

An Artificially Intelligent Jazz Performer

Geber L. Ramalho,

Departamento de Informática -Universidade Federal de Pernambuco
Caixa Postal 7851, 50740-540 Recife, PE, Brazil
glr@di.ufpe.br — <http://www.di.ufpe.br/~glr>

Pierre-Yves Rolland and Jean-Gabriel Ganascia

Laboratoire d'Informatique de Paris VI (LIP6) - Université Paris VI
4, Place Jussieu - 75252 - Paris Cedex 05 - France
rolland@apa.lip6.fr, ganascia@apa.lip6.fr
<http://www-apa.lip6.fr/~rolland>, <http://www-apa.lip6.fr/ACASA/ganascia>

ABSTRACT

This paper presents an intelligent agent model for simulating the behavior of a jazz bass player during live performance. In jazz performance, there is a strikingly large gap between the instructions given in a chord grid and the music actually being played. To bridge this gap, we integrate Artificial Intelligence (AI) methods within the intelligent agent paradigm, focusing on two critical aspects. First, the experience acquired by musicians in terms of previously heard or played melodic fragments, which are stored in the agent's "musical memory". Second, the use of these known fragments within the evolving context of live improvisation. In previous papers, we have presented a model for an improvising bass player, emphasizing the underlying problem solving method. In this paper, we describe the fully implemented model and give more details on how melodic fragments are represented, indexed and retrieved.

1. INTRODUCTION

In live performances (such as theater, orchestra, dance or music), the artists must follow a script, which constrains their behavior. However, in most cases they must adapt this script to the environment. The interaction within a typical small jazz ensemble (soloist, pianist, bass player and drummer) playing for an audience can be sketched by a network where each player and the audience are represented by nodes (Cf. Figure 1). A chord grid contains a sequence of chords (e.g., Fm7, Bb7(9), EbMaj7) typically found in "real/fake books" (Sher 1991), which represents the underlying harmony of the song. Figure 2 shows the chord grid of "Stella by Starlight" (by N. Washington and V Young). Each musician's choices depend on three information sources: (1) the chord grid contents, (2) other musician's playing up to the present instant, and (3) the musician's own playing up to the present instant. The way human musicians actually consider these information sources to take musical decisions is unknown. This state of affairs characterizes computational modeling of jazz performance as an interesting and difficult problem to treat.

Besides their scientific interest, there is a growing demand for tonal music improvisation and accompaniment systems. These systems can automatically generate melodic lines (e.g., saxophone and bass), rhythmic lines (e.g., drums) and/or chord voicing chaining (e.g., piano) in a particular music style according to a given chord grid. They have been used as arrangement assistants (avoiding the detailed specification of the parts of some instruments), as rehearsal partners (letting the user play his/her instrument, while the computer plays the other ones), and as improvisation teachers (giving examples of possible improvisations for the chord sequences given by the user). So far, most of these systems have been dedicated to jazz (Baggi 1992; Brown & Sidley 1993; Giomi & Ligabue 1991; Hidaka, Goto & Muraoka 1995; Hodgson 1996; Horowitz 1995; Johnson-Laird 1991; Pachet 1990; Pennycook, Stammen & Reynolds 1993; Spector & Alpern 1995; Ulrich 1977; Walker 1994; Ramalho 1997a, Rowe 1993). However, other musical styles are also found, such as rock, reggae, bossa nova, etc. (Ames & Domino 1992; Band-in-a-box 1995; Levitt 1993).

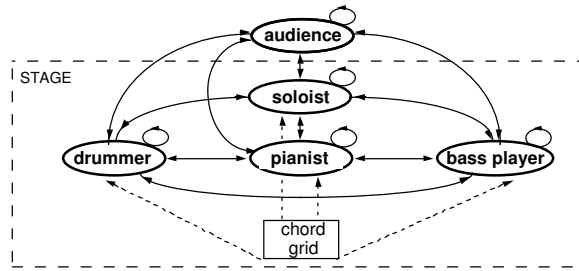


Figure 1 - Interaction model of a jazz ensemble

The jazz performance problem discussed above fits well within the cybernetics scope of study, due to the interaction network established among the performers. Born in the early forties, the cybernetic movement refers to three main conceptual constituents: *input/output*, *feedback* and *information*. Input/output refers implicitly to the notion of automata that simulate some elementary function (e.g., neurons). Feedback refers to the network of connections among automata, and to all the possible loops in this network. Finally, information corresponds to the means due to which all transformation and transfer operations take place. The recent evolution of the cybernetic paradigm is twofold. One approach is mainly concerned with the global behavior of networks and with the ability for networks to automatically learn connections among automata: it corresponds to the study of the so-called connectionism and dynamic systems (Fausett 1994). The other approach focuses on more complex automata, which are seen as intelligent agents cooperating in an environment. The latter approach is usually classified as Distributed Artificial Intelligence, or multi-agents systems (Gasser 1991; Jennings & Wooldridge 1995).

E m7(b5)	A 7		C m 7	F 7	
F m 7	Bb 7		Eb maj7	Ab 7	
Bb maj7	Em7(b5)	A7	D m7	G m7	C7
F maj7	G m7	C7	Am7(b5)	D 7	
G 7	G7		C m7	C m7	
Eb m7	Eb m7	Ab 7	Bb maj7	Bb maj7	
E m7(b5)	A 7		D m7(b5)	G 7	
C m7(b5)	F 7		Bb maj7	Bb maj7	

Figure 2 - Chord grid example

Our approach has been more closely related to Distributed Artificial Intelligence, since our project has addressed the construction of an intelligent agent, a jazz performer, interacting with its environment. However, unlike typical multi-agent models, we have not been concerned with the global evolution of the system, or with learning connections among the musicians or between the audience and the musicians. Instead of focusing on the system's behavior as a whole, we have concentrated our research on the way a particular performer interacts with the environment. More precisely, we have modelled the behavior of a bass player, whose task is to create and play a melody (bass line) in real time.

To accomplish this task, our agent reuses *melodic fragments*, originally played by human players. These fragments, called *cases* in reference to Case-Based Reasoning (Kolodner 1993), are stored in the agent's *musical memory*. According to the perceived context (the other agents' playing, the chord grid and the bass line played so far), the agent chooses the most adequate fragment in its memory, adapts it and "appends" it to the bass line being played.

The technique of reusing previously stored melodic and rhythmic fragments to compose new melodies and rhythmic lines has been increasingly applied in the design of tonal music improvisation and accompaniment systems, particularly in jazz styles (Ramalho, 1997b). However, the success of this technique depends on critical design choices concerning (rhythmic or melodic) fragments' nature,

representation, indexing, preference and adaptation. We propose an original problem solving method (Newell & Simon 1972) in which domain knowledge is intensively used. This additional knowledge provides more appropriate solutions to musical fragment reuse than those proposed by the current existing systems.

In previous papers (e.g., Ramalho and Ganascia 1994b) we have presented a preliminary model of an improvising bass player from an AI perspective, emphasizing the underlying problem solving method. In this paper, we will describe the fully implemented model and give more details on how the melodic fragments are represented, indexed and retrieved. The paper is structured as follows. Section 2 is concerned with the presentation of the interaction model, which corresponds to the external environment. Section 3 will outline the internal structure of the bass player which is built on an atypical problem solver where the goal is to play and where the inference engine combines two AI techniques: Case-Based and Rule-Based reasoning. Section 4 discusses results and points out possible extensions. Conclusions are given in the last section.

2. OUR JAZZ PLAYER AGENT

An *agent* is an entity (a program, a robot) that *perceives* its environment through sensors (camera, keyboard, microphone, etc.) and *acts* upon the environment by the means of effectors (hands, speakers, etc.). A *rational agent* is an agent that follows a *rationality principle*, according to which, given the perceptual data, the agent will choose the action that best satisfies its goals according to its *knowledge* (McCarthy & Hayes 1969; Newell 1982). From a computational point of view, the importance of the rational agent framework is to facilitate both analysis and design of intelligent programs. These goals are achieved since the agent's behavior can be described at an abstract level involving only perceptual data, goals, actions and knowledge. This avoids premature considerations about knowledge representation languages and inference mechanisms that will be used in the implementation of the actual agent.

2.1. Agent's architecture

The left-hand part of Figure 3 shows the basic components of a bass playing agent's environment, i.e., the chord grid, the other musicians and the audience. Instead of trying to deal with the complexity of real-time perception, we simulate the environment by means of a *scenario*, which will be explained in the next section.

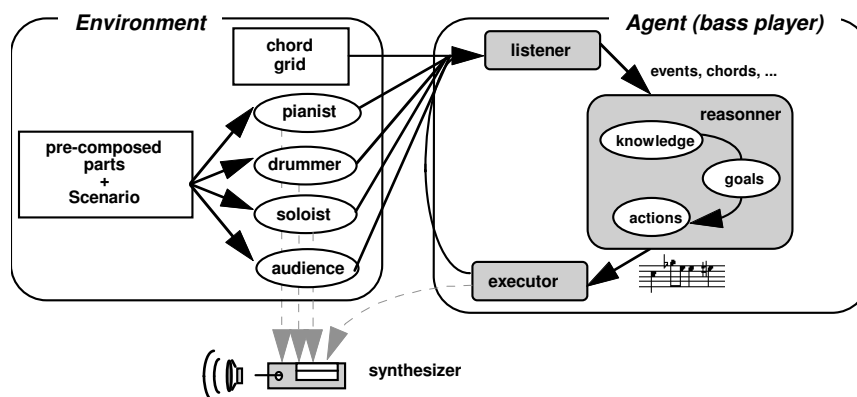


Figure 3 - Overall picture of our jazz playing agent structure

To define our agent we used the classic agent architecture adopted in AI and robotics applications (Ambros-Ingerson & Steel 1988; Russel & Norvig 1995). This architecture reflects the division of the agent's tasks into perception, reasoning (or planning) and acting. The right-hand part of Figure 3 depicts our bass-playing agent, which is composed of three specialized agents. The *listener*, that gathers the perceptual data, the *reasoner*, that plans the note to be played, and the *executor* that actually plays the notes at their

appropriate starting time. The agents' *own performance*, as played by the bass executor up to the present moment, is fed back into the agent's input. This reflects the fact that what a human performer has just played has a major influence on choices regarding what to play next (avoiding too numerous repetitions or smoothing transitions between phrases, for example).

