FOUR
Structural Coupling in a Society of Musical Agents

Peter Beyls

4.1 INTRODUCTION

4.1.1 Autopoiesis as an Alternative to Mapping

The current chapter describes aspects of the output section of Oscar, an interactive music system aimed at supporting compelling man-machine interaction in the domain of open, nonidiomatic improvisation. Oscar explores both symbolic musical pattern processing (Cope 1991) and artificial life–oriented sub-symbolic computing (Miranda 2002; Miranda 2004). The system includes a listening component that uses genetic algorithms (GA) to breed the appropriate sensors to capture significant changes in an incoming stream of MIDI data (Beyls 2005a). The musical output adopts the principles of self-organization (Prigogine 1980) in a distributed network of agents engaged in social interaction.

In most a-life research, an agent is characterized by two important features: the level of its functional autonomy and its ability to adapt to large swings in context. Autonomy implies the use of sensors and the creation of relationships between perception and action. An autonomous agent can adapt according to its behavioral history and current perception of the environment. Adaptation is a global emergent quality: surprisingly coherent yet unpredictable behavior follows spontaneously from the expression of a large mass of inter-agent affinities. Emergence is thus a consequence of self-organization, the creation of structural organization without any central design agency. The survival of any biological system seems to rely on the self-organizing properties of its constituting cells, a process referred to as *autopoiesis* (Maturana and Varela 1984). The notion of autopoiesis—which implies perpetual self-production, the continuous creation of new answers while facing an unpredictable environment—is central to this chapter.

In the context of this chapter, musical improvisation is regarded as the expression of variable relationships among a collection of agents, including a human performer. A relationship is thought of as a structural coupling between two interacting entities, which may be either synthetic or organic. Complexity in a living system is viewed as the result of a self-organizing process. External pressure may force changes to the organism, but the palette of potential changes is made available by the process of self-organization, a concept known as structural determinism. The interplay between man and machine, thought of as interacting complex dynamic systems (Thomson and Stuart 1986), thus follows a series of structural couplings. A human being can be seen as an environment with its own structural dynamics to which the machine agents interact. However, human and machine agents are viewed as operationally separate. This implies that the environment cannot simply control the interactions between an organism and that environment. In contrast, the structure of the organism determines what kind of effect an external disturbance will cause. The structural coupling between organism and environment is characterized as a form of influence, rather than instruction. In other words, the focus is on the kind of structures that are affected inside a receiver agent rather than on the kind of information that is transmitted by an agent.

This chapter suggests viewing autopoiesis as an alternative to conventional mapping in systems for interactive music composition. Man and machine are seen as sources of mutual influence rather than control. The essence of life—and, in my work, of deeply engaging musical interaction—is imagined as the existence of continuous structural changes in many dimensions inside the interacting partners, with the understanding that changes happen within the scope of structural limits that defines them guaranteeing survival. When man and machine support rewarding interactions that happen over longer time spans, one can conclude that they are compatible; in other words, they are adapted to each other. This is the process of *ontogenesis*, that is, the individual history of structural changes in an organism while its basic structural organization (its identity) and ability to adapt remain intact.

I am particularly interested in the quality of the dynamics of the interaction in a virtual society of agents. I developed experiments where a simulated external environment produces fluctuating activations impacting on individual agents. Other versions of the same system derive activations from a human musician, but this is beyond the scope of the present chapter. Oscar even develops interesting structures without any external activation. In this case, the society performs as an autonomous system that continuously rearranges the positions of its agent-components. Agents form temporary structural associations that are translated into music as MIDI codes. The music thus reflects the structural changes in time while global structural integrity (survival of the system) is guaranteed.

The remainder of this chapter provides a brief overview of related work and then describes how internal motivations and external influence combine to generate emergent musical functionality.

4.1.2. Related Work in Context

Many theories that address complexity analysis make use of distributed models. Minsky (1985) aims at explaining cognitive processes as globally emergent properties as a consequence of local interaction among simple components. The fluid dynamics metaphor as described by Hofstadter (1995) also is intended to explain cognitive processes. In particular, Hofstadter's notion of the flickering-cluster image is highly suggestive. The stable-seeming fluidlike properties of thought emerge as a statistical consequence of a myriad of tiny, invisible, independent, subcognitive acts taking place in parallel. However, the Oscar system views clusters as unpredictable and variable partial configurations within a fixed pool of agents.

The animated microworld of Ventrella (1993) explores attraction and repulsion in a simulated society. The party-planner model developed earlier by Goldstein (Dewdney 1987) also investigates similar dynamic principles to model social configurations by specifying idealized physical distances between people. In addition, the discipline of a-life has spawned many incarnations of distributed computational models, which vary according to the field of application—for instance, of particles, molecules, agents, or artificial creatures. Particle systems have a long history in the computer simulation of complex natural phenomena. They are instructive early examples of studying complexity by considering relationships among small building blocks, in contrast to using differential equations.

The boids model developed by Reynolds (1987) describes the flocking of birds as an emergent process. Unemi and Bisig (2005) produced a straightforward implementation of the boids model for musical purposes, where the movements of a person interacting with the program (captured with a camera) influence the flocking behavior. The result is a mixed-reality system where global audiovisual behavior follows from a combination of internal and external behavior.

