Using Rules to support Case-Base Reasoning for harmonizing melodies

J. Sabater, J. L. Arcos, R. López de Mántaras

Artificial Intelligence Research Institute (IIIA)

Spanish National Research Council (CSIC)

Campus UAB, 08193 Bellaterra, Spain

{jsabater,arcos,mantaras}@iiia.csic.es

Abstract

This paper deals with the problem of harmonizing melodies based on a multimodal reasoning approach where general knowledge about harmonization, represented by rules, supports concrete knowledge represented by cases. Rules are applied when the cases cannot provide a solution. The combination of these two reasoning methods has proven to be useful and in general can be useful in domains where it is difficult to find enough cases and is not suitable to work only with general rules.

Introduction

The problem of harmonizing melodies has been usually approached using rule-based systems. In fact, this approach seems to be the most natural way because, traditionally, the art of harmonizing in music schools has been taught by means of rules. But if we analyze the phenomenon of harmony we will see that these rules are only a method (usually imperfect) to reflect the organization and structure inside a musical composition. In other words, the rules don't make the music, is the music which makes the rules. Then, why to rely only on a set of imperfect rules when we can also have the source of this rules, i.e. the compositions?

This idea sounds very nice and CBR methods seems to be the best form to implement it but, as we saw in our first attempts, one needs a large number of cases for the system to work properly. However it is cumbersome to obtain and represent as cases a large number of harmonized melodies. To solve this problem we have opted for a hybrid system that uses rules and CBR working together.

The input of our system, called GYMEL, is a single musical phrase and the output is the same musical phrase with a set of chord sequences. Every chord sequence is an accompaniment for that musical phrase. We have tested our system using popular songs but the idea is valid for other kind of music as long as one has a set of examples and some general harmony rules for that kind of music.

GYMEL, an overview

Figure 1 shows the general structure of GYMEL (Sabater 1997). There are two main modules: a CBR module and a rule-based module. CBR module is the first module used by GYMEL in order to try to solve the problem using CBR methods. When the CBR module cannot find a solution for a given point, GYMEL tries to apply some rules using the rule-based module. The rule system proposes a solution and the CBR module tries to continue from that point.

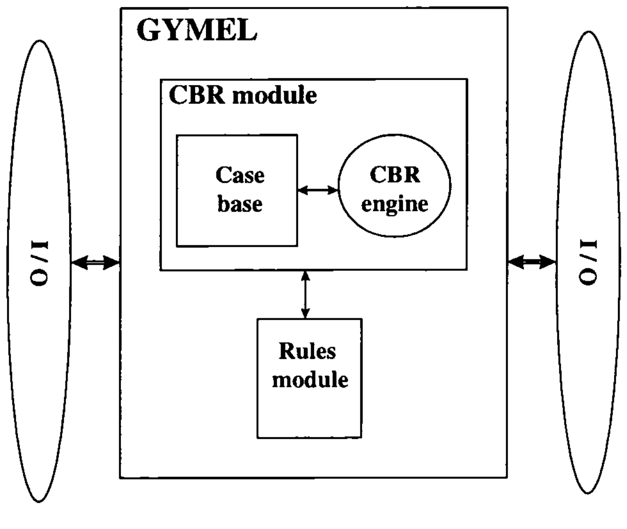


Figure 1. General structure of GYMEL.

GYMEL has been developed using NOOS (Arcos 1997), a reflective object-centered representation language designed to support knowledge modeling of problem solving and learning.

The next section describes the main features of Noos. Then, the paper describes the CBR module and the Rule-based module, and finally we present the results and give some conclusions.

GYMEL and the NOOS language

NOOS is based on feature terms (Plaza 1995). Feature terms are record-like data structures embodying a collection of features.

Modeling a problem in NOOS requires the specification of three different types of knowledge: **domain knowledge**, **problem solving knowledge**, and **metalevel knowledge**.

Domain knowledge specifies a set of concepts, a set of relations among concepts, and problem data that are relevant for an application. Concepts and relations define the domain ontology of an application. For instance, the domain ontology of GYMEL is composed by concepts such as notes, chords, tonalities, etc.

Problem data, described using the domain ontology, define specific situations (specific problems) that have to be solved. For instance, specific musical phrases to be harmonized.

Problem solving knowledge specifies the set of tasks to be solved in an application. For instance, the main task of GYMEL is to infer a sequence of chords for a given melody. Methods model the ways to solve tasks. Methods can be elementary or can be decomposed into subtasks. These new (sub) tasks may be achieved by other methods. A method defines an execution order of subtasks and a specific combination of the results of the subtasks in order to solve the task it performs. For a given task, there may be multiple alternative methods that may be capable of solving the task in different situations. This recursive decomposition of task into subtasks by means of a method is called the task/method decomposition.