2.2. The listener

The full interaction model sketched on Figure 1 involves hard issues on real time music perception (Pennycook, Stammen & Reynolds 1993; Rowe 1993). For instance, all musicians need to track the others' tempo (beat induction problem) (Allen & Dannenberg 1990; Desain & Honing 1994). In addition, musicians often need to detect phrase boundaries of what the other performers are playing (musical segmentation problem) (Dowling 1986; Lerdaahl & Jackendoff 1983; Narmour 1989). Musicians also need to evaluate other musicians' performance in terms of more abstract musical properties, such as dissonance, style, syncopation, etc. (see e.g., Longuet-Higgins 1984, Sloboda 1985).

In order to avoid tackling these delicate problems, we have decided to simulate the environment by means of the notion of a *scenario*: a simpler yet powerful representation of the evolving context. This scenario, which is given to the system in addition to the chord grid, is composed of two kinds of events (i.e., state descriptions): (1) the *other performers events* (e.g., "pianist is using dorian mode", "drummer is playing louder and louder" or "soloist is playing arpeggio based on the current chord"); (2) the *audience events* (e.g., "audience applauds" or "police comes"). The language we use to represent musical scenario events is inspired by Cypher system (Rowe 1993). This language allows the representation of various *global musical properties* (e.g., pitch range, loudness, temporal density, etc.) as well as the tracking of their variation in time. Influenced by ideas of Hidaka (Hidaka, Goto & Muraoka 1995) and Baggi (Baggi 1992), we have extended musical events' description to highlight the global musical atmosphere. For instance, "the temperature gets hot" when the other musicians are playing more notes, louder, in a higher tessitura and in more syncopated way. In addition, "the atmosphere is more colorful" when musicians are using chromatic scale, making chord substitutions and playing more dissonantly. Figure 4 shows a hierarchy of object-oriented classes used to represent scenario events. For instance, the fact "the pianist is playing more and more notes since beat 20 (to current beat 36)" is represented by an instance of `RhythmicEvent` whose attribute-values are: `lapse = 20-36`, `musician = 'pianist'`, `type = 'density'`, `value = 'medium'`, `variation = 'ascendant'`.

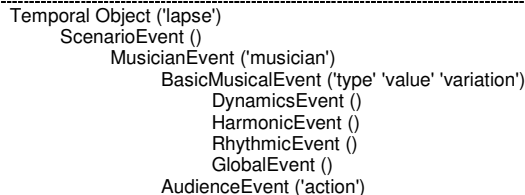


Figure 4 - The hierarchy of classes used to represent scenario events

The soloist, pianist, drummer and audience shown in Figure 3 are simplified agents whose main task is to extract, in real time, events from their respective scenario's part and then signaling these events to the bass player. This way, the information about the future cannot be accessed by the bass player, since the scenario events are only available at their specified starting time (see Figure 5). The synchronization between the various agents is carried out using a discrete time clock.

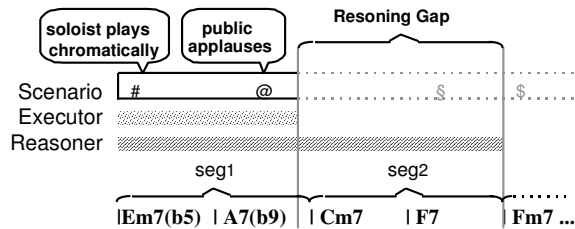


Figure 5 - Reasoning gap and scenario events access with respect to the reasoner and executor's current positions

2.3. The executor

The executor behaves as a scheduler. It simply holds a full description of the notes (pitch, starting time, duration, amplitude) planned by the bass reasoner, and plays these notes on a MIDI synthesizer at their precise starting time (see Figure 3).

The notes that are currently being played by the executor have been planned earlier by the improviser. In other words, there is a time lag (called *reasoning gap*) between the creation of the bass line (by the improviser) and its actual execution. The improviser will typically plan notes for the n^{th} chord grid segment while the executor is playing the notes corresponding to the $(n - 1)^{\text{th}}$ segment, or an even earlier one. Interleaved planning and execution is a technique widely used in robotics and in other real-time planning applications (Russel & Norvig 1995). As shown on Figure 5, the current time position of the executor determines what scenario events are available to the improviser.

Besides its scheduling task, the executor performs some changes in the notes planned by the reasoner. For instance, the starting time and duration of the notes are changed according to the tempo, to give a swing-like feel. Figure 6 illustrates some of these changes. Another important change performed by the executor is the modification of the last notes of a given melodic fragment in order to smooth the passage between it and the next fragment. This avoids melodic jumps and improves the homogeneity and fluidity of the bass line.

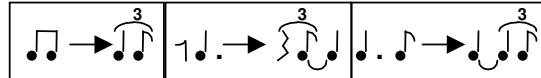


Figure 6 - Examples of changes in notes duration and starting time

2.4. The reasoner

The reasoner is the most complex agent to design, since it must be able to choose, in real time, the notes which will be played in the next grid segment. This must be done according to different information sources: the chord grid's contents, what the other musicians have been playing so far (i.e., the scenario's contents), and what the agent itself has been playing. The main difficulty in building this agent is the acquisition and representation of the knowledge used by musicians to interpret and combine all this information. Often, musicians cannot explain their choices at the note level in terms of rules, constraints or any known formalism (Pachet 1990; Baggi 1992; Ramalho 1997a). They usually justify what they have played in terms of more "abstract" *musical properties*, such as swing, tension, contour, density, etc. However, the interaction between these concepts is not simple to be determined. For instance, a soloist may not be able to explain the choice of each note (pitch, amplitude and duration) of a passage, even if a given scale is consciously used. Let us mention that such abstract musical properties are crucial for successfully modeling other kinds of musical tasks, e.g., pattern extraction (Rolland & Ganascia 1996), and interpretation learning (Widmer 1994).

2.4.1. Reuse of melodic and rhythmic fragments

During the knowledge elicitation work, we have realized that the difficulty of the interviewed jazzmen in given an analytical explanation of their choices is partially due to the empirical way they learn how to create music. Although the jazzmen use rules they have learned in schools, these rules do not embody all the knowledge they employ when playing. For example, there is no logical rule chaining that can directly instantiate important musical properties such as tension, style, swing and contrast, in terms of notes. Especially in jazz, musicians learn how to play by listening to and imitating performances of famous musicians (Ramalho & Ganascia 1994a; Sabatella 1996). Through this empirical learning process, the musicians acquire a set of examples of melodic/rhythmic phrases (and chord chaining) which can be reused in the future. This claim is corroborated by the work of musicologists on the identification of typical melodic fragments used by famous jazzmen, such as Charlie Parker (Owens 1974), Bill Evans (Smith 1993), Miles Davis (Baker 1980), John Coltrane (Kernfeld 1983), Lester Young (Rottlieb 1991).

Usually, the literature employs the term “pattern” in a broad sense, including any melodic or rhythmic fragment occurred once in a given corpus. However, the word “pattern” should designate only *recurrent* musical structures, i.e., melodic fragments that occur frequently enough, according to some specific threshold (Rolland 1998; Rolland & Ganascia 1999). In this paper, we prefer to use the term “fragment” due to its generality: all patterns are fragments, but the inverse does not hold. Incidentally, most of the musical improvisation and accompaniment systems impose no restrictions on the occurrence frequency of what they call “patterns”.

From the computer science standpoint, what is important is that melodic/rhythmic fragments reuse is an extremely powerful technique to minimize knowledge acquisition problems. First, these fragments can be easily acquired either by consulting experts or the literature (Aabersold 1979; Coker 1970), or by using melodic/rhythmic pattern extraction programs (Rolland & Ganascia 1999; Pennycook et al. 1993; Rolland 1998; Rowe 1993). Second, melodic/rhythmic fragments represent knowledge *in extension* (Woods 1975), since they implicitly codify concrete example solutions (e.g., appropriate phrases) for a given musical problem (e.g., a chord sequence). For instance, it is hard to identify complete and fine-grained “rules” that indicate how to build a bass melodic phrase for a given chord sequence with respect to different contexts (e.g., position within the song, what the musicians are playing, etc.) and desired musical properties (e.g., density, melodic contour, etc.). Figure 7 shows two bass line fragments that could be retrieved, transposed and reused to Fm7(b5) Bb7. In a situation where a simple and smooth phrase is wanted, the left-hand side fragment is more adequate. In a situation where a slightly dissonant and syncopated phrase in the medium range is wanted, containing several notes and using mainly chord notes, then the right hand side fragment is preferable.

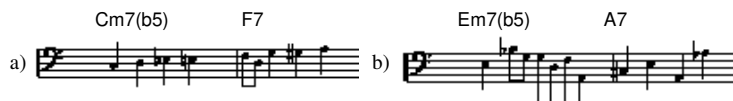


Figure 7 - Two bass line fragments as played originally by Ron Carter on *Stella by Starlight* (Aabersold 1979)

The idea of fragment reuse is to retrieve and adapt previously stored melodic or rhythmic fragment and to chain them in order to compose new melodic or rhythmic lines, as illustrated in Figure 8. The current fragment reuse context C may be characterized by the following elements: (1) the melodic/rhythmic line played so far by the system and by the other musicians/systems; and (2) the chord grid-related information (previous, current and future chords; current position within the song; song style; song tempo, etc.). This does not mean that playing jazz is limited to chaining previously known fragments. Nevertheless, this is a successful way of constructing tonal music improvisation and accompaniment systems.

```

While current-position < end-of -grid, do:
  Determine the next chord grid segment S whose notes will be computed;
  Describe the current context C with respect to S;
  Retrieve, from the library L, a melodic/rhythmic fragment F that is the most adequate with respect to C;
  Adapt F to C, obtaining F';
  Add F' to the melodic or rhythmic line played so far

```

Figure 8 - Basic loop of the systematic fragment reuse algorithm

To implement the fragment reuse strategy, we have introduced in our agent model the notion of musical memory, which is a repository of melodic fragments as played originally by famous jazz bass players. Furthermore, we have placed this strategy within *Case Based-Reasoning* (CBR) theoretical framework (Kolodner 1995), which is a powerful tool for highlighting and proposing solutions for the main issues in musical fragment reuse. In fact, musical fragment reuse can be viewed in the CBR perspective, since melodic and rhythmic fragments are episodes that can be reused in a similar context in the future. A case is generally composed of the description of a problem and its corresponding solution. When a new problem is presented to a case-based system, its relevant properties (indexes) must be described. The system uses these indexes to perform a search into the case base, in order to retrieve a case whose problem part is the most similar to the new given problem. The solution associated to the previously stored problem is adapted to solve the new one. Under this standpoint, the main issues on musical fragment reuse are the following:

- What are the fragments' nature (information content) and length?
- How should fragments be represented?
- How should the case base (fragment library) be organized?
- What are the good indexes for storing a musical fragment so as to retrieve it adequately in the future?
- What are the criteria for preferring a particular fragment in the case base?
- How should the retrieved fragment be adapted to the current situation?