The Swarm simulation environment developed at the Santa Fe Institute (Minar, Burkhart, Langton, and Askenazi 1996) is also a noteworthy example. The swarm idea was adapted for musical purposes by Blackwell and Young (2004): they developed an interactive music improviser. Their system maps the positions of particles (or boids) to positions in a MIDI space. An external human improviser may act as a temporary target for the swarm. Style scripts provide additional parametric control over the nature of the interaction, thus further conditioning the musical output. The Eden interactive installation project (McCormack 2003) is an example of distributed audiovisual intelligence. This system features agents that listen, act, and otherwise evolve in a virtual world connected to the physical world via infrared sensors. It is also worth mentioning the work by Dahlstedt and McBurney (2006), who developed a system that addresses musical autonomy in a society of software agents programmed to generate music. Finally, I have introduced an artificial chemistry-inspired molecular collision model of musical interaction (Beyls 2005b).

4.2 IMPLEMENTATION OF OSCAR

4.2.1 Definition of an Agent

An agent may be either a graphic object in two-dimensional (2D) space or a MIDI player object. Noteworthy instance variables are:

- Physical position: This is the position in the action pane (of the system's graphic user interface (GUI)) and angle of movement. These values are equivalent to the initial conditions of a complex dynamic system
- Energy level: Agents dissipate less energy when their position remains stationary than when they are moving. An agent becomes stationary either when the net values of all affinities impacting it sum to zero, or when its energy level is lower than a given threshold. The energy dissipation factor when the agent is active and the recovery factor when it is asleep are unequal, and this inequality contributes to global nonlinear behavior
- First critical distance (parameter *critical-distance-1*): This determines the sensitivity of an agent in communicating with any neighbor
- Second critical distance (parameter *critical-distance-2*): All agents within a radius of the second critical distance of an agent are considered neighbors of that agent, corresponding to the CritGap fader on the system's GUI (see Figure 4.4 below)
- Activation: The activation is a signed quantity (ranging from -100 to +100) intended to function as a source of qualitative information. In particular, it has a great impact on the musical interpretation of the clusters with which an agent happens to be associated. Activation is subject to changes owing to pressure from the environment, for example, a human musician. However, this chapter focuses on autonomous behavior following experiments with fixed activations or activations that fluctuate randomly. A probabilistic algorithm is used to generate a slowly changing activation level that moves within values ranging from -100 to +100
- Orientation: This designates two types of interaction between any two agents and is represented by a variable, which can have a value of either 0or 1. If the value is equal to 0, then the agent will address the stress problem by attempting to move to an alternative location that would result in less stress impinging on it. Conversely, if the value is equal to 1, then the agent will engage in interactions with all other agents in its immediate proximity
- Personality dataset: This is a list of pitch intervals, durations, and velocities from which sublists are taken to construct melodies

As a simulated physical object, the agent moves in two-dimensional space while dissipating energy proportional to the distance traveled. When energy becomes lower than a given threshold, the agent enters a sleep state for predefined period of time, measured in cycles of iteration. When it wakes up, the agent's new full energy level is set according to an external global parameter. Both the energy ceiling and the length of the sleep cycle introduce nonlinearity and unpredictability in the system. As agents condition each other's movement, they also influence each other's energy consumption.

Every agent has two options for local interaction, controlled by the value of its orientation bit. If this value is equal to 0, then it will try to lower its perceived stress by changing its physical position. Otherwise, if the orientation bit value is equal to 1, then it will interact with some of its neighbors. The list of actual neighbors is computed by considering all agents within a critical distance from a reference agent.

Figures 4.1 and 4.2 show two configurations of sixteen agents each, considering the activity in agent a1. In Figure 4.1, the orientation bit of a1 is switched off (i.e., equal to 0), whereas in Figure 4.2 this bit is switched on (i.e., equal to 1). The status of the orientation bit is reflected in the color of the agent: yellow corresponds to being switched off and blue to being switched on. The curved arrows represent the inter-agent affinities. However, for clarity, not all affinities are shown. For example, the curved arrow labeled D78 denotes the preferred social distance expressed from agent a7 toward a8. The arrow labeled D87 is the expression of a8 toward a7. One may easily imagine complex behavior when the forces implied in D78 and D87 are very different.

Both alternative settings of the orientation bit are detailed next.

4.2.2 Behavior According to Social Stress

Agents are viewed as a society in which a social context exists. This context is in constant flux because of a specific affinity that every agent expresses toward every other fellow agent. Global affinity for a single agent is articulated as a force imposed upon that agent such that it is at a certain preferred distance from all other agents. Because these affinities might conflict, the result is an emergent push-pull behavior. In addition, the net impact on any given individual agent is the result of contributions of affinities from all agents that are not asleep. An agent thus moves spontaneously because it aims at minimizing its own individual tension relative to the rest of the society.

The algorithm works as follows: every agent considers a position at a distance (given by the delta parameter, typically a value between 5 and 20) relative to its current spatial position in the two-dimensional field of action. As depicted in Figure 4.1, eight neighboring positions are considered, and the new tension is computed and catalogued. Finally, the agent will select the position that guarantees the least

