NetNeg: A Connectionist-Agent Integrated System for Representing Musical Knowledge

Dan Gang and **Claudia V. Goldman** * and **Daniel Lehmann** and **Jeffrey S. Rosenschein**
Institute of Computer Science
The Hebrew University
Givat Ram, Jerusalem, Israel
ph: 011-972-2-658-5353 fax: 011-972-2-658-5439
Email: dang, clag, lehmann, jeff@cs.huji.ac.il
URL: http://www.cs.huji.ac.il/$\{\sim clag, \sim dang, \sim jeff\}$

Abstract

The system presented here shows the feasibility of modeling the knowledge involved in a complex musical activity by integrating sub-symbolic and symbolic processes. This research focuses on the question of whether there is any advantage in integrating a neural network together with a distributed artificial intelligence approach within the music domain.

The system is designed to perform in real time and to be used for interactive computer music composition or performance. The hybrid approach introduced in this work enables the musician to encode his knowledge and aesthetic taste into different modules. This is done by applying three distinct functions: rules, fuzzy concepts, and learning.

As a case study, we began experimenting with first species two-part counterpoint melodies. We have developed a hybrid system composed of a connectionist module and an agent-based module to combine the sub-symbolic and symbolic levels to achieve this task. The technique presented here to represent musical knowledge constitutes a new approach for composing polyphonic music.

Introduction

Simulating and modeling a musician's activities are tasks that are appropriate for experimentation within the framework of artificial intelligence. The cognitive processes a musician undergoes are complex and nontrivial to model. Whenever we are involved in any musical activity, we are faced with symbolic and subsymbolic processes. While listening, our aesthetic judgment cannot be applied by following explicit rules solely. Applying a learning mechanism to model the task of listening (Bharucha & Todd 1991; Berger & Gang 1997) has been shown to be a convenient (and appropriate) way to overcome the explicit formulation of rules. Nevertheless, some of these aesthetic processes might have a symbolic representation and might be specified by a rule-based system (Jackendoff 1991).

The contribution of this work to AI is in the area of knowledge representation. To demonstrate the advantages and computational power of our hybrid knowledge representation, we design a model that describes the different aspects a user might be interested in considering when involved in a musical activity. The approach we suggest in this work enables the musician to encode his knowledge, intuitions, and aesthetic taste into different modules. The system captures these aspects by computing and applying three distinct functions: rules, fuzzy concepts, and learning.

The primary goal is to design a system that will perform in real time. Such a system will be useful for interactive computer music. The human musician and the machine will mutually interact in live performance or during the composition process. In these creative processes the machine will enable flexible representation and processing of the user's musical knowledge. The user's musical experience and her musical aesthetic taste (or the musical experience and the musical taste of other users) will be reflected during the interaction with the machine.

The current work follows and simulates real time musical contrapuntal processes. Contrapuntal music is a common musical form and is desired for many interactive musical situations. Representing musical knowledge might add musical intelligence to the machine response. However, creating contrapuntal music in real time is a complex and challenging domain of research. A single paradigm may not be sufficient to deal with this kind of music. We therefore suggest using this music creation task as a domain in which to test the feasibility of our hybrid knowledge representation.

To examine the performance of the hybrid system, we chose a specific task for implementation. The specific case study we have chosen to experiment with is the polyphonic vocal style of the Sixteenth Century; more specifically, we investigate two-part species counterpoint (i.e, bicinia).

First Species Counterpoint: this is the first species, the most restrictive one, out of five species defined in species counterpoint music. Following (Randel 1996):

The progressive arrangement of the method, in dialogue form, with the rules for each species depen-

* Dan Gang and Claudia Goldman are supported by Eshkol Fellowships from the Israel Ministry of Science

dent more or less on the restrictions of the preceding was widely admired for its pedagogical value.

In first species counterpoint, the cantus firmus (i.e., one of the two voices) is given in a whole note representation. The counterpoint (i.e., the second voice) is created by matching a note against another note in the cantus firmus. This matching follows specific rules as we explain below.