Other developers of music improvisation and accompaniment systems have reached conclusions similar to ours. They have adopted the reuse of melodic or rhythmic fragments (or "cases") as the systematic strategy (Ulrich 1977; Baggi 1992; Band-in-a-box 1995; Hodgson 1996) or as an additional strategy (Spector & Alpern 1995; Pennycook, Stammen & Reynolds 1993; Walker 1994). The existing systems employ different kinds (from melodic fragments to pitch interval sequences) and lengths (from one-chord to many-measure fragments) of musical fragments, as well as different representation schemes (from simple character strings to sophisticated object-oriented representations). These systems employ a random-based strategy to retrieve musical fragments. However, each of them take advantage of different information (from fragment's desired properties to the similarity between the fragment's original context and the current context) in order to bind the random choices. They also apply different adaptations (from none to significant ones) on the retrieved fragment with respect to the current context.

In our model, a case corresponds to a bass melodic fragment (the solution part) exactly as it has originally been played (pitch, onset, duration and loudness). The problem description part consists of a set of indexes indicating two sorts of information: the context within which the fragment was played (e.g., underlying chords, local tonality, location within the song, etc.); and the musical properties concerning the fragment (e.g., dissonance, melodic contour, etc.). The fragments may have different lengths, corresponding to particular chord sequences, called *chord chunks* (e.g., II-V, II-V-I). The appropriate fragment is retrieved according (1) to how similar the context in which it was played is to the current context, and (2) how close the fragment fits the required musical properties.

It is out of the scope of this paper to discuss in details the pros and cons of each design choice made by the current systems (see Ramalho 1997b for this). Here, we will concentrate on describing accurately our approach to determine melodic fragments' length, as well as the indexing and preference criteria for melodic fragment reuse.

2.4.2. Overview of the reasoner's functioning

Some researches (Johnson-Laird 1992; Pressing 1988) have pointed out the difficulties in formalizing improvisation and accompaniment tasks as problem solving (Newell & Simon 1972). The formalization begins by the definition of the states describing the domain problem and the operators, which yield the passage from one state to another. Then, problem solving can be formally characterized by a process of applying operators to a given initial state until the final or goal state is reached. Taking as an initial state of the problem space a time segment with no notes, the music composition problem would consist of filling this segment with notes satisfying some criteria. The main difficulty is the determination of well-defined and static goals (i.e., the criteria for recognizing the final state). In fact, at the beginning of the performance, musicians seem to have only vague, sometimes contradictory, ideas about what they will really play. These ideas only become coherent and precise during the process of playing. Worse, they can change continuously according to the environment events. For instance, the performer reacts accordingly to what is being played by the other musicians.

Despite these obstacles, we claim that it is possible to formalize music creation activities, such as improvisation and accompaniment, as problem solving. To do this, problem solving should include the process departing from “vague ideas” to precise criteria under which a given set of notes is considered an acceptable solution (Ramalho & Ganascia 1994b). More precisely, the music accompaniment or improvisation process can be formalized as two main problem-solving stages. The first stage is the determination of the criteria (length and musical properties) that will guide the notes to be played. The last stage is the effective materialization of these criteria in terms of notes. In other words, the agent needs to know both how to set and how to solve the problem, and must execute these two tasks continuously, in order to capture the environment changes along the performance.

According to this perspective, our agent's composition process was implemented as a succession of four reasoning sub-processes, which is repeated until the end of the grid:

1. *Grid segmentation*: the agent establishes the chord grid segment with respect to which the notes will be computed and played. This segmentation is done by recognizing particular chord sequences called *chord chunks* (Section 3.1);
2. *PACTs activation*: some *PACTs* (*Potential ACTIONs*) are activated according to the environment perceptual information (i.e., the grid, the scenario events occurred so far, and the bass executor's own output). These *PACTs* (described in Section 3.2) are performance intentions that arise and vanish at definite times, such as “play diatonic scale in the ascending direction during this measure”. They serve to determine an initial set of musical properties (such as dissonance, scale, melodic contour, etc.), which will influence the choice of the best stored melodic fragment;
3. *PACTs selection and assembly*: this stage (Section 3.4) aims at combining musical properties associated to the activated *PACT*, as well as solving the conflicts among them, in order to get a single *PACT*. The activated *PACTs* considered in this stage are those whose lifetimes “intersect” the current chord grid;
4. *Fragment retrieval and adaptation*: the query to the musical memory (Section 3.5) is formulated using the information of the resulting *PACT* plus the description of the current context (i.e., current grid segment, last grid segment, recent scenario events, etc.). Once retrieved, the fragment is adapted to the current musical context.

The first three steps above define a set of criteria in terms of musical properties. It is important to emphasize the dynamic, “on the flight”, characteristic of these stages. To our knowledge, excepting (Pachet 1990, Walker 1994), most of the improvisation and accompaniment systems uses fixed pre-defined criteria to generate the music from the beginning to the end. In other words, the machine cannot change autonomously and dynamically the criteria during the improvisation or accompaniment. This solution does not take into account the impromptu changes in the environment and the natural evolution of the performance. Without an on-line criteria adjustment, the very essence of jazz is compromised, since the spontaneous interaction between musicians cannot be apprehended.

The last step actually concretizes these criteria (musical properties) by retrieving from the musical memory a most appropriate melodic fragment with respect to them. It would be possible to retrieve a melodic fragment based only on the description of the current context (i.e., underlying chords, type of chord sequence, position within the form, etc.), as shown in Figure 8. However, PACTs activation and assembling aid to improve the control on melodic fragment's retrieval. For instance, instead of simply addressing the request "find a melodic fragment played on the chord sequence "G7 Cmaj7", lasting 8 beats, at the beginning of the chorus", we can enrich the query by appending other criteria, such as "a fragment that is chord-based, has few notes, and is on a low tessitura".

3. THE REASONER IN DETAILS

In this Section, we will present the main stages of the composition process carried out by the reasoner during the performance.

3.1. Grid segmentation

The choice of melodic fragments *length* should take into account the musical plausibility (fixed vs. variable length) and the granularity (local vs. global information) (Ramalho 1997b).

The "natural" structure for building melodic lines is the (melodic) phrase, which has different lengths (Lerdahl & Jackendoff 1983; Narmour 1989). However, unlike natural language, there is no clear definition for the beginning and end of these musical phrases. Because of the difficulties in identifying musical phrases boundaries, some music systems adopt a fixed-length composition or improvisation "step". This step is usually equal to one beat (Johnson-Laird 1991; Hidaka, Goto & Muraoka, 1995), one measure (Giomi & Ligabue 1991) or two measures (Brown & Sidley 1993).

Regarding the reasoning granularity, one should find a compromise to guarantee both continuity and quick reaction. On one hand, it is difficult to control the overall coherence using too short musical fragment, such as one beat or one note, since the line generated is too "broken". Moreover, some musical properties (e.g., density), that may serve as guidelines for choosing the best musical fragment, cannot be measured adequately for too short fragments. On the other hand, very long fragments are inadequate since, while the system is playing a melodic or rhythmic fragment, many important events may occur in the environment (e.g., the soloist is playing chromatic scales). The system will not be able to react quickly, changing what it has planned to play accordingly.

The best compromise (regarding both plausibility and granularity) employed by the existing systems so far is to choose musical fragments corresponding to a single chord (Band-in-a-box 95; Pachet 90) or a particular chord chunk (Hodgson 96; Ramalho 97a). Chord chunks, such as II-V, II-V-I, VI-II-V-I, are abundantly catalogued in the Jazz literature (Baudoin 1990) because of the role they play in jazz harmonic analysis, listening and extemporization. Many jazz improvisation methods suggest that such chord chunks can be the basis for constructing one's own improvisations, and listening to and retaining Masters' solos and accompaniments (Baker 1980). In our work, we have adopted the hypothesis that the bass player's improvisation and accompaniment processes progress by steps, each corresponding to a chord chunk.

The recognition of chord chunks is performed by a particular harmonic analysis system based on a more general analysis system that our research team developed earlier (Pachet et al. 1996). Contrary to the latter system's task that provides a complete hierarchical analysis of tonalities in a chord grid, our analysis task is simpler, since it is only concerned with chunk recognition. However, as described below, we needed to introduce some particular rules for solving conflicts and for completing grid segmentation, taking into account chords duration and testing the availability of melodic fragments in the musical memory for a given chord chunk.

The computation of the grid segmentation follows roughly three steps:

- *Chunk recognition*: for a sequence of chords, all chord chunks belonging to a given lexicon are identified. This lexicon is composed of all chord chunks underlying the melodic fragments contained into the agents' musical memory;
- *Chunks conflict resolution*: the conflicts due to time overlapping between chunks are solved;
- *Grid filling in*: the chords that do not belong to any chunk are analyzed as single-chord chunks.

