est celle des auteurs de KRL [BOBR& WINO85], qui le définissent essentiellement comme un processus de "pattern matching". Dans cette optique, raisonner repose sur le cycle suivant : 1) Trouver un ensemble de frames qui vérifie un ensemble de conditions ou contraintes, puis 2) modifier cet ensemble de frames. Ce cycle d'inférence est effectué jusqu'à ce qu'un certain but soit atteint. "Trouver l'ensemble de frames", la première étape du cycle, est réalisé par l'opération de pattern-matching, sur laquelle la plupart des efforts se sont concentrés. C'est dans cet esprit que, par exemple [FIKE&KEHL 85] affirment que le mécanisme principal de raisonnement offert par un langage de frames repose sur le "remplissage de valeurs de slots", par exemple pour faire de la classification (qui s'occupe de remplir le slot "instanceOf").

Dans le contexte de la programmation par règles de production, ceci revient à dire que la nature du raisonnement n'aurait été considérée "qu'à moitié", ne s'occupant que du problème des parties gauches des règles (le pattern-matching). Le statut des membres droits des règles, qui représentent l'action du raisonnement, a reçu moins d'attention. Ainsi, le rôle du raisonnement, en tant que processus, est-il simplement de modifier - dans un sens volontairement large - un ensemble de frames (Cf. Figure 1).

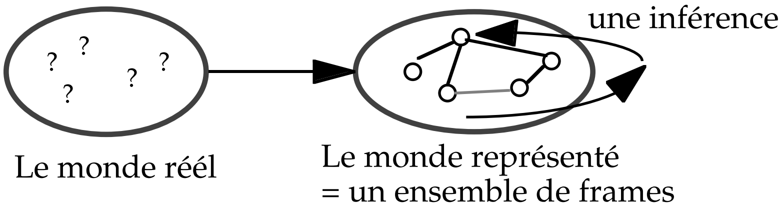

Figure 1. Le diagramme du raisonnement dans un univers de frames.

1. Deux rôles pour les slots

Cette imprécision du modèle du raisonnement est pour une part liée à la difficulté de donner une sémantique précise à la notion de slot. A ce titre, "remplir un slot" peut représenter des actions d'au moins deux types fort différents. Dans un système de *diagnostic,* par exemple, où l'on effectue un raisonnement à partir de données qui sont recueillies par des capteurs, la modification d'un slot peut représenter un *changement effectif du monde* sur lequel le système raisonne. Par exemple, la pression artérielle d'un patient recueillie par un capteur peut être représentée par le slot "pressionArterielle" du frame Patient202. Les informations fournies par le capteur se traduiront par des modifications des slots correspondants. D'autre part, le remplissage d'un slot peut représenter le *résultat d'une inférence*, ou de tout autre processus purement interne au système, qui sera, lui, totalement déconnecté du monde sur lequel le système raisonne. Ce serait le cas par exemple du slot "riskOf Infarctus" du même frame, qui sera renseigné à la suite d'un raisonnement (d'une classification, de l'application d'un ensemble de règles, de démons, etc.). En d'autres termes, les slots peuvent représenter soit le monde tel qu'il est *perçu*, soit le monde tel qu'il est *conçu.*

Nous proposons ici un raffinement du modèle de base du raisonnement (Fig. 1), prenant explicitement en compte la différence entre monde perçu et monde conçu. Cependant, au lieu d'essayer de distinguer plusieurs types de slots, comme l'a fait par exemple [WOOD 85], notre approche consiste à distinguer deux catégories *d'objets* : les *objets perçus* et les *objets conçus*. De plus nous proposons un cadre de représentation de ces objets reposant sur une double *simulation*, et utilisant pleinement les paradigmes de la programmation par objets. Cette distinction nous permet de proposer une définition du statut du raisonnement, comme un processus modifiant les objets conçus à partir d'observations de configurations des objets perçus. Dans notre cadre, ce processus est représenté par des règles de production.