The specific task we examine is different in one important aspect from the cognitive task of composing counterpoint melodies: our system creates *both* parts of the melody. The computation performed by the system is quite similar to the process that a composer would go through, but it differs from it in at least two major respects. First, we do not incorporate any backtracking process, whereas the composer may freely go back and forth. Second, our system has a very myopic view of the melody it is creating. When it decides on the next element, it has no knowledge of the overall structure of the piece, it has only knowledge of the last element, some vague knowledge (decayed) about the previous part of the melody and no knowledge about its continuation.

In spite of the simplicity of the problem, our approach can serve as a basis for further investigation of more complex musical problems. This simple domain could have been formulated in a rule based system. Nevertheless, our aim was to choose a starting domain that we could evaluate and in which we could control the complexity of the process.

Artificial Intelligence and Music

We shall describe extant work on those aspects of AI techniques in Music that apply directly to this work. A detailed overview of music systems that use AI tools can be found in (Camurri 1993).

The Sub-symbolic Approach

Knowledge learned by the machine is expressed by the states and the connections between simple processing units (neurons).

Listening, performing, and some other musical activities can be represented using a sequential stream of information. The choice of Jordan's sequential net (Jordan 1986) is appealing in such cases. Jordan's sequential net is a version of the back-propagation algorithm. Using the learning algorithm, the sequential net is able to learn and predict sequential elements (such as the sequence of a melody's notes or harmonic progression). The sequential net contains three fully-connected layers. The first layer contains a pool of state units and plan units. The second layer is the hidden layer, and the third layer is the output layer. The output layer and the state units contain the same number of units. The output layer is fed back into the state units of the first layer for the computation of the next sequential element. The value of a state unit at time t is the sum of its value at time $t-1$ multiplied by some decay parameter (the value of the decay parameter is between 0 to 1) and the value of the corresponding output unit at time $t-1$. The state units represent the context of the current sequential element, and the output layer represents the prediction of the net for the next sequential element.

Peter Todd (Todd 1991) suggested exploiting the Jordan sequential net for predicting sequential musical elements. His neural network model presented a connectionist approach for algorithmic composition. In the learning phase, the net learns a set of melodies' notes. Each melody is associated with a unique label encoded in the plan units. In the generalization phase, new melodies are produced by interpolation and extrapolation of the labels' values encoded in the plan units. The resulting melodies share similarities with the melodies within the learning set. These similarities are unique and different from those resulting from other methods, and they are interesting from the compositional aspect.

The Hybrid Approach

In the hybrid approach, knowledge about the world is represented by an integration of a sub-symbolic system and a symbolic system (e.g., a neural net can be integrated with a rule-based agent system).

A key motivation for the hybrid approach is the assumption that handling the complexity of AI tasks is beyond the reach of a single paradigm. Melanie Hilario (Hilario 1995) distinguishes among various hybrid approaches (in her terminology, neurosymbolic integration). She suggests classifying approaches into two strategies: unified strategies and hybrid strategies.

Unified strategies enrich neural networks with symbolic capabilities. Hybrid strategies combine neural networks and symbolic approaches at different levels. Hybrid neurosymbolic models can be either translational or functional hybrids. Translational hybrid systems use neural networks as the processors. The symbolic approach is applied on the network input and targets. Functional hybrid systems exploit both the neural network and symbolic components equally.

The system we present in this work is functional hybrid. Moreover, it is loosely coupled, since each of the components (i.e., the symbolic and sub-symbolic) act locally in time and space, and the interaction between them is always initiated by one of them. In our case, the integration of both components is appropriate to the chain-processing integration mode as explained in (Hilario 1995). Specifically, we can look at one of the processes as doing the main task, and the other as pre/post processing the relevant information. In NetNeg, either of the two modules can be viewed as the main module, and in charge of the other.

To the best of the authors' knowledge, the integration of a symbolic system with a sub-symbolic system, and in particular the integration of a rule-based agent module with a neural network module for representing musical knowledge, is novel.

Software Agents

Agents are functional, independent software modules that are programmed to act on behalf of the user. Among the salient features of software agents are autonomy, adaptation, and sociability.

One of the main research areas DAI is concerned with regards applying coordination protocols to multiagent systems. The agents programmed to follow these protocols, or agents that are able to learn how to behave, coordinate with other agents in the same system in order to achieve global goals, or to avoid conflicts.