Figure 9 shows a rule for recognizing a V-I major. A chunk is only recognized if the musical memory contains a bass fragment played on the same kind of chunk, i.e., the chunks' tonalities may be different, but their shapes and rhythmic structures must be identical. Because of this three step processing structure, the simplest grid segmentation occurs when the lexicon contains no (composed) chord chunk. In that case, the grid will be segmented chord by chord.

```

Rule: majorTwoFive
For c1, and c2 instances of Chord and for a given set of chunks referred to as Lexicon
IF
    meets(c1,c2).      "c2 begins when c1 ends"
    duration(c1) = 4.
    duration(c2) = 4.
    isMinor(c1).
    not(isHalfDiminished(c1)).
    isDominant(c2).
    intervalBetweenRoots(c1,c2) = fourth.
    includesShape(Lexicon, MajorTwoFive, rhythmicStructure(c1, c2)))
THEN
    x := Create a MajorTwoFive.
    tonality(x) := majorScale(fourth(rootPitchClass(c2))).
    Complete description of x given c1, c2 and c3.
    Insert x in the fact base

```

Figure 9- Example of rule for recognizing a chord chunk

Conflict resolution is done according to preferences concerning chord chunk types and lengths (larger chunks are preferred to smaller ones if the first chunk's lapse completely contains the second's). For instance, a sequence like "E7 Am7 Dm7 G7 Cmaj7" will be segmented into "E7-Am7" and "Dm7-G7-Cmaj7" instead of "E7" and "Am7-Dm7-G7-Cmaj7". The preferred chunks are kept in the base fact, while the others are discarded.

Figure 10 illustrates a possible segmentation of "Stella by Starlight" according to a given chord chunk lexicon. This lexicon is composed of the chord chunks the bass player can recognize at a given moment, i.e., the set of chord chunks that compose the bass player's musical memory, as discussed later.

Besides the identification of the next chord chunk, the agent's task includes the description of each chunk according to the following features:

- harmonic shape (e.g., II-V, II-SubV-I, V-I, VI-II-V-I);
- local tonality (e.g., Eb major, A minor);
- time lapse; underlying chords; rhythmic structure (i.e., duration of each chord);
- position in form (e.g., beginning, turnback or turnaround);
- section (e.g., in a 32-bar AABA standard, a chunk beginning at the bar 9 is in the second section);
- repetition (i.e., the cardinality of the current chorus);
- backward resolution (i.e. interval between the root of the previous chunk last chord and the root of the current chunk first chord);
- forward resolution (i.e. interval between the root of the current chunk last chord and the root of the next chunk first chord— taking into account grid circularity).

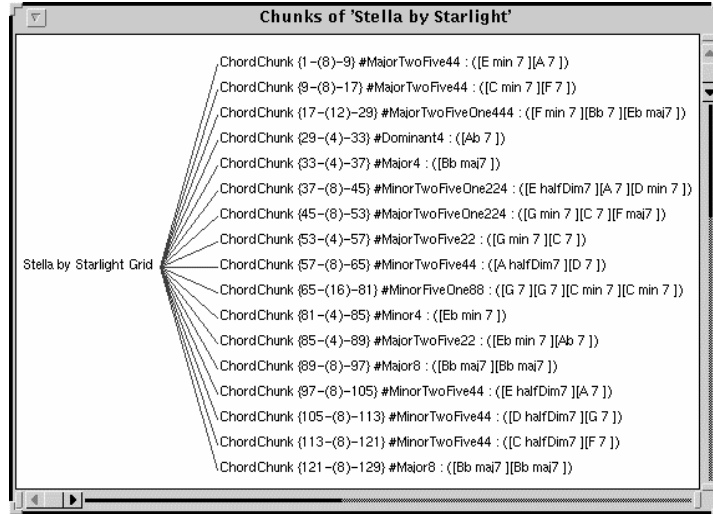


Figure 10 - Example of chord grid segmentation. The chunks elements displayed are: the lapse, harmonic shape and rhythmic structure (as a unique word) plus the underlying chords.

All this information is important to evaluate how similar the current chunk (context) is to the musical fragments' original underlying chunks (context). Figure 12 shows examples of detailed description of two particular chord chunks, as they are described in the musical memory of our agent.

3.2. The notions of PACTs and musical memory

In our approach, we assumed that, at any given time, the agent has “intentions” related to notes that it plans to play immediately or at a later moment. These intentions may concern the notes themselves or some musical properties (e.g., “syncopateness”, intensity, scale) of the notes. We represent such intentions under the form of PACTs as originally introduced by Pachet (Pachet 1990). Examples of PACTs are: “play syncopated phrase during this last section”, “play louder and louder until the end of the improvisation section”, “two measures from now start playing using dorian mode”.

The notion of PACTs constitutes the keystone of our model, unifying the representation of both the concrete melodic fragments in the musical memory and the abstract musical properties. In fact, PACTs provide a knowledge representation framework whose flexibility favors the description of musical material according to different points of views and at different abstract levels (Ramalho & Ganascia 1994b). For instance, the melodic fragments that constitute the musical memory are represented as fully specialized PACTs, called *playable PACTs*. These PACTs contain descriptions of a melodic fragment in terms of its sequence of notes (represented by their onset, duration, pitch and loudness) as well as the above mentioned musical properties. Through a knowledge acquisition work with some jazz bass players, we have identified a core vocabulary of musical properties for describing bass line fragments. These properties are: loudness, amplitude contour, pitch contour, tessitura, scale, dissonance, syncopation, density, rhythmic style (quarter-based, half-based, or eight-based), leading tone (whether leading tones are used), inversion (whether the tonic is played in the first beat), line style (chord-based or stepwise), pull down, repeated notes (whether repeating notes technique is used), and “classiness” (how recurrent the fragment is).

The PACTs that describe partially or completely melodic fragments in declarative terms are called *Standard PACTs*. A different kind of PACT, called *Transforming PACT*, describes transformations applied to a playable PACT. An example of the latter is the PACT “now, play this lick transposed one step higher”. Figure 11 illustrates the hierarchy of object-oriented classes used to represent the two main kinds of PACTs. This is a simplified hierarchy, as a detailed description of PACTs implementation is out of the scope of this paper (see Ramalho 97, for further details).

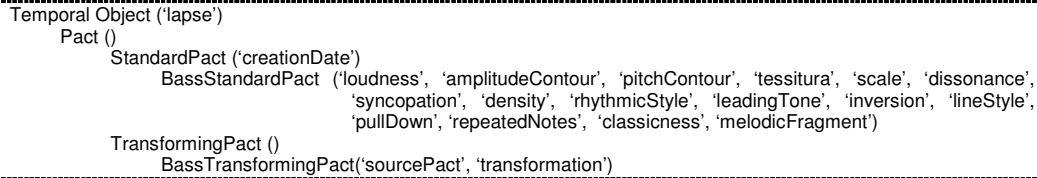


Figure 11 - The hierarchy of PACTs classes

Once the notion of PACT has been defined, we can now introduce more precisely the musical memory. A case in the musical memory is composed of one primary standard PACT (called *consequent PACT*) and one secondary PACT (called *antecedent PACT*). The consequent PACT contains the melodic fragment that will actually be retrieved and reused when the agent is playing, while the antecedent one corresponds to the PACT played just before. Figure 12 shows the interface used for case acquisition of a fragment of Ron Carter's bass line played in "Stella by Starlight" (Aabersold 1979). For both the antecedent and the consequent PACT, three main kinds of information are stored:

- the melodic fragment itself (bottom window of Figure 12);
- the description of the underlying chord grid segment (middle window of Figure 12);
- its musical properties (top window of Figure 12).

Case #7

Antecedent Pact	Consequent Pact
loudness: #forte	loudness: #forte
amplContour: #nor	amplContour: #nor
tessitura: #medium	tessitura: #low
pitchContour: #nor	pitchContour: #desc
dissonance: #low	dissonance: #medium
scale: #medium	scale: #medium
density: #quarterBased	density: #quarterBased
rhythmStyle: #slow	rhythmStyle: #slow
syncopation: #arpeggio	syncopation: #arpeggio
lineStyle: #true	lineStyle: #true
first inversion: #false	first inversion: #false
repeated notes: #false	repeated notes: #false
leading tone: #false	leading tone: #true
pull down: #false	pull down: #false
classiness: #false	classiness: #false
musStyle: #false	musStyle: #false
transform: #false	transform: #false
lapse: (37-45)	lapse: (45-53)
shape: #MinorTwoFiveOne	shape: #MajorTwoFiveOne
rhythmStruct: #2 2 4	rhythmStruct: #2 2 4

Case #7

Antecedent Pact Context	Consequent Pact Context
shape: #MinorTwoFiveOne	shape: #MajorTwoFiveOne
rhythmStruct: #2 2 4	rhythmStruct: #2 2 4
tonality: D Harmonic Minor Scale	tonality: F Major Scale
backRes: Augmented fourth	backRes: Perfect fourth
fourRes: Perfect fourth	fourRes: Major second
position: #other	position: #other
section: 2	section: 2
repetition: 1	repetition: 1
chords: E halfDim7, A7, D min 7	chords: G min 7, C7, F maj7
lineName: Stella by Starlight	lineName: Stella by Starlight
performer: Ron Carter	performer: Ron Carter
gridName: Stella by Starlight	gridName: Stella by Starlight
gridStyle: standard	gridStyle: standard
tempo: 180	tempo: 180

unnamed

File Notes Edit insert note set rest silence Global Edit Browse Graphics MIDI Various

1 Em7(b5) A7 Dm7 Gm7 C7

5

Figure 12 - Example of a musical memory case

We now comment briefly on how these three kinds of information are considered in the various systems. In representing the melodic fragment, we take into account all the notes' basic dimensions (i.e.,

pitch, beginning, duration, amplitude, and timbre). According to the Case-Based Reasoning paradigm, the closer a fragment is to a transcription of an actual improvisation or accompaniment fragment performed by a professional musician, the better it is, since it carries more information and implicit knowledge. By losing rhythm information in the fragment description, Ulrich's system discards one of the musical knowledge's aspects that is hard to formalize (Ramalho 1997a). Moreover, it is simpler to "grasp" the mutual interaction among the various dimensions by coding them altogether. This interaction is in fact difficult to formalize. Hodgson and Band-in-a-box designers also adopt that same policy as us for coding the fragments.

The majority of the current systems restrain the context description to the fragment's underlying current chords solely. Band-in-a-box, Hodgson's and Baggi's extend the context representation to include the next segment's first chord or the interval between the current segment's last chord root and the root of the subsequent one. As discussed in Section 2.4.1 and Section 3.1, we enrich the current grid segment description by adding information about the local tonality, chord chunk type, etc. The inclusion of the antecedent PACT aims to extend the description of the context in which the (consequent) fragment was played. In fact, the antecedent PACT imposes prerequisites in terms of harmonic and melodic continuity to the next fragment to be played (i.e., the consequent PACT itself).