Nous illustrons notre proposition par un exemple tiré d'une base de connaissances en musique tonale [PACH 94] : l'analyse de grilles d'accords de Jazz. Bien que ce domaine soit inhabituel pour la communauté des chercheurs en IA, nous l'avons choisi car il nous semble particulièrement représentatif de notre architecture. Aucune connaissance particulière du domaine n'est requise pour suivre notre argumentation. Le reste du papier est organisé comme suit : le paragraphe 2 décrit le problème, le paragraphe 3 notre cadre de représentation. Les paragraphes 4 et 5 montrent comment la simulation par objets permet de représenter les mondes perçus et conçus correspondant au domaine. Le paragraphe 6 décrit le processus de raisonnement représenté par des règles de production, qui effectuent un lien entre les deux mondes. Le dernier paragraphe propose un diagramme plus précis pour le statut du raisonnement, et discute l'utilisation de notre approche comme méthodologie possible pour la construction de bases de connaissances par systèmes hybrides.

1. Exemple : l'analyse harmonique de grilles d'accords de Jazz

L'analyse harmonique est un terrain idéal pour tester les techniques de représentation de connaissances inférentielles. Des connaissances complexes sont mises en jeu, beaucoup d'exemples bien documentés sont accessibles. Nous nous intéressons en particulier à l'analyse de grilles d'accords de jazz "standard", comme celles que l'on trouve dans les fameux recueils [REAL 81] ou [FAKE 86]. Le problème consiste, pour une séquence d'accord donnée comme celle de la figure 2 (nous ne prenons pas en compte les mélodies ici), à trouver une interprétation correcte de l'accord, c'est à dire une tonalité dans laquelle l'accord doit s'analyser (Cf. figure 3). Hors de tout contexte, chaque accord peut être analysé dans un certain nombre de tonalité (entre 5 et 10 en moyenne). Tout le problème est de trouver la bonne, en fonction des tonalités des accords voisins, et de connaissances sur l'harmonie en général. Une fois terminée, l'analyse permet au musicien d'identifier les gammes à partir lesquelles il pourra improviser librement, sans rompre la cohérence harmonique de la grille (mais ceci est une autre histoire).
Le problème de l'analyse musicale n'est pas nouveau au sein de la communauté IA & musique. Deux travaux sont particulièrement représentatifs du type d'effort pour représenter le raisonnement musical : [WINO 93] propose d'utiliser un formalisme de grammaires étendues pour représenter le raisonnement harmonique; [STEE 83] s'est intéressé à trouver un ensemble de règles génératives permettant de représenter toutes les variations possibles autour du schéma de la grille de blues de 12 mesures. Dans ces deux cas, ce qui est visé est plus l'établissement d'un *modèle formel du domaine*, qui capture une certain essence du corpus musical modélisé, s'inspirant largement des travaux sur les grammaires musicales [LERD&JACK 83], que d'un véritable mécanisme rendant compte de l'*activité de raisonnement* elle-même. En outre, si l'approche de [STEE 84] est fort séduisante, elle n'est pas directement implémentable, car elle fait abondamment usage de règles *context-dependent*.
Notre but ici est différent : il s'agit de simuler le raisonnement humain, d'une part pour construire un système qui résolve effectivement le problème de manière satisfaisante sur les grilles des corpus cités, et d'autre part pour tenter de mieux comprendre ce processus.

2. Deux catégories d'objets

En concevant ce système d'analyse, nous avons identifié deux catégories d'objets. D'une part des objets qui représentent des entités explicitement *perçues par l'expert* : les notes (qui ne sont pas forcément des entités simples a représenter !), les altérations (dièses, bémols et autres signes diacritiques), les intervalles (cette notion est déjà plus abstraite, car non représentée explicitement, néanmoins nous la considérons comme faisant partie du décor de base de l'expert, nous y reviendrons), les accords (explicitement représentés par leurs nom), ainsi que la grille elle-même, sa taille, ses mesures, etc. Il est important de noter que cette catégorie ne correspond pas forcément à l'ensemble des objets directement visibles (ou tangibles). Le cas des intervalles est en ce sens typique: on ne peut "voir" un intervalle que si l'on sait déjà beaucoup de choses sur la musique tonale (de même qu'un expert aux échecs "voit" des configurations de pièces qui sont invisibles aux yeux des novices). Ces objets représentent la perception de l'expert "avant" la résolution du problème (ici l'analyse). Cette perception, si l'on ne sait pas modéliser son processus, nécessite déjà des connaissances (que l'on ne représente pas ici!).


Figure 2. Un exemple de grille à analyser.