Metalevel (or reflective) knowledge is knowledge about domain knowledge and problem solving knowledge. Intuitively, metalevel knowledge can be used to model criteria for preferring some methods over other methods for a task in a specific situation.

Problems to be solved by GYMEL are represented as complex structured cases. Figure 2 shows an example of a partially expanded case structure.

The feature Hard-Group is a list of representative notes in the melody (one note per measure). This list is built starting from the Melody.

We will say that one note is a representative note using the next heuristics:

- If the note is in the first half of the measure and, in this measure, there is not a representative note, we take this note as a representative note.

- If the note after a representative note has a longer duration, this note will be the representative note for this measure.

As we will see, the feature Hard-Group together with the feature Harmony are the base for making comparisons between cases.

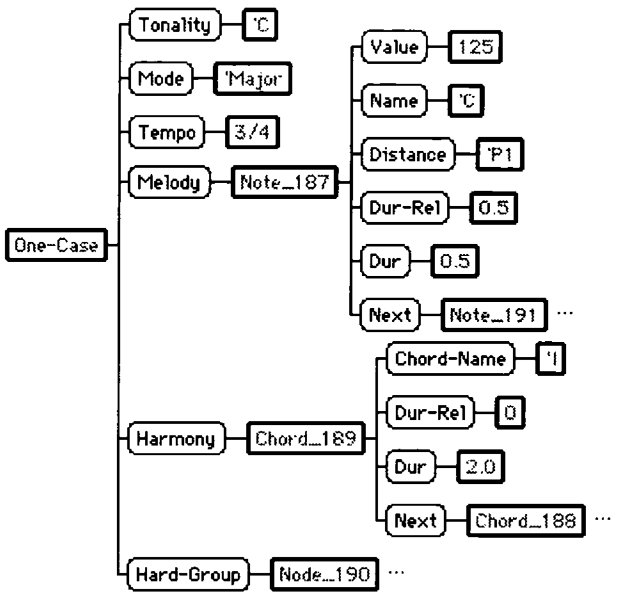


Figure 2. A NOOS Browser visualizing a partially expanded case structure. Features are represented as thin boxes and dots indicate not expanded terms.

Once a problem is solved, NOOS automatically memorizes (stores and indexes) that problem. The collection of problems that a system has solved is called the Episodic memory of NOOS. The problems solved by NOOS are accessible and retrievable. This introspection capability of NOOS is the basic building block for integrating learning, and specifically case-based reasoning, into NOOS.

The CBR module

First, GYMEL has to retrieve from the case base all those cases that could have some useful information to harmonize the new melody. This task requires making some kind of comparison between the problem melody and all the examples in the case base. This comparison becomes very difficult if we work with all notes in each melody. Therefore, we must select only the most representative notes. GYMEL selects one representative note for each measure. Furthermore, to find a matching between two melodies in the case base turned out to be a very difficult problem also. The solution is to work with melodies not as a global entity but as a set of simple problems (representative notes in each measure of the melody). Each representative note is therefore a simple problem and we must find one chord for that note. Each chord does not depend only on that representative note but also on the previous chords.

GYMEL focuses the retrieval as a pattern search on the case base. One pattern shows the situation of the problem melody in a local area. The objective is to find, in the melodies of the case base, the same situation and transfer the corresponding solution. Figure 3a shows such pattern and Figure 3b its representation in the NOOS language.

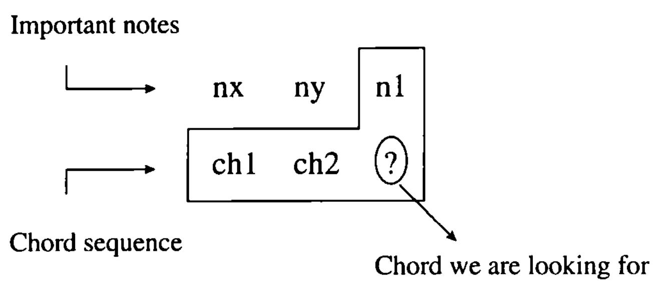


Figure 3a. Find a chord for n1 taking also into account the two previous chords ch1 and ch2.

```
(define (node)
 (note (define (note)
        (Distance (>> distance of n1)))
 (chords (define (chord)
          (Chord-Name any))))
 (prev (define (node)
        (chords (define (chord)
                 (Chord-Name (>> Chord-Name of ch2)))
        (prev (define (node)
               (chords (define (chord)
                        (Chord-Name (>> Chord-Name of ch1))))))
```

Figure 3b. Pattern representation in NOOS language.