Regarding indexing, most of the current systems consider only a small set of musical properties to describe the musical fragments. NeurSwing (Baggi 1992) includes fragment density, and Hodgson includes melodic contour, dissonance and few other properties. Fragment length is taken into account by all the systems. Our system, instead, incorporates a quite exhaustive list of properties for describing bass line fragments in jazz style, as enumerated at the beginning of this section (e.g., loudness, amplitude contour, pitch contour, tessitura, etc.). The use of a rich *indexing vocabulary* enables a more accurate retrieval and also improves fragment reusability, since it facilitates the computation of the similarity between two melodic/rhythmic fragments (Ramalho 1997b). On the other hand, the inclusion of rich context descriptions and desired musical properties as additional criteria in fragment retrieval demand more domain knowledge as well as the implementation of further reasoning mechanisms. In fact, the system designer must perform a knowledge acquisition effort in order to identify all the relevant features, the relationships between them, the conditions under which a given property is desired, etc. This work depends on the collaboration with the expert and may be quite long. There is a trade off between the fragment retrieval powerfulness and the system implementation simplicity.

As seen previously, the grid segmentation process depends on a given lexicon of chord chunks. This lexicon corresponds to the set of underlying chunks of each consequent PACT in the musical memory. Thus, considering the grid segmentation process, in order that *any* chord grid be able to be segmented, this memory must contain a minimal set of PACTs "covering" all possible single-chord chunks: major, minor, dominant, half diminished and diminished.

3.3. PACTs activation

PACTs activation knowledge is represented in terms of *production rules* of the type "If situation S is perceived then activate PACT P". By means of a first order logic forward chaining inference engine, new PACTs are activated at each cycle (i.e., a chord chunk) according to different perceptual data. The activation of a PACT corresponds to the assignment of values to its attributes, i.e., the object-oriented instantiation of a given PACT class. Besides the knowledge representation facilities they offer (Watterman & Hayes-Roth 1978), production rules are highly adequate to implement agents' reactions to dynamic PACTs environments (Laird, Newell & Rosembloom 1987; Russel & Norvig 1995). All activated PACTs are stored in the agent's working memory, which operates as the fact base of classical expert systems. For each inference cycle, this working memory is updated by eliminating "obsolete" PACTs—i.e., PACTs whose lifetime ends before the beginning of the current chunk lapse—and by adding recently activated PACTs.

Many kinds of perceptual data may trigger PACTs activation. Some PACTs are activated according to specific *grid chords* or *chunks*. Here are some examples of these activation rules:

- IF the agent's "current chord chunk CC contains an altered chord, THEN activate PACT "play dissonant during chord chunk CC";
- IF the agent's current chord chunk C contains short chords (lasting less than 3 beats), THEN activate PACT "play the tonic of each chord at its first beat";
- IF the agent's current chord chunk has the same shape (e.g., II-V, II-V-I) as the one played just before, AND the agent is at an improvisation chorus, THEN activate a PACT with the same duration, different line style (e.g., arpeggio or stepwise), and different dissonance degree with respect to the last played PACT. (Figure 13 shows the implementation of this rule).

Other PACTs can be activated taking into account the *chord grid's structure, style and tempo*, as exemplified below:

- IF the agent is at the very beginning of the theme exposition chorus, THEN activate PACT "play arpeggio-based line, with tonics at the first beat of each chords during the next 8 measures";
- IF the song is a ballad played in a slow tempo (typically less than 120 quarternotes per minute), THEN activate PACT "play syncopated with low density (i.e., few notes) until the end of the song";
- IF the agent is at the very beginning of the improvisation chorus, THEN activate PACT "play gradually more and more notes until the end of improvisation part".

```

Rule: theSameShapeAsPreviousChunkWithDifferentColor
For bass-player bp
IF   isAtImprovisation(bp).
    shape(currentChordChunk(bp)) = shape(chordChunk(lastPlayedPact(bp))).
THEN
    | p |
    p := new(BassStandardPact).
    lapse(p) := lapse(currentChunk(bp))
    dissonance(p) := different(dissonance(lastPlayedPact(bp))).
    lineStyle(p) := different(lineStyle(lastPlayedPact(bp))).
    firstInversion(p) := not(firstInversion(lastPlayedPact(bp))).
    addPact(p, workingMemory(bp))

```

Figure 13 - Example of PACT activation rule

The recently detected *scenario events* also fire the activation of PACTs. Here are some examples of scenario-dependent activation rules:

- IF the drummer is playing quieter, THEN activate PACT "play quieter during the current chord chunk";
- IF soloist is using chromatic scale notes, THEN activate PACT "play with low dissonance (arpeggio-based line) during the current chord chunk";
- IF the soloist is playing many notes (high density), THEN activate PACT "play with low density";
- IF the environment's "temperature" is hot (i.e., musicians are playing more notes, louder, more syncopated and in a higher tessitura), THEN activate PACT "play hot during the current chord chunk".

Finally, PACTs are activated with respect to the *bass line played so far*, as exemplified below:

- IF the agent is at the beginning of an improvisation chorus and its bass line has been chord-based (arpeggio) during the two last chord chunks or more, THEN activate PACT "play walking bass until the end of the improvisation";
- IF the agent has been playing stepwise in ascendant direction for more than two measures, THEN activate PACT "make a drop during the current chunk".

As can be realized, most of these activating PACTs rules concern abstract musical properties of the notes to be played rather than the specific note-level information (i.e., pitch, onset, duration, amplitude of each note). As we said before, acquiring knowledge about the choice of specific notes is too hard. PACTs yield a more flexible way of acquiring and representing musical knowledge, since it is much simpler for musicians to justify their choices in terms of these general musical properties. In fact, we have verified that it is extremely easy to improve the agent performance by simply adding new PACTs activation rules to the knowledge base. During the development of our system, human bass players have listened to and played the

bass lines generated by our system. After each of these evaluations, we had no problem in adding new rules in order to activate more precise and adequate PACTs, achieving better results in melodic fragment retrieval.

So far, we have coded 35 rules for PACT activation that reflect part of the “common sense rules” we have identified based on our interviews with bass players. It is important to notice that activation rules have an influence on stylistic and/or aesthetic characteristics of the music produced. More precisely, depending of the rule set being used by ImPact, the bass lines it generates can get closer to particular bass playing styles or even to particular artists. The current set of rules specifically reflects the playing style and techniques of the bass players we interviewed, although we chose to select, among possible rules, those that appeared as the most common and versatile ones.

One of the difficulties in designing agents for dynamic and non-episodic environments concerns the selection of the past environment events to be considered in reasoning (Russel & Norvig 1995). Since PACTs can be activated with different start-time and duration, they can be notably useful to minimize the problem of environment filtering. Instead of trying to filter the most relevant events from the beginning of the task, it is easier to schedule some future, potential actions according to what the agent is doing now. For instance, let us suppose that the agent intends to create a rhythmic contrast during the bridge (3rd section of an AABA-structured song). Some measures before the “bridge” the agent can activate the two following PACTs: (a) “play few notes from now to the beginning of the next bridge” and (b) “play a lot of notes during the next bridge”. When the agent “arrives” at the bridge, it will naturally find the previously activated PACT (i.e., PACT “b”).

These two characteristics of PACT (abstract description of musical properties and temporal scheduling) enable the agent to implement a sort of *least commitment planning* strategy (Russel & Norvig 1995), which is crucial in complex environments.

3.4. PACTs selection and assembly

Once the new PACTs have been activated and pushed into the agent’s *short term memory*, the agent must *select* all the relevant PACTs with respect to the current chord chunk. This selection process simply consists of choosing from the short term memory all the PACTs (activated at the current step or in the past) whose lifetime overlaps the current chord chunk’s lapse. These selected PACTs will be assembled into a single PACT, which will serve to guide the melodic fragment retrieval from the musical memory (the case base).

Each PACT is activated according to different perceptual data, more or less independently of the PACTs activated previously. For this reason, the set of selected PACTs for the current chunk may contain *incompatible* PACTs. For instance, the PACTs “play in descending direction” and “play an ascending arpeggio in first inversion” are incompatible with respect to the property “pitch contour”, as well as the PACT “play an ascending arpeggio in first inversion” is incompatible with “play stepwise” with respect to the property “bass line style”.

The set of selected PACTs may also contain pairs of compatible PACTs, i.e., PACTs that carry complementary information and can therefore be *combined* into a new PACT. For instance, the PACT “use the chromatic scale” combined with “play in ascendant direction” yields “play a chromatic scale in ascendant direction”. Sometimes, when the information of the two compatible PACTs are put side by side, it is possible to compute unknown values of further attributes not yet instantiated. For example, the PACT “play quite dissonant notes” may be combined with the PACT “play chord-based notes”, yielding “play chord-based quite dissonant notes, avoiding the root at the first beat and adding passage notes”. In fact, the way to augment dissonance, respecting the chord-based constraint, is to avoid the tonic note and to use passing notes instead.

We defined an original problem solving method which solves the incompatibilities taking advantage of the fact that PACTs may be combined. According to our method, the agent’s initial state is the set of selected activated PACTs (whose lifetimes intersect the current chunk lapse). The final state (goal) is a playable PACT; i.e., a PACT whose attributes have specified values, including the very melodic fragment (Ramalho & Ganascia 1994b). In fact, the core property of PACTs is that they may be combined into “more

playable” PACTs as a function of their compatibility. Our problem solving (assembly) process is performed by the successive application of three basic problem-solving operators: *delete*, *combine* and *propagate*.

The *deletion* operator is used to solve conflicts between two incompatible PACTs. PACTs that are more “playable” are preferred. For instance, between the incompatible PACTs “play ascending arpeggio” and “play in descending direction”, the former is preferred. When this criterion is not enough to decide, PACTs’ creation date is considered: more recent PACTs are preferred since they may represent an urgent reaction to the environment. If the indecision remains, longer PACTs are preferred since they can improve the bass line homogeneity.

The *combination* operator transforms compatible PACTs into a new, more playable, one. For instance, “play a eighth-based rhythm” combined with “play chord-based style” yields “play a eighth-based rhythm, choosing chord notes”.

The *propagation* operator is applied to a single PACT when its specialized attributes (attributes with a specified value) can be used to compute the values of other attributes. In the combination example cited above, given the values of *rhythmicStyle* = *eight-based* and *bassStyle* = *chord-based*, it is possible to set the value of the attribute *repeatedNotes* to *true*. In other words, a strategy of getting low dissonance in high-density rhythm is to repeat the notes, i.e., doubling each (chord) note chosen.