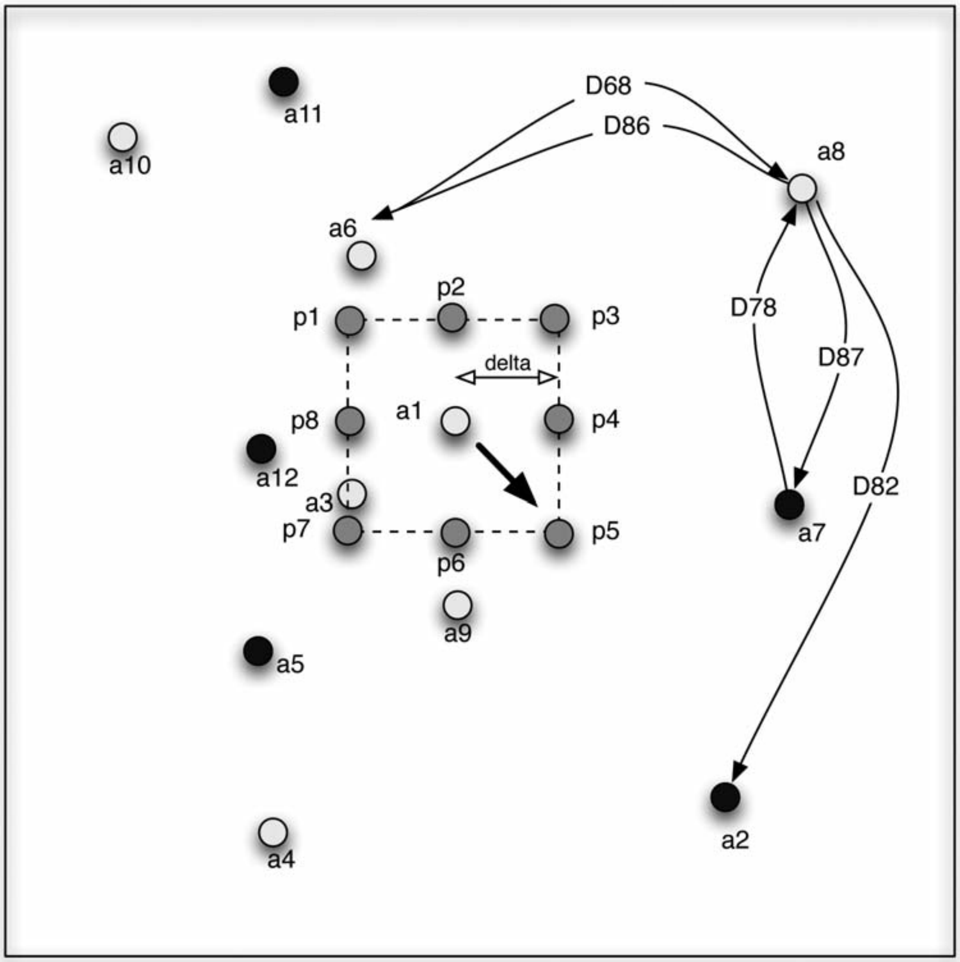

FIGURE 4.1 Computing the local stress of an agent given the current global configuration.

stress. Since all agents follow the same rule, the network of mutual tensions is pulled toward a local minimum value. In other words, all agents contribute toward minimizing the stress in their society; this is referred to as *emergent functionality*. Affinities for sixteen agents are represented by a matrix of 16 × 16 elements holding values between 0 and 400; this matrix can be inspected and edited by the user at will. High affinity values will produce the effect of agents spreading away from each

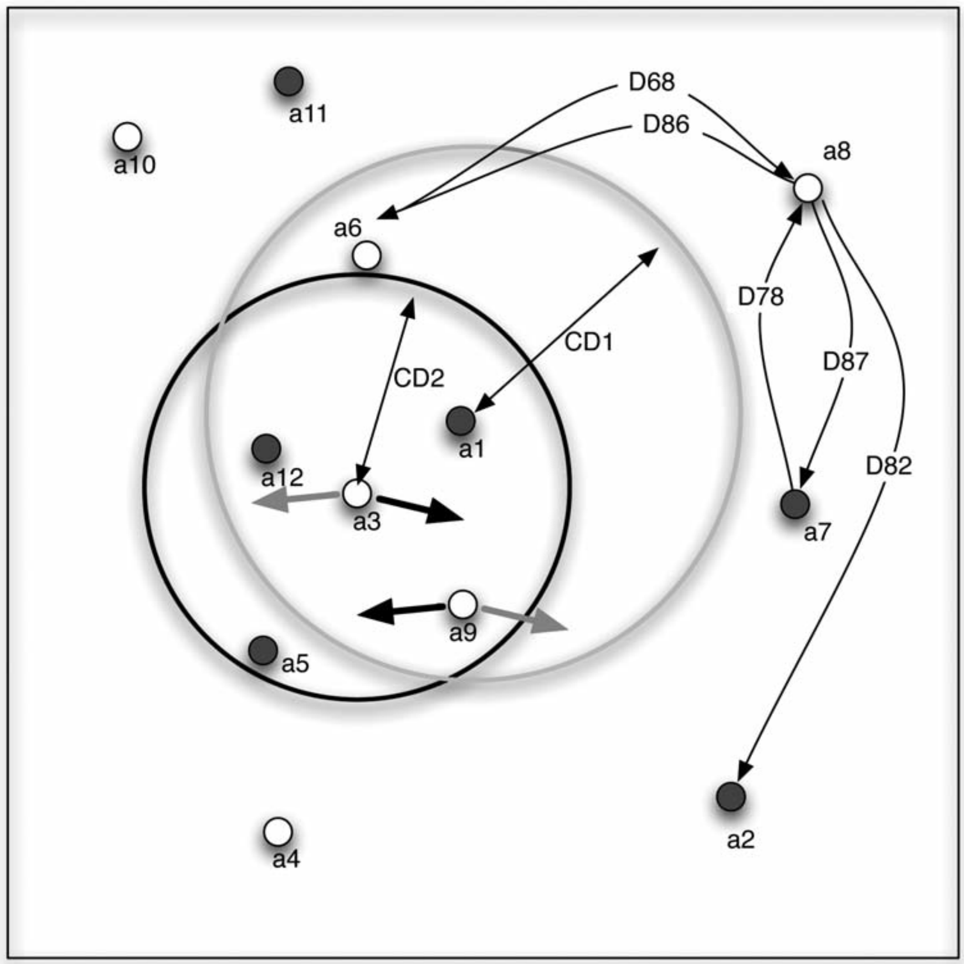

FIGURE 4.2 Agents a3 and a9 engaged in local information interchange.