A pattern in GYMEL is a linked sequence of **node** feature terms. Every **node** represents a representative note and the chord associated to it. Once we have defined a pattern we use it to search in the case base. This operation will give us different possibilities for that situation.

The next step is to adapt those possibilities to the problem. This action is straightforward because GYMEL uses a chord representation, which is independent from tonality. Chord names denote the distance between the fundamental note of the chord and the tonic of the tonality. For example, a IV-Major in the tonality of C is the F-Major chord but, in the tonality of B, is the E-Major chord. The name of the notes uses a similar representation. Then, for example, the p5 note in the tonality of C is the G note but, in the tonality of B, is the F note. This representation makes easier the tasks of retrieval and reuse.

We repeat these steps for every representative note in the melody (i.e. all notes in the feature Hard-Group). GYMEL perform backtracking in order to analyze all possibilities.

An example

Let us suppose we have a set of representative notes from a melody like Figure 4.

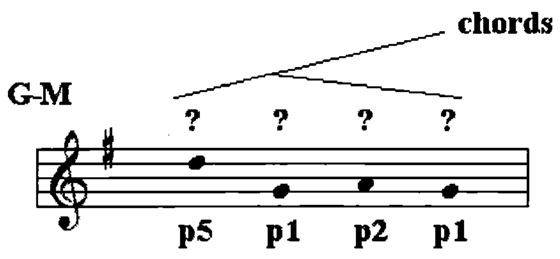


Figure 4.

In order to decide the first chord, GYMEL search for all cases with a P5 in the first representative note. Let us suppose it finds a I.

Now, it has to find a chord for the second note. The pattern for this situation is given in Figure 5.

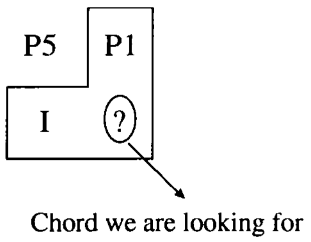


Figure 5.

That is, we must find a chord for P1, as a representative note, taking into account that the previous chord is a I. Let us suppose GYMEL finds two possibilities in the case base: IV and V_7. It selects the IV and later it will analyze the other possibility. Now, the search pattern is that of Figure 6.

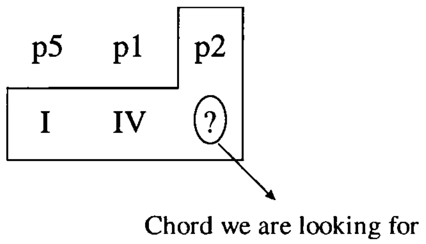


Figure 6.

The search in the case base results in a V_7. The last pattern is that of Figure 7 and contributes to the solution with a I.

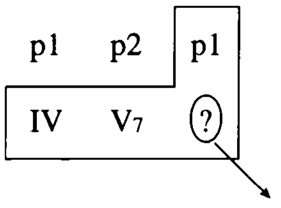


Figure 7.

At this point, GYMEL has finished a chord sequence (I - IV - V7 - I). After storing this solution, it will analyze other possibilities left along the way (for example the V7 for the note P1 in pattern showed in Figure 5).

The Rule-based module

We have seen the main procedure in the CBR module section but, what happens when that module cannot find a chord for a given situation because the case base is too small?

The rule-based module solves this problem proposing a possible chord to the CBR module using general knowledge about harmonization (Coker 1987) (Piston 1978) (Zamacois 1978). Broadly speaking, while the CBR module uses more specific knowledge about how to harmonize a melody, the rule-based module uses more general knowledge to solve situations not reflected in the case base. With this approach we can work with a small case base without limiting our capacity to find acceptable solutions.

Presently, the rule-based module is very simple (it contains only a few rules and the first rule applicable is the one that is selected) and as a consequence it may happen that a solution is neither found by the rules. However it is relatively easy to extend the rule system in order to represent better the basic general knowledge existing about harmonization. However we want to stress the fact that the rules have to be of a general nature, that is, as much independent as possible of the particular type of music (though always within western tonal music). It is precisely through the cases the way in which we can grasp the peculiarities of the different types of music (popular, modern, classic, etc). Since the case base grows by incorporating the melodies that the system harmonizes, the whole system improves by experience contrarily to a pure rule based system.

Although at the beginning if the case base is small the rule-based module is called often, we anticipate that the system will resort less and less to the rules as the case base will contain more and more cases.