In principle, when a PACT carries all the information about the musical properties the fragment must exhibit, the propagation operator is applied to it in order to determine the value of the attribute *melodicFragment* (see Figure 11). In other words, when all musical properties are set, the program starts the retrieval process in the musical memory, searching for the most adequate fragment with respect to these properties. In practice, there is no guarantee that the assembly process will yield a PACT specifying all fragment musical properties used, because the activated PACTs may have covered only a few properties. Thus, when only one PACT remains in the assembly space, the program forces the retrieval of a melodic fragment even if musical properties are lacking.

The assembly process is implemented by means of production rules. The rules that implement the *deletion* and *combination* operators contain in their premises some compatibility tests for a pair of PACTs. According to these tests, the action part of these rules specifies which kind of combination, if any, can take place between the two PACTs. The *propagation* operator is implemented by rules that aim to verify whether some new attribute value can be derived from the attributes that have just been specified. The Case-Based Reasoning mechanisms, which serve to retrieve the most adequate bass line fragment, are fired by the propagation operator when the conditions discussed in the previous paragraph are met. The presentation of the formal specifications of incompatibility measurements, as well as the algorithms for combining PACTs and propagating information within a PACT, are out of the scope of this paper (see Ramalho 1997, for details). Figure 14 shows one of the rules for detecting incompatibility between two PACTs. In this case, since the PACTs have exactly the same instantiated attributes, they are fully incompatible.

```

Rule: totalDirectCoverageIncompatibility
For all PACTs p1 and p2
IF   p1 ≠ p2.
      specializedAttributes(p1) = specializedAttributes(p2).
      IsPreferredTo(p1, p2)
THEN
      Remove p2 from the base fact

```

Figure 14 - Example of a rule for detecting incompatibility between two PACTs

3.5. Case retrieval and adaptation

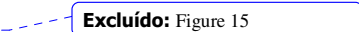
In fragment reuse strategy (see Figure 8), the retrieval process aims to find a fragment whose description satisfies a given *query*. Formally, the query $Q = (C, D)$, is composed of two elements: $C = \{c_1, \dots, c_k\}$ is a set of attribute-value pairs describing the current context (e.g., chords = ‘Cm7 F7’, tempo = 120, previouslyPlayedNotes = ‘F# E G’, etc.), and $D = \{d_1, \dots, d_j\}$ is another set of attribute-value pairs describing the desired musical properties (e.g., dissonance = low, rhythmStyle = quarter-based, etc.). A function $F(Q, L)$ establishes how Q is considered for choosing the best fragment from the fragment library L .

Not all the systems that are based on fragment reuse perform retrieval exactly as stated in the previous paragraph, but the presented formulation is the most general and serve cover the different retrieval strategies employed. The policies adopted by the existing systems differ among them in the following aspects: the richness of C's attributes; the richness of D's attributes; and the retrieval function $F(Q,L)$. As discussed earlier (Sections 2.4.1 and 3.2), most of the existing systems restrict indexing vocabulary to few features of the current context, and none or very few musical properties of fragments are included. The use of a short indexing vocabulary minimizes the knowledge acquisition and implementation efforts. However, the more attributes are specified in the query, the more accurate is fragment retrieval (Ramalho 97b). The retrieval function is random-based in the majority of the existing systems. The query criteria expressed by C and D descriptions serve just to bind or weight up the random choice. The Band-in-a-box retrieval strategy, instead, works with a user-provided priority weight associated to each fragment. This weight biases the random choices.

Avoiding randomness, our agent chooses the best fragment according to a mathematical similarity measure between the query (called *target case*) and each of the musical memory fragments (called *source cases*). This measure is performed using a *k-nearest neighbor classification*, a technique commonly used in Case-Based Reasoning (Aamodt & Plaza, 1994). This similarity measure is shown on equation 1.

$$similarity(T, S) = \frac{\sum_{i=1}^n w_i \times sim(a_{iT}, a_{iS})}{\sum_{i=1}^n w_i}, \quad (1)$$

T and S are, respectively, the target and source case having attribute set $A = \{a_1, \dots, a_n\}$. Each attribute a_i has a variable weight w_i associated with it. The weights serve to give different importance degree to the attributes (e.g., tessitura may be less important than rhythmic style). The primitive function *sim* determines the similarity between two attributes-values (e.g., $sim(tessitura_S = \text{low}, tessitura_T = \text{low}) = 1$; $sim(tessitura_S = \text{high}, tessitura_T = \text{low}) = 0$; $sim(tessitura_S = \text{medium}, tessitura_T = \text{low}) = 0.5$). The similarity between two melodic fragments is then a function of the similarity between each of its musical properties and grid context descriptors.

Once we know how to compute the similarity between any two cases, the retrieval process is straightforward.  enumerates the main steps of fragment retrieval and adaptation. As presented in section 3.2, the antecedent part of the target case is composed of the previous played PACT plus its underlying chord chunk. The consequent one is composed of the PACT resulting from the assembly process, plus the description of the current chord chunk. Following a general strategy for case retrieval, we have broken the retrieval process into two main stages: *matching*, where the N most relevant cases are selected; and *ranking*, where the best one is chosen. It is convenient to break retrieval into two stages since one can use simpler similarity assessments can be used during matching whereas the finer, more time-consuming, procedures are left to the ranking stage.

Excluido: Figure 15

```
Function bestFragment(targetCase, MusicalMemory, N)
  candidateCases := sameChunkCases(MusicalMemory, chordChunk(consequentPact(targetCase)))
  Order candidateCases according to the function similarity(targetCase, c)
  matchingCases := firstNElements(N, candidateCases)
  sourceCase := bestCase(targetCase, matchingCases)
  d := identifyDifferences(sourceCase, targetCase)
  newSourceCase := adapt(sourceCase, d)
  Return newSourceCase
```

Figure 15 - Main steps of fragment retrieval and adaptation

If the case base has many cases, the strategy of performing matching before ranking may not be enough to guarantee a good performance in a real time application, since similarity-based retrieval may be time-consuming. A hierarchical organization of the case base or some parallel search strategies are useful to speed up the retrieval process (Kolodner, 1993). Our system partitions the musical memory into groups of fragments. All fragments in each group have the same chord chunk shape (e.g., II-V-I minor, V-I major) and chord chunk rhythmic structure (e.g., 4-4-4, 8-8). This partition yields a good reduction in the search. From the 256 cases in the musical memory, only 20 in average are examined in each query.

In order to illustrate the role of PACTs in the retrieval process, let us inspect a simple example. The aim is to compute the notes for the grid segment corresponding to the 5th and 6th measures (Am7(b5))

D7) of beginning of “Autumn Leaves”, as shown in Figure 16. The tempo is slow (less than 200 quarters per minute). In such tempo and at the beginning of the exposition chorus, various “default” PACTs are activated: “play in low tessitura”, “play with low dissonance”, “prefer chord notes”, “play with low syncopation”, and “play with few notes”. The resulting assembled PACT biases the case retrieval process to choose a particular Ron Carter’s fragment (Figure 17a). Let us suppose that the soloist started to play chromatically at the fourth measure. In this case, the PACT “play with high dissonance” is activated to force the choice of a different fragment (Figure 17b).

Once the best fragment is chosen, it is necessary to adapt it to the current situation. The adaptation process’ goal is to assure harmonic coherence (with respect to the underlying chords) and melodic continuity (with respect to what the fragments have played so far). Moreover, some adaptation will take place to minimize the differences between the target and the source case’s attribute values. For instance, the target case would require a high-density, dissonant fragment. The retrieved case is dissonant, but contains only few notes. In this situation, the transformation called “insert more notes” is identified and then applied to the retrieved fragment.

Fragment adaptation is difficult to implement because of two reasons. First, some transformations are hard. For instance, there is no general procedure for adding/deleting notes in a melodic fragment. Second, the transformations are not all orthogonal. There is no guarantee that a transformation on a given retrieved fragment’s property can be applied without changing fragment’s other properties. For example, it is too difficult to change the melodic contour (e.g., from descending to ascending) of a fragment without changing its harmonic content. It is necessary to order transformations to control their mutual interference. Unfortunately, ordering any set of non-orthogonal musical transformations is an opened issue. As a consequence, the majority of the existing systems restrain the adaptation to simple and quite orthogonal transformations, such as time shift, tempo change, pitch transposition, and amplitude change. Hodgson’s and our system, however, change the pitch and/or duration of some notes, typically the last ones. This is useful to “soften” fragments chaining. NeurSwing makes more radical changes by introducing passing notes in the bass line, in order to augment density. NeurSwing also changes frequently a fifth by a flat fifth in the piano and bass parts, in order to augment tension.



Figure 16- Partially generated bass line on Autumn Leaves. The reused melodic fragments were originally played by Ron Carter, respectively, on Cherokee at measures 79-81 and on What is this thing called love (Wittcl) at measure 7 (Aabersold 1979)



Figure 17- Two different bass line continuations on Autumn Leaves. The new fragment was originally played by Ron Carter respectively in What is this thing called love at measures 13-14 and Stella by starlight at measures 61-62 (Aabersold 1979)

In order to attenuate the limitations of the fragments adaptation, we have introduced an adaptability measure as an additional criterion for fragment choice during the retrieval’s ranking stage (Ramalho, 1997). Frequently a fragment fits many desired properties but is not adaptable. For instance, it may not be entirely transposable to the current tonality without violating the instrument’s tessitura. In some situations, it is not straightforward to make transpositions to adjust some particular notes. In these cases, it

is much simpler to find out another fragment. In short, a similarity between the current and the original situation does not always guarantee a great adaptability. This problem was also detected by researchers working on case-based systems devoted to design tasks (Smyth & Keane, 1994; 1995). For such tasks, the adaptability-based retrieval strategy has been producing better results.

4. RESULTS AND POSSIBLE EXTENSIONS

In this section, we introduce more details about the implementation of the presented agent model. The obtained results, as well as, the future direction for further development are also discussed.