other. Conversely, small values will tend to cause the agents to form clusters. By intuition, one may understand that a particular mix of random values will implicitly impose complex, rather animated behavior. In this case, the society becomes a complex dynamic system that exhibits all kinds of cyclic and chaotic attractors. The society as a whole may produce a wide range of interesting oscillations depending on the specification of the matrices. Experiments 4 and 5 below reveal different evolutions of structural changes over forty cycles.

4.2.3 Behavior According to Local Interaction

When the orientation bit is switched on, an agent will consider its proximity and possibly engage in a bonding process. All agents within a given first critical distance are addressed: the agent acts as a catalyst in the sense that only its neighbors will execute local rules, not the agent itself. For every agent within that range, its respective neighbors are collected in turn. The collect-neighbors function has its own sensitivity parameter: the second critical distance. Referring to the situation depicted in Figure 4.2, as a neighbor of agent a1, agent a3 will interact with one of its neighbors, in this case, agent a9.

Every agent in the subset (a3, a6, a9, a12) will exchange information with one of its own neighbors; they swap the current value of their private angles of movement. Strange behavior emerges because this procedure acts as a specialized delay line. In addition, the difference between the two critical distances has a considerable impact on the resulting complexity. One may tune both sensitivities independently, in this way influencing global behavior without the option of exercising precise control. In addition, by changing the relative proportions of the agents' orientations, either (a) structural development in the society will either follow from the expression of affinities or (b) any two interacting agents will interfere with a subset of their immediate neighbors and exchange information on how they move in space.

A *sparkle* parameter acts as a probabilistic source of noise in the system. It is intended to keep the system away from point attractor behavior and follow the knowledge that organic networked systems (such as human brains) produce random firing patterns even without any apparent external stimulation (Harth 1995). When sparkle fires, the orientation bit of the current agent is temporarily inverted.

Two performance modes exist: (a) the agents' animation process and (b) the MIDI player process. They can either be synchronized or run independently. By definition, the animation process (running the simulation and the real-time visualization) updates at a chosen rate, typically 500 msecs to 5 secs. The player process computes a new melody as soon as the previous one has finished playing. When the performance modes are run independently, many simulation cycles may pass while a melody is played; thus, when the next one is computed, its contents will echo how much the world has changed since it was last sampled. Otherwise, when the performance modes are run in synchrony, the simulation waits until the player has finished playing the current melody. In this case, changes in the society will immediately be reflected in the music generated.

An attempt is made to describe the algorithms (Figure 4.3) in pseudo-code without oversimplification; please refer to the appendix at the end of this chapter. It begins by explaining the stress and orientation–bit conditioned animation procedure (see Figure 4.18 below). Then it describes the musical interpretation of the clusters (see Figure 4.19 below), followed by an explanation of how the reference or backbone melody is created (see Figure 4.20 below).