D'autre part des objets représentent un modèle nécessaire au raisonnement, qui est, lui, totalement abstrait, et purement de la paternité de l'expert. On trouvera ici des objets comme les *analyses harmoniques*, les *tonalités* (et, à un certain degré, les gammes), les *degrés* dans les tonalités, ainsi que les diverses formes musicales et structures de haut niveau comme les *anatoles*, les *deux-cinq-un*, les *résolutions*, et autres marches harmoniques, qui représentent certaines configurations d'accord particulièrement significatives. Bien que le but ultime de l'analyse puisse se représenter graphiquement (Cf. figure 3), ces objets ne sont pas directement visibles par l'expert, mais plutôt imaginés, construits mentalement au cours du raisonnement. Ce n'est qu'en fin de raisonnement qu'une partie d'entre eux acquièrent une représentation graphique, qui n'a alors qu'un rôle accessoire.

2. Notre cadre de représentation

1. Deux mondes en action

L'idée principale que nous voulons défendre ici est que le processus de raisonnement fait intervenir une *simulation*, - plutôt qu'une représentation - du monde réel. Plus précisément, le raisonnement fait intervenir deux simulations simultanément : une simulation du monde perçu, qui rend compte de la perception du domaine d'étude, de ses entités, et des relations entre celles-ci, et 2) une simulation du monde conçu qui inclut les objets abstraits, les formes et structures dont nous avons besoin pour mener à bien le raisonnement. Le raisonnement est alors vu comme un processus qui *observe* le monde perçu, pour *animer* le monde conçu.

*Fi*gure 3. La grille (partiellement) analysée.

Dans notre exemple musical, le monde perçu est composé d'objets comme les notes, altérations, accords, comme nous l'avons proposé plus haut. Nous ne percevons pas ces objets uniquement des structures fixes et inertes. Nous savons aussi leur associer des opérations, des comportements, nous savons comment ils "réagissent" à certaines opérations prototypiques. Par exemple, nous savons comment les altérations (dièse et bémol) se combinent entre elles (pour s'annuler en l'occurrence), comment les notes et les accords se transposent, comment l'on peut déduire (calculer) la liste des notes d'un accord, en ne connaissant que son nom (et inversement), etc. Il est important de noter ici qu'il s'agit plus que de représenter des structures ou des propriétés - au sens large. Il s'agit bien ici d'opérations, de calculs, que l'étudiant en musique apprend pendant plusieurs années, et qui participent pleinement au raisonnement musical en général.

Il en est de même pour le monde conçu. Ce monde est lui aussi composé entités - quoi que de nature différente - qui ont leurs structures, relations, et comportement. Par exemple, nous savons comment les tonalités se comportent les unes avec les autres, nous savons que certains accords sont substituables dans un certain contexte, que comment les gammes peuvent engendrer des accords avec certains degrés, etc.

2. Programmation par objets et simulation

Les remarques précédentes nous conduisent naturellement à utiliser la programmation par objets pour implémenter notre modèle. En effet, la PPO propose de modéliser le monde par un ensemble de classes définissant à la fois une structure (statique) et un comportement (des opérations) pour leurs instances. Des critiques virulentes ont été adressées contre l'utilisation de ces techniques pour faire de la *représentation* de connaissances au sens traditionnel du terme (Cf. par exemple [PATE 90]). Nous pensons que ces critiques sont en général tout à fait justifiées, notamment en ce qui concerne le mécanisme d'héritage de la PPO, qui ne permet certainement pas de classifier des catégories, du moins au sens ensembliste du terme. Cependant, dans l'optique d'un modèle à base de simulations, le choix de la PPO s'impose, notamment par la place que celle-ci donne aux opérations (les méthodes). Ce choix est naturellement justifié par les origines mêmes de la PPO (Simula, un langage initialement destiné à résoudre des problèmes de simulation). Comme le souligne [BEZI 87], la simulation fut un prétexte pour découvrir les concepts de la programmation par objets. Aujourd'hui, ces concepts ont fait leurs preuves dans des domaines qui ne relèvent plus à proprement parler de la simulation. En ce sens, la simulation est devenue un *moyen* plutôt qu'une *fin*, comme le disent [PUGH&al 87] : "The fundamental philosophy of object-oriented programming is that **all** programming can be viewed as simulation".