4.1. Implementation

Our system, called *ImPact*, was implemented using the Smalltalk-80 object-oriented programming language (Goldberg & Robson 1983). We reused the MusES system (Pachet 1994; Pachet et al. 1995) as a basic representation platform for representing the tonal musical concepts, like pitch classes, notes, keys, scales, chords, melodies, and so on. As of now, the system represents more than 220 Smalltalk *classes* and 4150 *methods*, half of which belongs originally to the MusES system. ImPact runs on the Macintosh, PC and Sun SparcStations. On a Sparc 10, ImPact can generate bass lines in real time, even in very fast tempi such as 600 quarter per minutes, which is largely enough for musical applications.

The mechanisms for rule base inference is handled by the NeOpus system (Pachet 1996), a first-order forward chaining inference engine implemented in Smalltalk-80. Currently, ImPact contains 84 production rules distributed among five specialized rule bases (for grid segmentation; PACT activation, assembly and information propagation; and implementation of the executor agent's changes on the planned notes). There are also meta-rule bases for providing a declarative control of the rules conflict resolution. Concerning the case base, it contains 256 cases made from 354 different bass line fragments (antecedent plus consequent ones). All the fragments were originally played by Ron Carter on six different songs (Aabersold 1979).

We also needed to develop some quite complex interfaces in order facilitate tasks such as case acquisition, case retrieval trace, PACTs activation and assembly, real-time monitoring of the agent's performance, and so on.

4.2. Experimental results

We have performed a great amount of experiments in order achieve an empirical validation of the implemented model (Ramalho 1997a). This validation was divided into two points:

- the verification of the musical and computational repercussion of the main choices we have made in the agent model;
- the assessment of the musical quality of the bass lines generated by ImPact, compared to lines created both by human bass players and by some of the existing jazz accompaniment systems

It is difficult to set up precise evaluation criteria and perform quantitative measurements of musical artworks, whether created by a computer or by a human. However, professional musicians can carry out accurate analysis of artwork belonging to a particular musical style. That is the case for some instrument performances in classic jazz styles, such as bebop. In this perspective, the evaluation method we have adopted consisted of presenting them some ImPact's bass lines and noting down their evaluation. The bass lines were presented in sound form and also in score form, where the musician is asked to play the created line. The comparison between ImPact and the other system was also supervised by human bass players. Despite the fact that the evaluation remains qualitative, this method improves the accuracy.

The musical results obtained with our agent model have exceeded our optimist expectations. The professional bass players think that ImPact's bass lines are much better than a human beginner's bass lines. Of course, ImPact plays bass at a much lower level than a skilled jazz player such as Ron Carter does. However, its lines are largely satisfactory.

We have made some qualitative comparisons between ImPact and two existing jazz accompaniment systems: band-in-a-box and NeurSwing. According to the bass players who collaborate with us in this research, (including NeurSwing’s author himself), ImPact generated globally better bass lines. Figure 18 shows the evaluation criteria used to compare these systems. The criteria are still subjective, but they have clarified some differences, which, once again, improves assessment accuracy (Ramalho 1997a).

		Band-in-a-box	ImPact	NeurSwing
Development	(how is the line developed along the performance?)	**	**	*
Diversity	(how varied are the line fragments?)	*	**	***
Fluidity	(how smooth are the chord changes?)	**	**	***
Melodic richness	(how do melodic contours and scales vary?)	***	***	*
Rhythmic richness	(how much different figures are used?)	**	***	*
Sound result	(how are sound characteristics, such as timbre and accentuation, considered?)	***	**	***
Harmonic sense	(to what extent is harmony respected?)	**	***	*
Swing	(does it swing?)	**	**	**

Figure 18 - Comparative table of b bass lines (***=good, **=acceptable, *=mediocre)

The experiments have shed light on many issues of our model, such as fragment reuse *vs.* accompaniment “rules”, PACT-based retrieval *vs.* “naïve” case-based retrieval, chord chunks *vs.* isolated chords; adaptability retrieval *vs.* pure similarity retrieval, scenario influence *vs.* playing alone, etc. Generally, the most important assumptions have been experimentally confirmed. First, the results obtained with a musical memory are much better than a pure rule-based approach we previously implemented. The experiments also shown that the musical results are all the better as the musical memory is larger. Second, the query refinement produced by PACTs activation and assembly has yielded better musical results than a retrieval based only on the grid context description. Moreover, PACT-based retrieval has revealed to be a simple, efficient and effective way of implementing dynamic reactions to the environment. It is interesting to note we have not obtained good results using note-level rules, but the rules employed in setting the desired higher-level musical properties have a significant contribution in the retrieval of fragments, which are note-level. Third, some experiments shown that choosing a chord chunk or a single chord as the reasoning step produces almost the same results, chord chunks being slightly better for “smoothing” the transitions between the fragments. In other words, the granularities of these two alternatives are nearly equivalent. Fourth, the use of an additional adaptability measure during retrieval has also been confirmed as a better strategy than that based only on similarity. Finally, as discussed in next section, using a scenario has caused less effect than expected, because the influence goes in only one direction: from the scenario to the agent.

We have made other tests, such as measuring the influence of the agent’s lack of time in reasoning. This occurs when the reasoning gap is too short, i.e., when the executor is almost “catching” the reasoner (see section 2.3). At the beginning of our work, we thought that considering the available time to think could be an important element for simulating musicians’ use of “licks”. For this reason, we implemented an any-time computation algorithm that shortcuts the agent’s decision process, forcing the reuse of “classic” fragments (licks). However, there was no need of shortcuts in fast enough machines available today (Sparc 10 or Pentium 166). Experimentally, we have observed that there was neither any musical profit in augmenting artificially time constraints to force the reuse of licks. We then gave up the hypothesis.

The evaluation method, based on the presentation of the score to the bass players, has given us important insights on how to ameliorate ImPact’s performance. Being presented one of ImPact’s first bass lines, a human bass player made a striking comment: “this line is not logical”. We expected criticisms regarding the lack of creativity or feeling, but we were sure that the program was “logical” ! In reality, the bass player said that the bass line did not respect the “logical development” in which a line should start with simple, consonant fragments and become gradually more complex as the improvisation begins. We added some new PACT activation rules, such as “play with tonic on the first beat, at the beginning of each chorus”, “play consonant and with few notes during the first section of the theme exposition chorus”, “play more and more dissonant during the improvisation chorus”, etc. With these changes, ImPact could immediately generate bass lines with a coherent development. This episode has showed us how simple it is to add new knowledge to ImPact in order to ameliorate the best-fitted fragment choice.

4.3. Criticisms and future developments

The scenario was very useful to test the capacity of the agent to react appropriately to environment events, without having to face, upfront, all the problems related to real-time music perception in. However, experiments we have run showed that sometimes the unidirectional character of the dialog between the bass player and the environment has negative effects on the quality of the ensemble performance. As no feedback exists, the synergistic effects favorable to musical “dialogs” are inhibited.

To minimize this problem, while still avoiding the implementation of complex perception mechanisms, we are implementing a multi-agent model for the entire rhythm section, where each musician is simulated by an agent. Since each agent knows what it is playing, from the actual notes to the more abstract properties, it will be possible to simulate some dialogues or negotiations that typically take place among the musicians. This experiment will try to capture the liveliness and spontaneity of jazz ensemble performance, which is often achieved through impromptu, on-the-fly interaction between musicians.

Another important criticism concerns the agent’s lack of self-evaluation and learning capabilities. Our agent does not know whether a fragment it has played was musically adequate or not. All the aesthetic preferences and constraints are only used during the process of computing the notes to be played. In other words, the preferences and constraints are encoded beforehand in the PACTs activation rules and in cases’ content. Consequently, the agent cannot learn or develop a personal style, since it may play many times the same song without figuring out what worked better. Its aesthetic preferences and constraints will then remain the same. If the agent could evaluate the musical adequacy of the played fragments, it would be able to complete the entire Case-Based Reasoning cycle by inserting them back, after appropriate change/adaptation, into the musical memory.

Of course, the existing systems also lack this “post-playing” musical evaluation. To our knowledge, Cypher (Rowe 1993) is the only system that tries to make this evaluation, but it is actually not performed after the agent has played. The problem is that the rules for performing an accurate aesthetic analysis are neither formalized nor universal. They are rather intuitive or subjective. Despite this difficulty, we think that it is possible to build a musical critic for some particular musical styles and instruments. Our conviction is based on our observations of the precise comments the experts made during the development and validation of our system.

Finally, in order to strengthen the validation of the presented model, we intend to apply it to different instruments and styles. We have already started to work with Brazilian music, whose rhythmic complexity rises interesting challenges. In fact, western music harmony and melodic development are well-investigated disciplines. However, the rhythmic phenomenon seems to be more intuitive, more puzzling to formalize. A case-based approach could be fruitful in this situation as well.

5. CONCLUSIONS

In this article, we have presented a knowledge-intensive model of a jazz bass player in a live jazz ensemble. This agent model attempts to conciliate the need for producing highly relevant reactions to the external environment (i.e., the other musicians and the audience) and the necessity of improvising in a personal style. To meet that challenge, we have made a particular use of Case-Based Reasoning mechanisms through the notion of musical memory. This memory contains reusable melodic fragments, originally played by human bass players. Coupled with the case-based approach, the model employs rule-based inference mechanisms to refine and improve the control of case retrieval, according to the environment events. The notion of PACTs guarantees the combination of the multilateral information streams between musical memory and environment. Melodic fragments retrieval is all the better as more PACTs activation rules are available. However, even in contexts where no such rules may be applied, the system can find a reasonable solution, i.e., a melodic fragment that was played in a similar context. The most important point of this hybrid model is its facility to incorporate new knowledge in order to improve its performance. New cases or new rules can be modularly inserted to the system without needing any change of the rest of the model components.

The reuse of melodic and rhythmic fragments is a promising technique in designing musical improvisation and accompaniment systems. This technique seems to apply even in composition tasks that have little or no chord grid guidance, such as those studied by Cope (Cope 1991). Musical fragment reuse minimizes knowledge acquisition problems, however, a great amount of knowledge is still needed to guide the indexing, retrieval and adaptation of the musical fragments. The systems that incorporate this knowledge extensively exhibit the best musical results. In fact, the intelligence of a system based on fragment reuse resides in its capability of interpreting a given situation in order to choose the most adequate fragment. The integration of different reasoning paradigms (case-, rule-, constraint- and neural net-based) that appropriately incorporate the available musical knowledge is the main research direction in building tonal musical improvisation and accompaniment systems.