One way used by agent designers to quantify the agents' performance is to let the agents compute a utility function. The algorithms that describe the agents' behavior take into consideration the utility values, and according to them, the agents choose their actions. Therefore, the actions that the agents will perform are strictly related to the computation of the utility function, which specifies the agents' interests.

In our case we deal with two part melody of first species counterpoint . We chose to represent each part of the melody with a separate agent. In this way each agent is able of evaluating the quality of its part. This will be computed with a utility function as described in this paper. Then, the agents negotiate to get the maximal utility possible given the constraints of the other agent (and its correspondent voice).

NetNeg's Architecture

In many musical styles, the composer creates different sequences of notes (i.e., melody lines) that will be played simultaneously. In contrapuntal styles, each sequence, taken in isolation, should follow some aesthetic criterion, and in addition the sequences should sound appropriate when combined. The musician achieves his overall result by compromising between the perfection of a single component and the combination of sequences as a whole. Thus, in this activity there is a constant tradeoff between the quality of a single sequence versus the quality of the combined group of sequences. When a musician is faced with such a task, he is involved in a cognitive process, that we suggest might be seen as a negotiation process. He has to compromise between the melodies' notes by choosing from among the permitted notes those that are preferable. A general view of the architecture of *NetNeg* is shown in Figure 1. NetNeg is composed of two sub-systems: a connectionist subsystem and a DAI-based subsystem. The role of the connectionist subsystem is to learn and generate individual parts of the polyphonic melody. In our implementation, the network learned to reproduce a series of learning examples that were taken from (Jeppesen 1992). Based on this learning process and the set of learning examples, the neural net is able to produce in the generalization phase new individual melody parts. In this phase, the network predicts in the output layer a vector of expectations for the next note in each part of the melody.

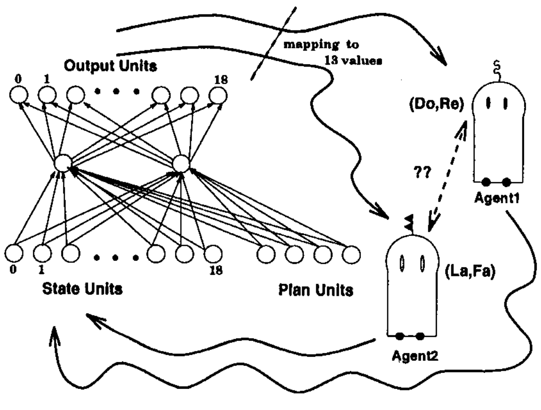

Figure 1: The NetNeg Architecture

Each agent represents one of the voices of the polyphonic music. It is responsible for choosing the tone that will be inserted in its voice at each unit of time. Each agent receives a different output vector from the network. On the one hand, each agent has to act according to its voice's aesthetic criteria; and on the other hand, it has to regard the other voice-agent such that both together will result in a two-part counterpoint. Both agents have to negotiate over all the other possible combinations to obtain a globally superior result. Thus, they influence the context with their agreement. Given this new context and the initial values of the plan units, the network will predict another output vector. This process continues sequentially until the melodies are completed.

We describe each of these modules separately and then present results from experiments performed with NetNeg.

The Connectionist Subsystem

Each part of the melody is produced independently by a neural network implemented in Planet (Miyata 1991). Todd (Todd 1991) previously suggested a sequential neural network that can learn and generate a sequence of melody notes. Currently, our neural network is based on the same idea, although we have extended it to include the representation of the contour of the melody.

We built a three-layer sequential net, that learns series of notes. Each series is a one part melody. Each sequence of notes is labeled by a vector of plan units. The net is a version of a feed-forward back-propagation net with feedback loops from the output layer to the state units (in the input layer). The state units in the input layer and the units in the output layer represent the pitch and the contour. The state units represent the context of the melody, which is composed of the notes produced so far. The output unit activation vector represents the distribution of the predictions for the next note in the melody for the given current context.

The role of the plan units is to label different sequences of notes. In the generalization phase, we can interpolate and extrapolate the values of the plan units so as to yield new melodies. At each step, the net is fed with the output values of the previous step in the state units together with the values of the plan units. These values will cause the next element in the sequence to appear in the output layer and it will be propagated as feedback into the state units (these connections do not appear in Figure 1). The current values of the state units are composed of the previous values multiplied by a decay parameter and the current output values.