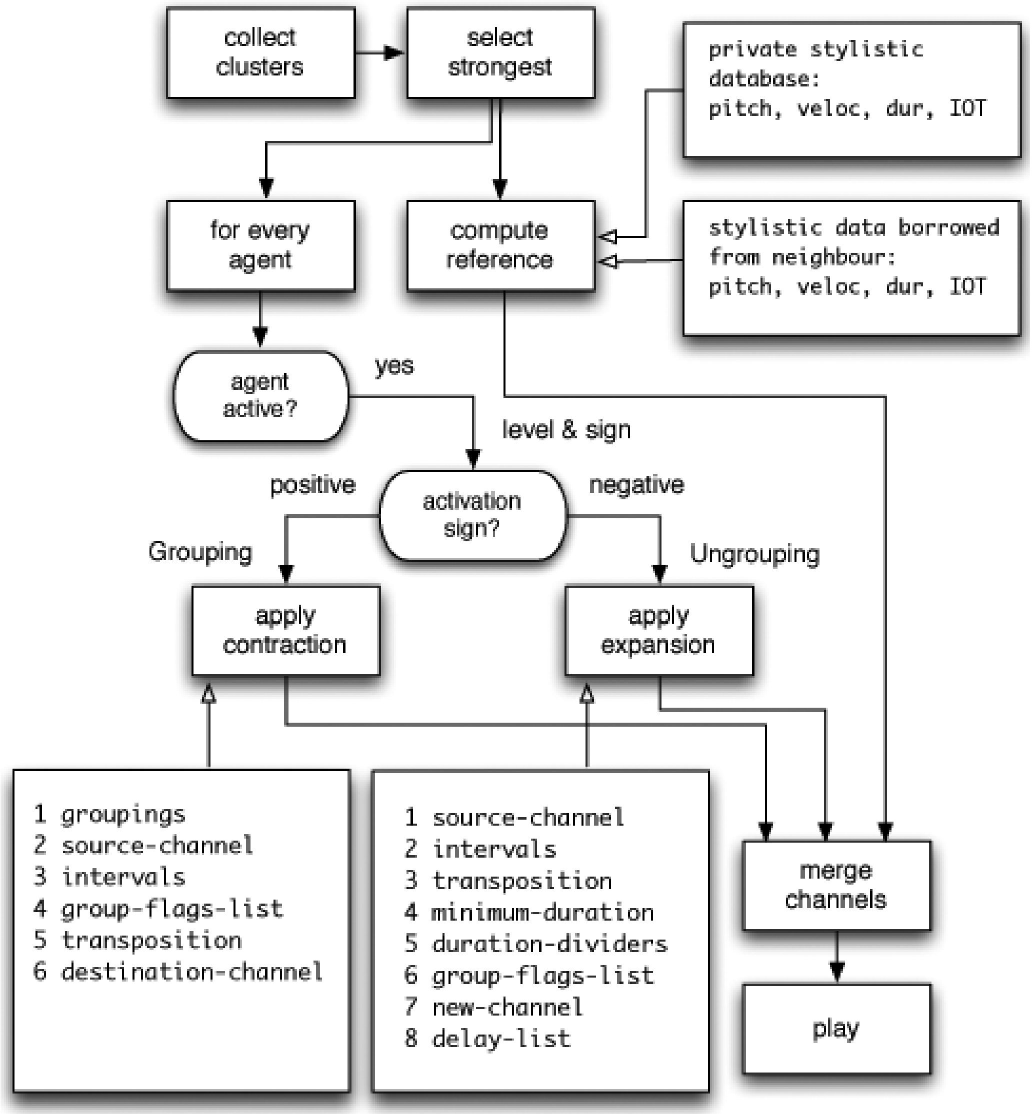

FIGURE 4.3 Musical interpretation of clusters according to sign and amplitude of agent activations.

4.3 MUSICAL ARTICULATION

4.3.1 Interpretation of Temporary Clusters

Oscar produces music based on the history of the relationships between individual agents—that is, the structural changes in the system in time. The resulting music is computed as follows: Every agent first creates a list of its neighbors. In order to evaluate a variety of sensitivity schemes, the value of the first critical-distance parameter is tunable: a bias value plus a diversity value, which is a deviation from bias in percentage points. Next, individual groups of agents are isolated according to their neighbors. The result is a variable series of temporary structures referred to as clusters. Clusters are visualized by drawing segments between agents falling in each other's zone of influence—that is, when the distance is below a given critical-distance threshold. Such clusters are viewed as emergent structures. They reflect how individual agents create temporary alliances, very similar to the notion of "flickering clusters" suggested by Hofstadter (1995, p. 3).

Figures 4.4 and 4.5 provide snapshots of typical configurations of sixteen agents. An agent is represented by a colored dot; the reference agent is red, and other agents are colored according to their orientation bit: green corresponds to the bit's being switched off and light blue to its being switched on. The label of the agents contains three elements: the ID (ranging from 0 to 15), the activation level (ranging from -100 to +100), and the current number of neighbors.

The affinity matrix underpinning the configuration in Figure 4.4 is filled with values equal to 100. This implies that all agents express the same preferred social distance toward each other. The agents blend into an emergent pattern that looks like a square rotated by 45 degrees. The four clusters are built as collections of two, three, four, and five agents, respectively. The first critical distance (CritDist on the GUI) is equal to 13.

Figure 4.5 portrays a situation with random contents of the affinity matrix: it is filled with random values ranging between 40 and 400. Two clusters emerge given a second critical distance value of 48. Each cluster contains seven agents.

The energy of all clusters is computed by averaging the energy of all their constituent agents. The cluster with the maximum energy is further selected; see the information flow shown in Figures 4.6 and 4.7. Note that small clusters (with few agents) may provide more energy than large ones; size and energy are thus interacting in subtle ways. A reference melody is created that will serve as a musical backbone (as explained below). Then the system considers the other agents in the cluster and examines their activation. If the status of a remaining agent is active, the agent contributes additional events to the musical backbone. Two options exist: expansion and contraction. Expansion creates a sequence of supplementary events from a single source event. Contraction, in contrast, creates a single new event from the data supplied by a group of existing backbone events. The results coalesce as a form of emergent musical orchestration.