3. Représentation du monde perçu

La programmation par objets a depuis longtemps été utilisée pour la construction de systèmes musicaux à base de connaissances. [SMAI&WIGG 90] furent les premiers à reconnaître l'intérêt des types abstraits pour représenter les "constituants" utilisés lors de processus d'analyse. Dans un autre registre, les systèmes MODE [POPE 91] et Kyma [SCAL 87] contiennent de nombreuses représentations (Smalltalk) d'objets sonores complexes, utilisés dans le contexte de la synthèse ou de la composition assistée par ordinateur.
Nous construisons dans cette lignée un modèle de notre monde perçu, à l'aide des techniques de la PPO. Ce modèle, appelé MusES, contient les définitions de la plupart des notions d'harmonie de base, sous forme de classes et d'objets Smalltalk. Ces classes et objets ont été conçus (sic) pour pouvoir résoudre naturellement la plupart des (petits) problèmes de solfège ou d'harmonie de base (Cf. pour plus de détails [PACH 94]). Par manque de place, nous allons simplement donner ici un aperçu de la manière dont MusES résout le problème de la notation enharmonique, problème typique de représentation musicale.

1. La notation enharmonique

Le problème de la notation enharmonique consiste simplement à trouver une représentation des notes (plus exactement des intraduisibles "pitch-classes", e.g. les notes indépendantes de l'octave) permettant de différencier entre des notes de noms différents, mais ayant la même hauteur (en musique tempérée), comme Do# et Ré*b*.
"Trouver une bonne représentation" consiste ici à trouver une représentation à partir de laquelle les problèmes d'harmonie et de solfège soient simples à résoudre, comme : calculer un intervalle de manière correcte (la quinte diminuée de Do# est Sol*b*, et non Fa#), calculer le nombre de demi-tons entre eux notes, calculer le nom d'un intervalle entre deux notes, ou représenter les propriétés algébriques des altérations (du type # + *b* = bécarre). La possibilité de différencier entre notes enharmoniques (comme Do# et Réb) est vitale dans le cadre de systèmes d'analyse. Elle est aussi vitale que l'orthographe dans les systèmes de compréhension automatique de texte: chaque nom de note porte en effet une certaine *intention* harmonique, qu'il faut savoir utiliser dans la représentation.
Si ce problème parait simple, il n'a pourtant jamais été abordé de manière complète dans les systèmes de représentation existants. [WINO 93] reconnaît l'importance du problème mais propose une solution ad hoc. Plus proche de nos préoccupations, [STEE 84] ignore le problème et ne s'intéresse qu'aux propriétés des séquences d'accords simplement déduites de l'ordonnancement de leurs éléments. Nous donnons ici un aperçu dont le problème peut être résolu, de manière élégante, en utilisant simplement les mécanismes de la programmation par objets.

2. Des altérations vues comme des méthodes polymorphes

La solution au problème consiste à 1) introduire plusieurs *classes* de notes (entendre de "pitch-classes") et 2) à considérer les altérations comme des méthodes polymorphes pour ces classes. Plus précisément, nous introduisons les 5 classes (au sens de la PPO) : NoteNaturelle, NoteDiese, NoteBemol, NoteDoubleDiese, NoteDoubleBemol (avec une classe abstraite intermédiaire NoteAlteree). Nous considérons ensuite les altérations comme des méthodes, qui seront définies et redéfinies dans chaque classe de note, pour prendre en compte les propriétés des altérations. Ainsi, les altérations sont donc vues comme des fonctions, faisant passer d'une classe de note à une autre. Par exemple, la méthode dièse de la classe NoteNaturelle, fera passer d'une NoteNaturelle à une NoteDiese (en d'autre terme, x# = dièse(x)). La méthode dièse de la classe NoteBemol fera, elle, passer d'une NoteBemol à une NoteNaturelle, (xb# = dièse(bemol(x)=x), etc. On définit alors les structures minimales pour les classes de notes permettant d'implémenter les opérations en question, ainsi que tous les calculs usuels sur les notes (Cf. [PACH 94] pour plus de détails).