Each note is represented by a vector of 19 bits containing 16 zeros and 3 ones: 8 bits represent the pitch and each note of the scale (here, the Dorian mode) is represented by 7 zeros and a single one, 9 bits for representing the size of the interval with the previous note (again a single one) and 2 bits for its direction, i.e., movement: ascending or descending (a single one).

In order to exploit the information encoded in the output units' activations, the pitch activations were combined with the interval and the movement activations. The interval and the movement units were used to support which pitch to choose. The activations of the output units were mapped into a vector of thirteen activations corresponding to the notes in more than an octave and a half.

Each agent receives the 13-length vector, and feeds the state units with their agreement (see Figure 1). Then, the network predicts another output vector given this new context and the initial values of the plan units. This process continues sequentially until the melodies are completed.

The DAI-Based Subsystem

The agent module was implemented by using the Mice testbed (Montgomery *et al.* 1992). In the implementation presented in this work, each voice of the bicinia is represented by an agent. Since we are dealing with two counterpoint melodies, then in DAI terms we design a multiagent system composed of two agents. The global goal of the system is to compose the two part melody following the rules of the style. In addition, each single agent has its own individual goal, i.e., to compose its melody by choosing the *right notes*. In particular, each agent has to act according to the aesthetic criteria that exist for its voice; at the same time, it has to compose the voice in a manner compatible with the other voice-agent such that both together will result in a two-part counterpoint.

At every time unit in our simulations, each agent receives from the network a vector of activations for all the notes among which it can choose. Were the agent alone in the system, it would have chosen the note that got the highest activation from the neural network, meaning that this note is the one most expected to be next in the melody. Both agents' choices might conflict with respect to the rules of the style and their own preferences. Therefore, we apply a negotiation protocol (Rosenschein & Zlotkin 1994) to allow the agents to coordinate and achieve their mutual goals. The agents will negotiate over all the other possible combinations to obtain a globally superior result.

In principle, each agent can suggest any of the n possible notes received from the network. Not all of these pairs of note combinations are legal according to the rules of the species. In addition, there are specific combinations that are preferred over others in the current context. This idea is expressed in this module by computing a utility function for each pair of notes. In this sense, the goal of the agents is to agree on the pair of notes that is legal and also achieves the maximal utility value among all options.

At each time unit, for each pair of notes, the agents start a negotiation process at the end of which a new note is added to each of the current melodies. Each agent sends to the other all of its notes, one at a time, and saves the pair consisting of its note and the other agent's note that a) is legal according to the first species style rules and b) has yielded the maximal utility so far. At the end of this process, the pair that has achieved maximal utility is chosen. Both agents feed their networks with this result as the current context so that the networks can predict the next output. Each agent, then, receives a new input based on this output, and the negotiation step is repeated until the melody is completed.

The term in the utility function that encodes the rules of a given style expresses (in our implementation) the rules of the polyphonic vocal style of the sixteenth century as they appeared in (Jeppesen 1992). A pair of notes is considered legal according to the following rules:

1. The intervals between pairs of notes in the two part melodies should not be dissonant (i.e., the second, fourth, and seventh intervals are not allowed).
2. There should be perfect consonance (i.e., unison, octave, and perfect fifth intervals) in the first and last places of the melody.
3. Unison is only permitted in the first or last places of the melody.
4. Hidden and parallel fifths and octaves are not permitted.[1]
5. The difference between the previous and the current interval (when it is a fifth or an octave) should be two (this is our modification).
6. The interval between both tones cannot be greater than a tenth.
7. At most four thirds or sixths are allowed.
8. If both parts skip in the same direction, neither of them will skip more than a fourth.
9. In each part, the new tone is different from the previous one.

[1]Following (Randel 1996), "parallel motion of perfect intervals is forbidden, nor may any perfect interval be approached by similar motion".

10. No more than two perfect consonants in the two part counterpoint, not including the first and last notes, are allowed (this is our modification).

The utility function we chose is one example of a function that computes all the aspects we described in the Section on NetNeg's architecture. In words, the utility (i.e., the quality) of a pair of notes T_1^t and T_2^t at time t will be zero when this combination does not match any rules of those explained above. In the other cases, the quality of this pair is given by a value computed from both tones' activations produced by the neural net. The final result of this utility is updated with a term representing how much each agent prefers contrary motion between the two candidate tones, T_1^t and T_2^t. The formal definition of the utility function may be found in (Goldman *et al.* 1999).