ACKNOWLEDGEMENTS

We would like to acknowledge all people from LIP6 and from Departamento de Informática (UFPE) for their encouragement and fruitful discussions, especially François Pachet, Jean-Daniel Zucker, Stephan Grolimund, Vincent Corruble and Flávia Barros. We also thank the Brazilian Education Ministry (MEC/CAPES) for the financial support.

6. REFERENCES

- Aabersold, J. (1979). *Play-along: Paying Dues*. New Albany: Janey Aabersold Pub.
- Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Fundamental Issues, Methodological Variations, and system Approaches. *Artificial Intelligence Communications*, 7(1), 39-59.
- Allen, P., & Dannenberg, R. (1990). Tracking Musical Beats in Real Time. In *International Computer Music Conference*, (pp. 140-3). Glasgow: International Computer Music Association.
- Ambros-Ingerson, J., & Steel, S. (1988). Integrating Planning, Execution and Monitoring. In *Sixth National Conference on Artificial Intelligence*, (pp. 83-8). The AAAI Press.
- Ames, C., & Domino, M. (1992). Cybernetic Composer: an overview. In M. Balaban, K. Ebiciglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 186-205). Menlo Park: The AAAI Press
- Baggi, D. (1992). NeurSwing: An Intelligent Workbench for the Investigation of Swing in Jazz. In D. Baggi (Eds.), *Computer-Generated Music* (pp. 79-93). IEEE Computer Society Press.
- Baker, D. (1980). *Miles Davis Trumpet*. Giants of Jazz Series. Lebanon: Studio 224 Ed.
- Band-in-a-box (1995). *Band-in-a-box Pro 6.0*. Canada: PG Music Inc
- Baudoin, P. (1990). *Jazz: mode d'emploi, vol. 1 and 2*. Paris: Editions Outre Mesure
- Brown, D., & Sidley, S. (1993). The Expression of Aesthetic Principles as Syntactic Structures and Heuristic Preferences and Constraints in a Computer Program that Composes Jazz Improvisations. In AAAI Spring Symposium Workshop on Artificial Intelligence & Creativity, (pp. 133-6). Stanford: The AAAI Press
- Coker, J. (1970). *Patterns for Jazz*. Lebanon: Studio Publications/Recordings.
- Cope, D. (1991). *Computers and Musical Style*. Oxford: Oxford University Press.
- Desain, P., & Honing, H. (1994). Advanced Issues in Beat Induction Modeling: Syncopation, Tempo and Timing. In *International Computer Music Conference*, (pp. 92-4). Aarhus: International Computer Music Association.
- Dowling, H. (1986). *Music Cognition*. Orlando: Academic Press.
- Fausett, L. (1994). *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Englewood Cliffs, NJ: Prentice Hall,
- Gasser, L. (1991). Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics. *Artificial Intelligence*, 47:107-138
- Giomi, F., & Ligabue, M. (1991). Computational Generation and Study of Jazz Music. *Interface*, 20(1), 47-63

- Goldberg, A., & Robson, D. (1983). *Smalltalk-80: The Language and its Implementation*. London: Addison-Wesley.
- Hidaka, I., Goto, M., & Muraoka, Y. (1995). An Automatic Jazz Accompaniment System Reacting to Solo. In *International Computer Music Conference*, (pp. 167-70). Banff: International Computer Music Association.
- Hodgson, P. (1996). Modeling Cognition in Creative Musical Improvisation (unpublished poster). In *International Conference on Music Perception and Cognition*. Montreal.
- Horowitz, D. (1995). Representing Musical Knowledge in Jazz Improvisation System. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 16-23). Montreal
- Johnson-Laird, P. (1991). Jazz improvisation: a theory at the computational level. In P. Howell, R. West, & I. Cross (Eds.), *Representing Musical Structure* (pp. 291-325). London: Academic Press.
- Johnson-Laird, P. (1992). *The Computer and the Mind*. London: Fontana.
- Kernfeld, B. (1983). Two Coltranes. *Annual Review of Jazz Studies*, 2, 7-66
- Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo: Morgan Kaufmann.
- Laird, J., Newell, A., & Rosebloom, P. (1987). SOAR: An Architecture of General Intelligence. *Artificial Intelligence*, 33, 1-64.
- Lerdahl, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. . Massachusetts: The MIT Press.
- Levitt, D. (1993). A Representation for Musical Dialects. In S. Schwanauer & D. Levitt (Eds.), *Machine Models of Music* (pp. 455-69). Massachusetts: The MIT Press.
- Longuet-Higgins, C., & Lee, C. (1984). The Rhythmic Interpretation of Monophonic Music. *Music Perception*, 1(4), 424-41.
- McCarthy, J., & Hayes, P. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 6, 463-502.
- Narmour, E. (1989). *The Analysis and Cognition of Basic Melodic Structures: the Implication-Realization Model*. . Chicago: University of Chicago Press.
- Newell, A. (1982). The Knowledge Level. *Artificial Intelligence* 18, 87-127.
- Newell, A. & Simon, H. (1972) *Human Problem-Solving*. Englewood Cliffs, NJ: Prentice Hall
- Owens, C. (1989). Integrating Feature Extraction and Memory Search. In *Eleventh Annual Conference of the Cognitive Science Society*, Northvale: Erlbaum
- Pachet, F. (1990). Representing Knowledge Used by Jazz Musicians. In *International Computer Music Conference*, (pp. 285-8). International Computer Music Association.
- Pachet, F. (1994). The MusES system: an environment for experimenting with knowledge representation techniques in tonal harmony. In *First Brazilian Symposium on Computer Music - SBC&M '94*, (pp. 195-201). Caxambu: Computer Science Brazilian Society (SBC).
- Pachet, F. (1996). On the Embeddability of Production Rules in Oriented-Object Languages. *Journal of Oriented-Object Programming*, 8(4), 19-24
- Pachet, F., Ramalho, G., & Carrive, J. (1996). Representing Temporal Musical Objects and Reasoning in the MusES System. *to appear in Journal of New Music Research*, 3.
- Pennycook, B., Stammen, D., & Reynolds, D. (1993). Toward a Computer Model of Jazz Improviser. In *International Computer Music Conference*, (pp. 228-31). Tokyo: International Computer Music Association.
- Pressing, J. (1988). Improvisation: Methods and Models. In J. Sloboda (Eds.), *Generative Processes in Music: The Psychology of Performance , Improvisation and Composition* (pp. 129-78). Oxford: Oxford Science Publications.
- Ramalho, G. (1997a) *Construction d'un agent rationnel jouant du jazz*. Thèse de doctorat, Université Paris VI.

- Ramalho, G (1997b). Pattern Reuse in Tonal Music Improvisation and Accompaniment Systems. In Proceedings of the IVth Brazilian symposium on Computer Music. SBC: Brasilia. pp. 47-58
- Ramalho, G., & Ganascia, J.-G. (1994a). The Role of Musical Memory in Creativity and Learning: a Study of Jazz Performance. In M. Smith, A. Smaill, & G. Wiggins (Eds.), *Music Education: an Artificial Intelligence Perspective* (pp. 143-56). Workshop in Computing Series. London: Springer-Verlag
- Ramalho, G., & Ganascia, J.-G. (1994b). Simulating Creativity in Jazz Performance. In *Twelfth National Conference on Artificial Intelligence*, (pp. 108-13). AAAI '94. Seattle: The AAAI Press.
- Rolland, P.Y. (1998). FIEExPat : a Novel Algorithm for Musical Pattern Discovery. 12th Colloquium on Musical Informatics (XII CIM). Gorizia, Italy, September 24-26, 1998.
- Rolland, P.Y., Ganascia, J.G. (1998). Musical Pattern Extraction and Similarity Assessment. Contemporary Music Review. to appear).
- Rolland, P.Y., Ganascia, J.G. (1996). Automated Extraction of Prominent Motives in Jazz Solo Corpuses. In *Proceedings of the 4th International Conference on Music Perception and Cognition* (ICMPC'96) pp. 491-495. Montreal, August 1996. Montreal: McGill University.
- Rottlieb, L. (1991). Who So Sad, Pres? In L. Porter (Eds.), *A Lester Young Reader* (pp. 211-23). Washington: Smithsonian Institute Press.
- Rowe, R. (1993). *Interactive Music Systems*. . Massachusetts: The MIT Press.
- Russel, S., & Norvig, P. (1995). *Artificial Intelligence: a Modern Approach*. Englewood Cliffs: Prentice-Hamm, Inc.
- Sher, C. (1991). *The New Real Book (vol. 1 and 2)*. . Berkeley: Sher Music.
- Sabatella, M. (1996). *A Jazz Improvisation Primer*. <http://www.fornet.org/~marc/primer>.
- Sloboda, J. (1985). *The Musical Mind: the Cognitive Psychology of Music*. Oxford: Oxford University Press.
- Smith, G. (1993) Homer, Gregory, and Bill Evans? The Theory of Formulaic Composition in the Context of Jazz Piano Improvisation. Ph.D. Thesis, Harvard University.
- Spector, L., & Alpern, A. (1995). Induction and Recapitulation of Deep Musical Structure. In *Fourth Workshop on Artificial Intelligence and Music, IJCAI-95*, (pp. 41-8). Montreal.
- Ulrich, W. (1977). The Analysis and Synthesis of Jazz by Computer. In *Fifth International Joint Conference on Artificial Intelligence*, (pp. 865-72). Massachusetts.
- Walker, W. (1994) *A Conversation-Based Framework for Musical Improvisation*. Ph.D. Thesis, University of Illinois at Urbana-Champaign.
- Watterman, D., & Hayes-Roth, F. (1978). *Pattern-Directed Inference Systems*. New York: Academic Press.
- Widmer, G. (1994). The Synergy of Music Theory and AI: Learning Multi-Level Expressive Interpretation. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. AAAI Press/MIT Press, Cambridge, MA, pp.114-119.
- Woods, W. (1975). What's in a Link: Foundations for Semantic Networks. In *Representation and Understanding*, eds. D. Bobrow and A. Collins, 35-82. New York: Academic Press.
- Jennings, N. R. and Wooldridge, M. J. (1995). Intelligent Agents: Theory and Practice *The Knowledge Engineering Review*, 10 (2), pp.115-152.