3. Des notes aux intervalles, gammes et accords

Une fois la représentation des les notes établie, celle des objets plus élaborés est immédiate. Nous définissons la notions d'Intervalle, comme un objet capable essentiellement de calculer des notes extrémité. Puis les notions de Gamme et d'Accord sont introduites de la même manière, ayant chacun leur structure, leur comportement, et éventuellement (pour les gammes par exemple) leurs classifications propres. Le résultat est un système capable de répondre à la plupart des questions d'harmonie de base, par simulation. Par exemple, on demande à une gamme de se transposer en lui envoyant le message correspondant, à un intervalle de calculer la note de départ quand celle d'arrivée est connue, ou bien à un accord de rendre la liste de ses notes à partir de son nom, etc. Le monde perçu est alors constitué d'un ensemble d'objets ayant chacun leur comportement, et représentant l'essence des connaissances de base en harmonie. De par la nature explicitement procédurale de cette représentation, ce que nous représentons ici est bien une forme de "connaissance compilée dans les objets", simulant ainsi un processus de perception, ne mettant pas en oeuvre de processus de raisonnement explicite.

4. Analyses, tonalités, et le monde conçu

Le monde *conçu* est composé des objets relatifs à l'analyse. Parmi ces objets on trouve: les analyses elles-mêmes, les tonalités possibles des différents accords, les structures et formes harmoniques les plus connues. Comme nous l'avons souligné, ces objets ne sont pas directement perçus par l'expert, ou du moins, cette perception nécessite un certain raisonnement, qu'il s'agit justement de représenter explicitement. Nous représentons aussi ces objets avec le même formalisme, c'est a dire par une simulation, à l'aide de la programmation par objets. A titre d'exemple, une Analyse est un objet ayant comme structure un *degré* et une *gamme*. On peut alors demander à deux analyses de se comparer, à une deux listes d'analyses d'extraire leurs analyses communes, ou bien encore à une forme musicale (comme un deux-cinq-un) de calculer les analyses respectives de ses composants, etc.

Ces choix de représentation sont validés par la réalisation du système lui-même, et par sa réutilisation au sein de plusieurs systèmes à base de connaissances, notamment un système de génération automatique d'harmonisations, utilisant un mécanisme de satisfaction de contraintes, et un système de génération d'improvisations [RAMA&GANA 94]) (Cf. [PACH 94b]). Le succès du système MusES nous permet de valider notre approche de représentation par simulation du domaine d'étude (ici, le monde perçu). Bien sûr, nous n'avons pas *tout* représenté. Nous décrivons maintenant les règles du raisonnement, et montrons comment elles s'articulent en fonction des deux mondes décrits.

5. Les règles du processus de raisonnement

Notre approche, basée sur des interviews de musiciens et notre propre expérience, consiste à distinguer deux phases dans le raisonnement d'analyse:

- Une première phase de "reconnaissance de formes" pendant laquelle des formes, ou configurations d'objets, sont identifiées dans la grille. Par exemple, les deux-cinq-un se reconnaissent par une séquence de deux accords vérifiant certaines conditions (l'un est la quinte de l'autre, le premier est mineur, le deuxième septième, etc). Les objets à identifier sont des accords (donc des objets du monde perçu). Les conditions à remplir pour identifier les formes sont toutes exprimées par des requêtes (des messages) à ces objets (Cf. la règle `deuxCinqMajeur`). Cette catégorie comprend une vingtaine de règles, permettant de repérer les structures musicales les plus connues.

- Une deuxième phase de "remplissage". Cette étape consiste à 1) trouver les tonalités des formes reconnues dans la première étape, et 2) regrouper, autant que faire se peut, les accords isolés avec les formes reconnues. Ce regroupement se fait en utilisant des règles de regroupement, ayant une certaine parenté avec les règles de "grouping" de [LERD&JACK 83]. Elles assurent un nombre minimal de modulations, en essayant d'étendre le plus possible les analyses des formes reconnues aux accords adjacents (C. la règle `analyseAccordSuivant`). Cette catégorie comprend une dizaine de règles.