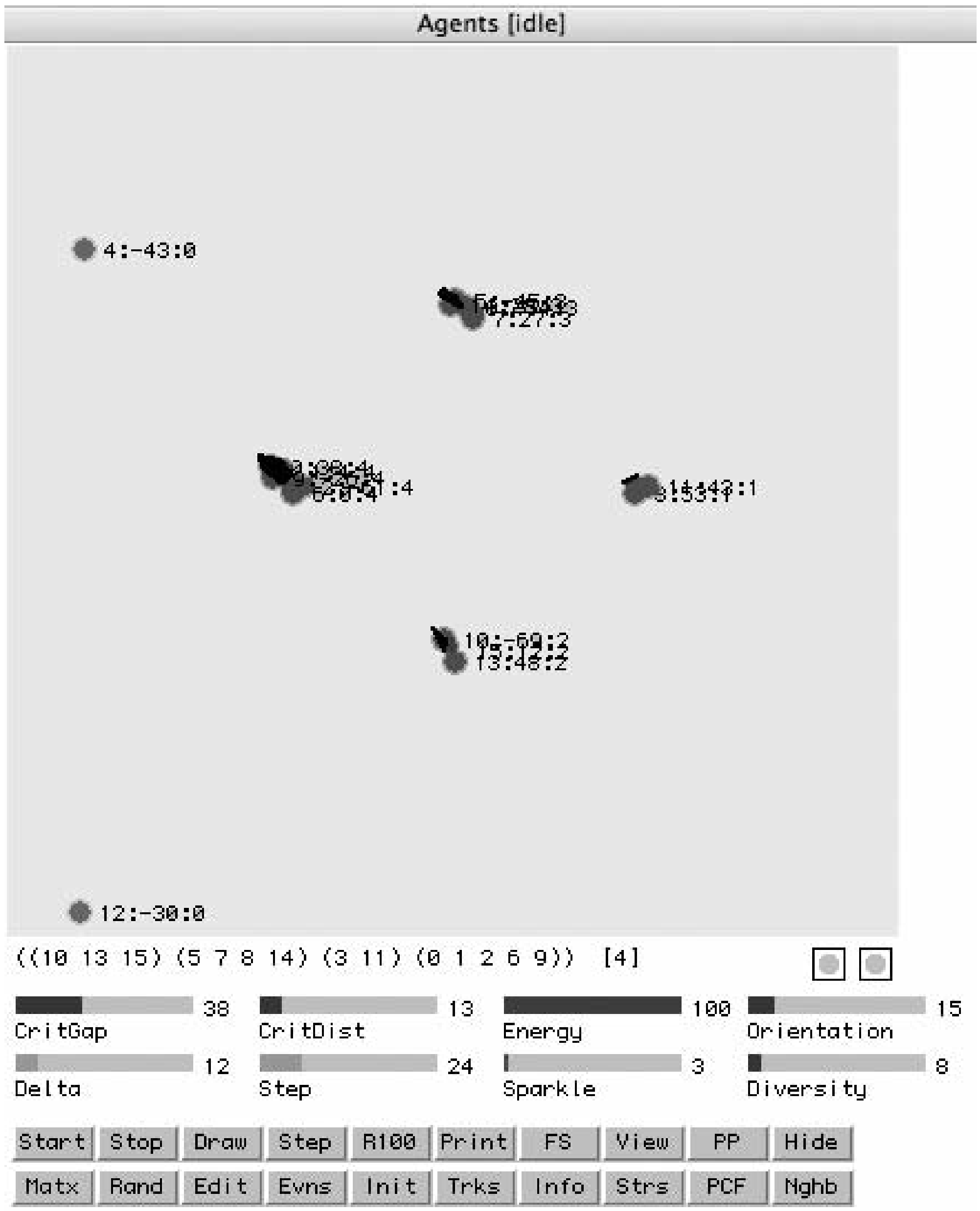

FIGURE 4.4 The main GUI showing action pane with emergent configuration of four temporary clusters; affinities matrix is filled uniformly with fixed values of 100.