Rules are a set of general conditional sentences. Some examples of these rules are:

- If the chord is the first chord of the sequence and the note is an important note of I chord then put a I.
- If the previous chord is a I or I_7 and the representative note is an important note of a II_7 chord then put a IIm_7.
- If the previous chord is a IV, IV_7, II or II_7 and the representative note is an important note of a V_7 chord then put a V_7.

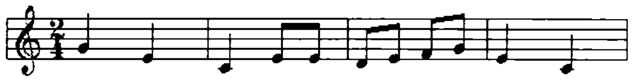
Testing GYMEL

For testing GYMEL we have used the "leave one out" technique with 20 musical phrases. The case base has 29 harmonizations from these musical phrases and the rule base contains basic general rules about harmonization. All musical phrases are from popular songs and every musical phrase has about 4 measures and a conclusive sense.

After the test, only in 2 instances the system did not find a solution and only one solution can be considered questionable, whereas using only the cases the system failed in 7 instances

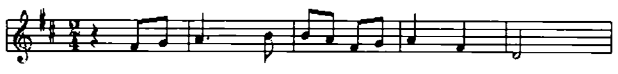
As an example of the obtained solutions, we show next the results for two musical phrases.

- Musical phrase from "Els putxinel.lis"



GYMEL's chord sequences:	I - IV - V7 - I
(one chord for measure)	I - IV - IVm7 - I

- Musical phrase from "Virolet Sant Pere"



GYMEL's chord sequence:	I - VIm7 - V7 - I
(One chord for measure)	

GYMEL can generate a MIDI file with the melody and the resultant harmonization using chords or arpeggios.

Figure 8 shows the output for the previous example using chords and Figure 9 another example using arpeggios.

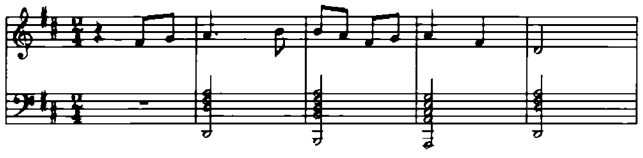


Figure 8.

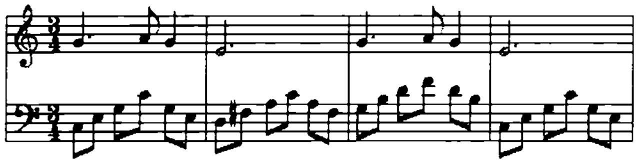


Figure 9.

Related work

There have been numerous works on rule-based harmonization but very few on case-based harmonization. Among them let us cite the work of (Macedo et al. 1997) and (Ramalho and Ganascia 1994). Macedo et al. uses analysis of music pieces from a seventeenth century composer as foundation for a restructuring process, providing a structured and constrained way of composing novel pieces, although keeping the essential traits of the composer's style. Ramalho and Ganascia address the problem of simulating creativity in Jazz performance.

To the best of our knowledge, there are not other work combining rules and cases for harmonization purposes.

Conclusions

We have seen that, for achieving the goal of harmonizing melodies using case base reasoning when the case-base is small due to the difficulty in obtaining the cases, the CBR has to be complemented by general knowledge expressed by means of rules. The combination of this two reasoning methods has been proven to be useful and in general can be useful in domains where it is difficult to find enough cases and is not suitable to work only with general rules.

7 References

Arcos, J.L. 1997. The NOOS representation language. PhD diss., Universitat Politècnica de Catalunya.

Coker, J. 1987. *Improvising Jazz*. Fireside, Simon & Schuster, New York.

Macedo, L., Pereira, F., Grilo, C., and Cardoso, A. 1997. Experimental Study of a Similarity Metric for Retrieving Pieces from Structured Plan Cases: Its Role in the Originality of Plan Case Solutions. In Leake, D., and Plaza, E. eds., Case-Based Reasoning, ICCBR-97, pages 575-586. Springer-Verlag.

Piston, W. 1978. *Armonía*. LABOR, Barcelona.

Plaza, E. 1996. Cases as terms: A feature term approach to the structured representation of cases. In Veloso, M., and Aamodt, A. eds., Case-Based Reasoning, ICCBR-95, pages 265-276. Springer-Verlag.

Ramalho, G., and Ganascia, J. 1994. Simulating Creativity in Jazz Performance. In Proceedings of the Twelfth National Conference on Artificial Intelligence. Volume One. Pages 108-113. AAAI Press / The MIT Press.

Sabater, J. 1997. GYMEL, sistema d'harmonitzaciò de melodies utilitzant raonament basat en casos. Master's thesis, Facultat de Ciències, secciò d'Enginyeria informàtica. Universitat Autònoma de Barcelona.

Zamacois, J. 1978. *Tratado de armonía I-II*. LABOR, Barcelona.