Pour représenter ce raisonnement, nous avons clairement besoin d'un mécanisme d'inférence, qui n'est pas fourni en standard par les langages de programmation par objets, quels qu'ils soient. Nous utilisons ici le système NéOpus, une extension de Smalltalk par un mécanisme de règles de production d'ordre 1 en chaînage avant [PACH95]. En NéOpus, les parties condition et action des règles sont des expressions quelconques, utilisant des variables pour designer les objets quantifiés.
Voici deux règles de la base. La première reconnaît un "deux-cinq" en majeur, lorsque la configuration correspondante est trouvée. La deuxième règle analyse un accord isolé adjacent à une forme déjà reconnue, lorsque c'est possible (i.e. cet accord est analysable dans la tonalité de la forme en question) :

Une règle de reconnaissance de forme:

```
deuxCinqmajeur
PourTout c1 c2 instances d' Accord:
Si
 c1 estMineur.
 c2 = c1 suivant.
 c1 nAPasDe: #quinteDiminuee.
 c2 aUne: #septiemeMineure.
 c2 tonique = c1 tonique quarte.
Alors
DeuxCinq new tonalite:
 (c2 tonique quarte gammeMajeure);
 accords: (Array with: c1 with: c2)
```

Une règle de remplissage

```
analyseAccordSuivant
PourTout c1 c2 instances
d'Accord:
Si
 c1 estAnalyse.
 c2 = c1 suivant.
 c2 nonAnalyse.
 c2 nonAmbigu.
 c2 tonalitesPossibles includes:
    c1 analyse gamme.
Alors
 c2 analyseDans: c1 analyse gamme
```

Notons alors que, conformément à notre approche, 1) les parties conditions des règles sont exprimés uniquement par des messages envoyés aux objets perçus, et 2) les parties action des règles consistent à créer ou modifier des objets conçus (ici des structures musicales et des analyses). En particulier, le résultat essentiel de l'analyse est représenté par les règles de groupement. Plus précisément, l'action finale (représenté par la méthode `analyseDans:`) consiste à sélectionner la bonne analyse parmi les analyses possibles. Ce n'est donc pas l'objet `Accord` lui-même qui est modifié par cette action, mais la liste de ses objets "satellites" (des instances de la classe `Analyse`, Cf. Figure 5). Le lien entre l'accord et ses analyses est entièrement géré par les règles.

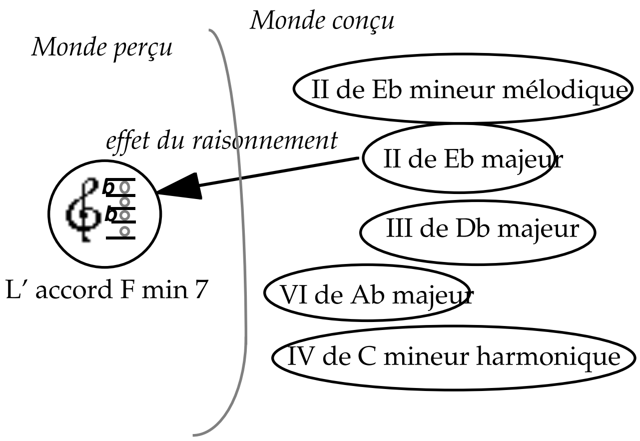

Figure 5. Un accord, ses analyses possibles, et la bonne.

6. Un cadre plus précis pour la représentation de connaissances

Cette base de connaissances n'est qu'une illustration, ici, de notre proposition de cadre général pour le raisonnement dans des langages hybrides (PPO + règles de production), qui fournit un diagramme un peu plus précis que le diagramme initial (Figure 1). Ce cadre est défini par les deux principes suivants:

- La distinction explicite entre eux catégories d'objets. D'une part les objets qui représentent une simulation de monde tel qu'il est perçu par l'expert, avant le raisonnement proprement dit. Cette perception nous l'avons vu, est elle-même un processus qui exige des connaissances (voire beaucoup de connaissances!). Nous n'essayons pas de modéliser ce processus (la perception), et nous plaçons délibérément après la perception.
- Un statut précis pour les règles de production servant à modéliser le raisonnement proprement dit. Leur rôle est en effet de construire le monde conçu, en laissant intact le monde perçu (qui lui, n'est modifiable que par l'intermédiaire de capteurs, ou tout autre "lien" avec le réel).