Experiments

We first ran each subsystem separately to examine the ability of each one of the two approaches (i.e, Neural Nets and agents) to cope with the general problem. We then ran the integrated system; we present results from all of these simulations. In this way, we show the ability of the whole system to produce results superior to the performance of either of the subsystems. Combining the modules gave us a more natural way of dealing with the processing of and representation of our task.

Running the Net Module

The task of the net was to learn to produce new two-part melodies. This case is different from the one faced by the whole system, in which only one-part melodies were taught. Therefore, we needed to represent both parts of the melody simultaneously. We used the same sequential net that was described above. In this case, we doubled the number of the units in each layer to represent two notes simultaneously, one for each part. In the learning phase the net was given four melodies, containing the two parts. Since our notes are taken from one and a half octaves, we represent the notes by their names (i.e., re), and those in the higher octave have an 8 concatenated to their names (i.e., re8). In this phase, the net learned the examples in the set with high accuracy. After short training[2] the net was able to completely reproduce the sequences without mistakes.

In the generalization phase we chose to interpolate the values of the plan units to produce new melodies. An example of a typical result follows:

`V2:`*re8 fa8** *sol8 la8 sol8 fa8 mi8 re8*
`V1:`*re8 sol si do8*** *mi re**** *do8 re8*

This resulting sequence reflects typical problems we encountered when dealing with this simple approach. The examples in the learning set imposed two different constraints on the net. The constraints regard the melodic intervals between the pitches in each part, and the combinations of pitches in both parts. The net is not able to cope with both constraints consistently, and thus it satisfies each, one at a time. For example in $*$, the combination chosen is not allowed in the specific style, although the melodic interval is fine. In $**$ the descending skip is not permitted, as well as the ascending skip in $* * *$, but both combinations are fine.

[2]For 20 hidden units it took less than 100 epochs to achieve an average error around 0.0001.

Running the Agent Module

The agents in our system know the rules of the specific style of the melodies we want to compose. They also know how to compute the system utility for a given pair of notes. We have run experiments with the agent module alone.

We run the module with the utility function described above, where the net's advice was assigned zero. Since we choose the pair of notes that get the maximal utility value at each step, the result is as follows:

`V2:`*re8 mi8 fa8 sol8 fa8 sol8 fa8 re8*
`V1:`*re8 do8 la sol la mi la re8*

The melody lacks the the aesthetics features that are learned by the net and are requested from each part. In both voices there are redundant notes (i.e, the appearance of note la in V1, and the series of notes in V2 from the third place to the seventh). There is no unique climax in any of the voices. There are two continuous skips in the last three notes in V1. There are too many steps in V2 (i.e., there is no balance between the skips and the steps).

We also observed that melodies can come to a dead end, when there is no pair of notes that can satisfy the rules of the specific style. In such cases, we have tested additional results that can be produced by the system when the agents are allowed to choose pairs with utility values smaller than the maximum.

Running NetNeg

In this section we present the main simulation performed on the whole system. In the training phase, the network learned to reproduce four melodies that were taken from (Jeppesen 1992). We have tested the performance of the network with different learning parameters, such as the number of hidden units and the values of the plan units.

In the generalization phase, given a specific vector of plan units, the network produces a new cantus firmi. We have chosen two different plan vectors for the net that will output the notes for each agent. We run the net, each time with the corresponding plan vectors, and mapped their outputs to two different thirteen activation values. Then, we run the DAI-based module with these inputs. The agents negotiate over the different pairs of possible combinations, computing for each the system utility. Finally, the agents agree upon a legal pair of notes that has yielded the maximal utility; alternatively, the agents might decide that no combination is legal, given the previous note in the melody.

In our current case, the nets are fed with the agents' agreement and the system continues to run. This process is executed until the two-part melodies are completed. Currently, the length of the melodies is fixed.