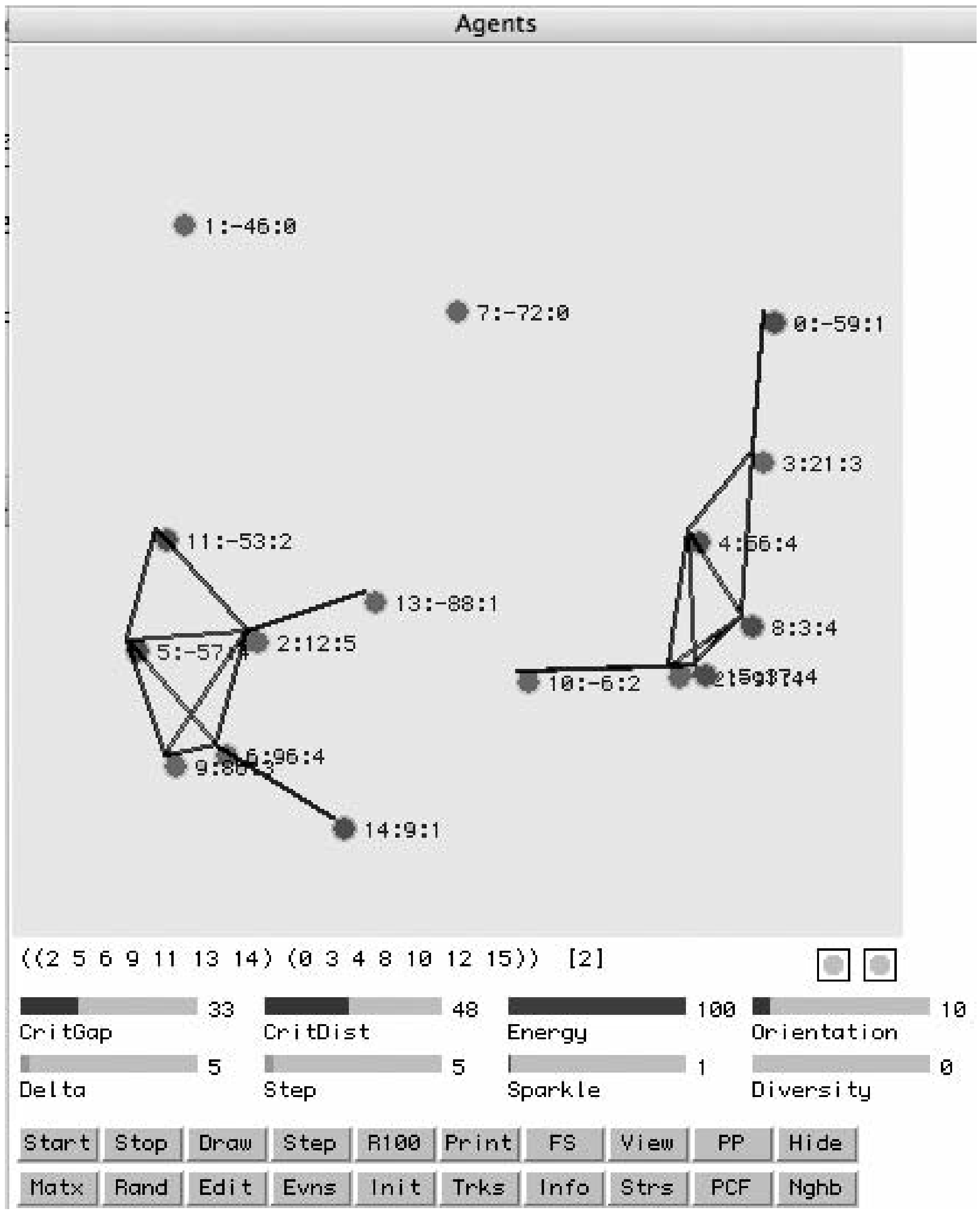

FIGURE 4.5 The main GUI showing action pane with two temporary clusters; affinities matrix is filled with random values between 40 and 400.

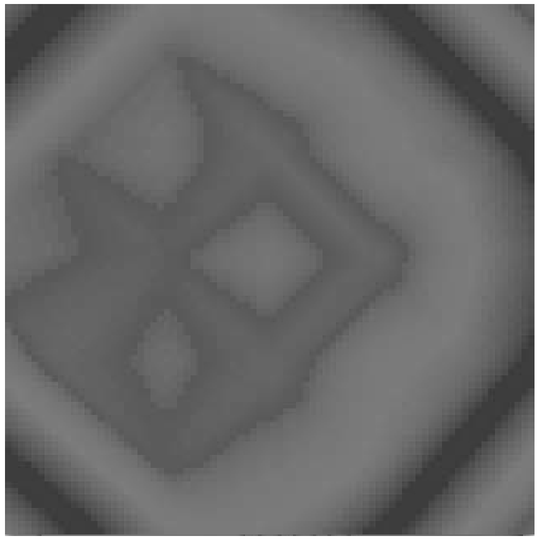

FIGURE 4.6 Tension landscape corresponding to the agents' configuration in Figure 4.1.

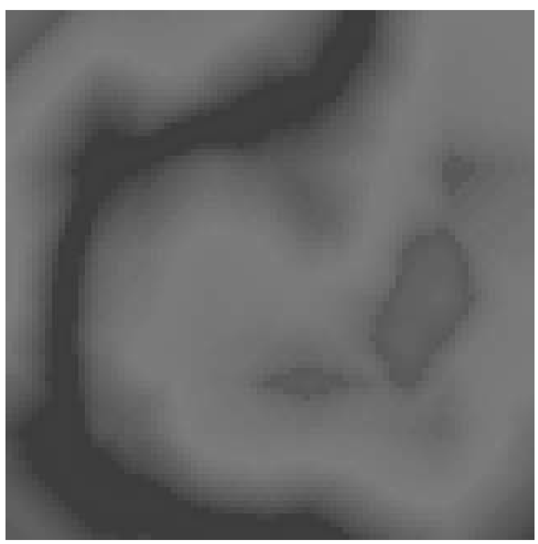

FIGURE 4.7 Tension landscape corresponding to the agents' configuration in Figure 4.2.

The actual parameter settings for the synthesis of musical events (low level) used in the simulations described here are less decisive than the parameter settings affecting the actual simulation (high level). Parameter ranges for musical events (i.e., pitch intervals, lists of potential MIDI velocities, durations, and inter-inset–inter-onset times) are typically generated from an algorithm involving weighted randomness. In contrast, all parameter settings that influence systems behavior (i.e., the affinities matrix, the settings of the orientation bits, and the critical distances) have strong impacts on the complexity of the spatiotemporal structures and consequently on the articulation of the music being generated. In other words, the music continuously reflects temporary arrangements of low-level musical detail (the private parameter lists inside any agent) created by forces at a much higher level of abstraction.

4.3.2 Computing a Backbone Reference Melody

The last section of the pseudo-code in the appendix (Figure 4.20) describes the process of creating the reference melody. The ID of the first agent in the strongest cluster (the reference agent) decides the MIDI channel (from 1 to 8) of all events to be added to the reference melody. In addition, private stylistic data may be borrowed from that agent or from a single neighbor agent. The temporary organization of the clusters (the structure of the cluster) and the specific agents that participate will influence the construction of the melody. A selection scheme using a 4-element bit pattern is used to guide the selection of a given parameter from either source. The selector holds all six possible 4-bit patterns, with 2 bits switched on. This results in six ways to fuse influence from two sources of parameters. These parameters are lists of pitch intervals, durations, and velocities. All lists are typically of different lengths. Because the lists are used cyclically, to produce a desired number of events, the information in them will interfere in subtle ways because some lists may actually synchronize (because they are of the same length), while other lists may slowly move in and out of phase (because they are of significantly different lengths). The energy of the reference agent will influence the number of events generated as well as the density (articulation by way of rests) of the events.