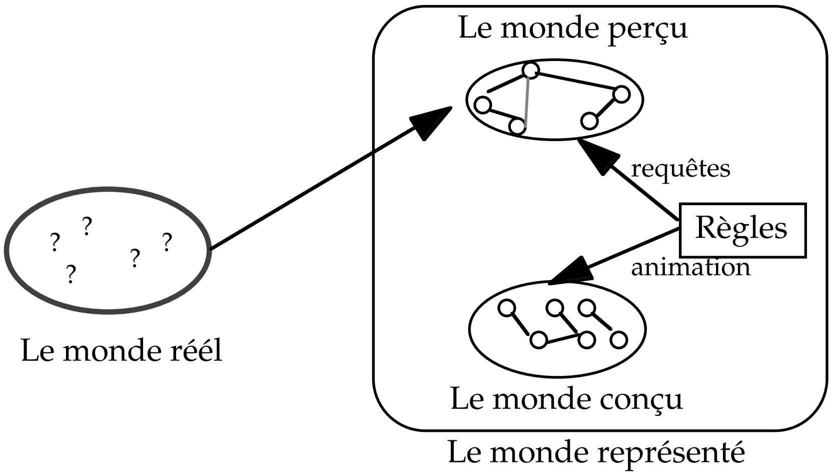

Figure 6. Le processus de raisonnement dans notre représentation par objets

Dans ce cadre, le raisonnement est donc vu comme une animation du monde conçu à partir d'observations du monde perçu (Cf. Figure 6).

Conclusion

Le système décrit permet effectivement d'analyse les grilles de Jazz les plus courantes, en faisant aussi bien qu'un expert sur les cas consensuels. Ce résultat en lui même mérite d'être mentionné, car nous ne connaissons pas de système comparable ayant d'aussi bonnes performances. Mais notre présentation est à prendre, ici, comme un *proposition de méthode*. En particulier, il manque à ce schéma des règles plus précises expliquant le "passage" du perçu au conçu, et, certainement une catégorisation plus fine des "effets" des règles, sur les deux mondes modélisés.
Cependant, cette proposition est le résultat d'analyses - non terminées - de plusieurs gros systèmes à base de connaissances utilisant le formalisme des objets (au sens de la PPO) et des règles de production. Le système NéoGanesh [DOJA&PACH 91] est une grosse base de connaissances réalisée en Smalltalk/NéOpus, qui contrôle des ventilateurs d'unités de soins intensifs en temps réel. Cette base de connaissances suit admirablement le schéma présenté ici. Dans ce système, le monde perçu (par les médecins) contient tous les objets décrivant les patients, leurs historiques, leurs symptômes, et toutes les données fournies par les capteurs (PCO2, pression sanguine, etc.). Le monde conçu contient les traitements (en cours ou proposés), les diagnostics, et les plans de soins établis par les médecins. Un traitement particulier du temps est intégré au mécanisme d'inférence, mais le processus de raisonnement respecte notre dichotomie: le monde perçu est modifié essentiellement par les capteurs, le monde conçu par les règles.
Un autre exemple de grosse base de connaissances dans le même contexte est le système ForreEnMat [LAUB 94], qui démontre des théorèmes en théorie des ensembles. Dans ce système, le monde perçu (par le mathématicien) est constitué des formules mathématiques et théorèmes à démontrer; le monde conçu est constitué des plans, stratégies, objets intermédiaires, lemmes, créés et modifiés au cours du raisonnement.
Dans les trois cas, la base de connaissance a été construite par essai-erreur, sans utiliser de "méthode" particulière, et encore moins l'embryon présenté ici. Cependant, un consensus a été atteint en ce qui concerne le besoin de distinguer plusieurs catégories d'*objets* (et non de liens, ou de slots), et pour donner un statut plus précis aux inférences dans de tels systèmes. Les efforts portent maintenant sur la construction d'un modèle plus formel du raisonnement dans notre univers, et une caractérisation des types de problèmes qu'elle permet de résoudre.

Références

[ATKI&LAUR 87] Atkinson, R. Laursen, J. Opus : A Smalltalk Production System. *Proc. d' OOPSLA'87*, pp. 377-387, (1987).
[BEZI 84] Bézivin, J. Simulation et langages orientés objet. *Bigre+Globule* n° 41, pp. 194-211, Novembre 1984.
[BEZI 87] Bézivin, J. Some experiments in Object-Oriented Simulation. *Proc. OOPSLA 87*, Orlando, 1987, pp. 394-405.
[BOBR&STEF 83] Bobrow, D. Stefik, M. The LOOPS Manual, Xerox Corporation (1983).
[BOBR&WINO 85] Bobrow, D. Winograd, T. *An Overview of KRL, a Knowledge Representation Language.* In [*Brachman&Levesque 85*], pp. 254-285 (ré-édition). (1985).
[BRAC&LEVE 85] Readings in Knowledge Representation, Ed. by R. Brachman and H. Levesque, Morgan Kaufmann, (1985).