The following is an example of a resulting melody:

V2: *re8* *fa8* *sol8***** *fa8* *mi8* *re8* *do#8* *re*
V1: *re8* *la** *sol* *re*** *mi* *fa* *la**** *re*

In this example, we can observe that the system gives aesthetic results, quite appropriate for the species style with which we have experimented. Both parts are consistent with the combination constraint, as opposed to the simulation we ran solely with the neural network, where this constraint was not satisfied. Comparing with the simulation run with the agents alone, no redundancy was found in this example. Nevertheless, there is a contour problem as pointed out in (*) and (**) in A1's melody (V1). According to Jeppesen (Jeppesen 1992), it is preferred to descend by a step and then perform a descending skip. After a descending skip, we are expected to have a compensating ascending movement. In (***), we prefer to approach the last note by a step. A2's melody (V2) is perfect with regard to the contour. There is a single climax as shown in (****).

By running both modules together, we see the combined effect of all the aspects taken into consideration in the utility function. The melody of agent A_2 is perfect. This reflects the melodies learned by the neural net. The contrary motion term expressed in our heuristic is revealed in the relation between both melodies. The neural network was not trained with melodies such as the one created by agent A_1. The rules are expressed, for example, in the consonance of the parts of the melody.

Summary and Future Work

We have presented a novel and powerful computational approach for encoding the knowledge, intuitions, and aesthetic taste of a musician in different modules. In this work, we presented an example of an implementation that enables a human to flexibly guide the system to compose music in a style he chooses, under real-time constraints. The user might express multiple views and levels of knowledge to this system. For example, if he knows examples of the music he wants the system to compose, he can train the neural network with these examples. If he wants the music to follow specific rules he might formulate them in the agent module. In addition, he can regard other factors in the computation of the utility function.

Issues to be further investigated include other ways to integrate both modules, the study of other species (second, third, and fourth species), and polyphonic music in more flexible and more abstract styles. We are also interested in examining how other representations and coordination protocols influence the performance of the system.

Acknowledgments

We would like to thank Gil Broza for implementing a C++ version of NetNeg. Dan Gang would like to thank to U.S.-Israel Educational Foundation for supporting him as Fulbright Post-Doctoral Fellow. Thanks for the Center for Computer Research in Music and Acoustics and to the Center for Computer Assisted Research in the Humanities at Stanford University, for the facilities and the pleasant atmosphere.

References

Berger, J., and Gang, D. 1997. A neural network model of metric perception and cognition in the audition of functional tonal music. In *Proceedings of the International Computer Music Association.*

Bharucha, J. J., and Todd, P. M. 1991. Modeling the perception of tonal structure with neural nets. In Todd, P. M., and Loy, D. G., eds., *Music and Connectionism.* M.I.T.

Camurri, A. 1993. Applications of artificial intelligence methodolo gies and tools for music description and processing. In Haus, G., ed., *Music Processing.* A-R Editions, Inc. 233–266.

Goldman, C. V.; Gang, D.; Rosenschein, J. S.; and Lehmann, D. 1999. Netneg: A connectionist-agent integrated system for representing musical knowledge. *Annals of Mathematics and Artificial Intelligence.* in print.

Hilario, M. 1995. An overview of strategies for neurosymbolic integration. In *Workshop on Connectionist-Symbolic Integration:From Unified to Hybrid Approaches at the Fourteenth International Joint Conference on Artificial Intelligence.*

Jackendoff, R. 1991. Musical parsing and musical affect. *Music Perception* 9(2):199–229.

Jeppesen, K. 1992. *Counterpoint The Polyphonic Vocal Style of the Sixteenth Century.* New York: Dover.

Jordan, M. 1986. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of The Eighth Annual Conference of the Cognitive Science Society.*

Miyata, Y. 1991. *A User's Guide to PlaNet Version 5.6.* Computer Science Dept., University of Colorado, Boulder.

Montgomery, T. A.; Lee, J.; Musliner, D. J.; Durfee, E. H.; Damouth, D.; and So, Y. 1992. *MICE Users Guide.* Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan.

Randel, D., ed. 1996. *The New Harvard Dictionary of Music.* Belknap Press of Harvard University Press.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter.* Cambridge, Massachusetts: MIT Press.

Todd, P. M. 1991. A connectionist approach to algorithmic composition. In Todd, P. M., and Loy, D. G., eds., *Music and Connectionism.* M.I.T.