[DOJA&PACH 92] Dojat, M. Pachet, F. NéoGanesh: an Extendable Knowledge-Based System for the Control of Mechanical Ventilation. *14 th Annual Int. Conf. of the IEEE Eng. in Medicine and Biology Society*, Oct. 29-Nov. 1st, Paris (1992).
[EBCI 92] Ebcioglu, K. An expert system for harmonizing chorales in the style of Bach. In *Understanding Music with A.I.* AAAI Press/ MIT Press, 1992. Ed. by Balaban M., Ebcioglu K., Laske O. (1992).
[GOLD&ROBS 89] Goldberg, A. Robson D. Smalltalk-80 : the language and its implementation. Addison-Wesley (1989).
[FAKE 86] The Fake book, Syosset, New York, (1986).
[FIKE&KEHL 85] Fikes, R. Kehler, T. The Role of Frame-Based Representations in Reasoning. *Communications of the ACM*, Sept. 1985, Vol. 28, N. 9, pp. 904-920, (1985).
[LAUB 94] Laublet, P. Hybrid Knowledge Representation and Theorem Proving in Mathematics. *Proc. of Conf. Artificial Intelligence in Mathematics*, pp. 181-196, Glasgow, April 1991. To appear in *Artificial Intelligence in Mathematics*, J.H. Johnson, S. McKee & A. Vella (Eds), Oxford University Press, (1994).
[LENA&GUHA 90] Lenat, D. Guha, R.V.Building Large Knowledge Bases. Representation and Inference in the Cyc Project. Addison-Wesley, (1990).
[LERD&JACK 83] Lerdhal, F. Jackendoff, R. A Generative Theory Of Tonal Music. Cambridge, MA, MIT Press, (1983).
[MINS 75] Minsky, M. A Framework for Representing Knowledge. In: P.H. Winston (Ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, (1975).
[PACH 94] Pachet, F. An Object-Oriented Representation of Pitch-Classes, Intervals, Scales and Chords. *JIM 94 - Journées d'informatique musicale*, Bordeaux, mars (1994).
[PACH 94b] Pachet, F. The MusES system : an environment for experimenting with knowledge representation techniques in tonal harmony. *Brazilian workshop on computer music*, Belo Horizonte, août (1994).
[PACH 95] Pachet F. On the embeddability of production rules in object-oriented languages. *Journal of Object-Oriented Programming*, 1995. A paraître.
[PATE 90] Patel-Schneider, Peter F. Practical, Object-Based Knowledge Representation for Knowledge-Based Systems. *Information Systems*, Vol. 15, N. 1, pp. 9-19, (1990).
[POPE 91] Pope, Steven. The Well-Tempered Object. MIT Press, (1991).
[PUGH&AL 87] Pugh, J.-R., LaLonde, W. R. Thomas, D. A. Introducing Object-Oriented Programming into the Computer Science Curriculum. *Proc. OOPSLA'87*, Orlando.
[RAMA&GANA 94] Ramalho G., Ganascia J.-G. Simulating Creativity in Jazz Performance. *Proc. AAAI'94*. Seattle, Août 1994. A paraître.
[REAL 81] The real book. The Real Book Press, (1981).
[SCAL 87] Scaletti, C. Kyma: An Object-oriented Language for Music Composition. in *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, 1987.
[SMAI&WIGG 90] Smaill, Alan. Wiggins, Geraint. Hierarchical music representation for composition and analysis. In Int. Coll. *Music and Computer Assistance*, pp. 261-279, Marseille, France, 3-6 oct. (1990).
[STEE 84] Steedman, M.J. A Generative Grammar for Jazz Chord Sequences. *Music Perception*, Fall 1984, Vol. n° 2, N° 1, pp. 52-77, (1984).
[WINO 93] Winograd, Terry. Linguistics and the Computer Analysis of Tonal Harmony. In *Machines Models of Music*, Edited by S. M. Schwanauer and D.A. Levitt, MIT Press, (1993).
[WOOD 85] Woods, W. What's in a link: Foundations for Semantic Networks. In [*Brachman & Levesque 85*], pp. 217-241 (ré-édition) (1985).