Once the backbone reference is available, it will function as a source for additional parallel events playing on different MIDI channels. The remaining agents in the current cluster may contribute events as a function of their activation. The sign of the activation of the agent acts as a switch for either the contraction (positive) or expansion (negative) procedure to be selected. The external environment typically provokes agent activation. However, the simulations discussed in this chapter make use of a highly flexible internal activation source, which will be described later.

4.3.3 Event Generation by Contraction

Contraction is computed using a grouping algorithm; it typically produces additional events characterized by features of specific groups of backbone events. Its parameters include groupings list, source channel, intervals, group-flags list, transposition, and destination channel (see again Figure 4.3). All argument lists are of different and arbitrary size. Items are addressed cyclically using an incremental global pointer calculated as the modulo of the length of the list in question. Interestingly, this implies that arguments interact in irregular ways. A minority of argument lists are created on the spot from a pool of candidate values: as a whole, these values constitute a small database of stylistic information. At this point we hit the atomic level of our system, one of the very few instances where factual knowledge (expressed as numerical data) is specified by an external human designer. Otherwise, stylistic data is borrowed from the private data of every subsequent agent in the cluster: pitch intervals, durations, velocities, activation, and energy level.

The algorithm first scans the backbone and creates groupings of two, three, or four source events (the groupings parameter), summing the duration of the groups for use by the shadow events to be generated. It considers only events in which the MIDI channel matches the source channel. In addition, pitches are offset (from the first pitch in the group), guided by the intervals parameter, while the transposition argument provides a global pitch offset. A variable named group-flags list refers to a Boolean list conditioning the new event to be added to the backbone or not, using the specified destination channel. The length and the number of true items in the group-flags list variable are proportional to the absolute value of activation of the agent.

4.3.4 Event Generation by Expansion

The expansion algorithm is the functional opposite of the contraction algorithm: it spawns many events from one source event. It applies eight parameters: source channel, intervals, transposition, minimum duration, duration dividers, group-flags list, new channel, and delay list (see again Figure 4.3). All events whose duration is equal to or higher than a given minimum value are collected, conditioned by the group-flags list parameter. The particular mix of Booleans may thus partition the source material in asymmetrical ways. The duration dividers (typically two, three, or four) split up the collected durations into many new values. Now the start time of the new event is postponed by a delay, the product of the new duration and the value taken from the delay-list argument. This method yields automatic synchronization: start times and/or end times of source and destination events will align in most cases. The application of intervals and transposition arguments is similar to that in the contraction algorithm introduced in the previous section.

4.3.5 Agent Activation via the Threshold Algorithm

The work reported here focuses on the activity inside a collection of agents; the agents' activations are derived from a simulated environment rather than a human interactor. I draw on a threshold algorithm to function as a qualitative source of uncertainty. The algorithm exploits just one parameter, the threshold percentage, and an internal status variable (0 or 1). A list is created from all clustered agents. While scanning that list, a random value between 0 and 100 is generated. If this value is lower than the value of a threshold-percentage parameter, then the status variable is inverted. Accordingly, if the value of the threshold-percentage parameter approaches 100, then the status variable will move back and forth, producing a predictable oscillating pattern. If the value of the threshold-percentage parameter is relatively (statistically) low, then it will take a long time for the status variable to change. Both extreme values of the threshold-percentage parameter thus produce predictable results. By tuning in between 0 and 100, one can generate a bit stream of variable predictability. Next, we use the value of the current-status variable to address the activation of the agents. When the status is equal to 0, the activation is lowered by 10 percent of its current absolute value, clipping the activation at the smallest amount, that is, -100. When the status is equal to 1, the activation is boosted by 10 percent up to a ceiling of +100. In addition, a value of 1 is implied in both cases because when the activation nears a value of 0, the changes stabilize. Also, when activation hits -100, that activation is centered at a random value between -40 and -60. When stuck at +100, a random value between 40 and 60 is attributed. The result of this scheme is that large activation values (either positive or negative) will fluctuate faster than small ones. The activation of the agents will echo this subtle activity of the threshold algorithm.

4.4 EXPERIMENTS

Figures 4.6 and 4.7 exhibit two different stress views, both as seen from the perspective of a single reference agent. In Figure 4.6, the affinities matrix is filled with a fixed value of 100 (configuration shown in Figure 4.4), whereas in Figure 4.7 it is filled with random values ranging between 0 and 400 (configuration shown in Figure 4.5). The upper limit is derived from the size of the action pane. Figure 4.6 shows a nearly symmetrical, diamond-like shape. This emergent pattern reflects the matrix's uniform contents. In contrast, Figure 4.7 shows the effect of a random distribution in the matrix.

The total stress of the reference agent is computed by summing the differences between the ideal position (guaranteeing a minimum stress value) and the current position for all agents in the society. This provides the magnitude of stress of a single agent toward all other agents given their current physical positions.

In order to visualize this global view, we can create a stress-view array of 80 × 80 elements, which will map to the 400 × 400 pixels action pane. The reference agent visits regular positions on the pane in coordinate steps of 5 pixels, implying a lower resolution for this simulation. A global picture of the tension experienced by a single agent thus accumulates in the stress-view array. Next, it is normalized to values between 0 and 100. The integers in the resulting array serve as pointers in a color lookup table that interpolates between light blue and deep red. The resulting image shows a two-dimensional tension landscape. It reveals the nonlinearity captured in the affinity matrix. More precisely, it shows the relative target tension for all possible positions (80 × 80, spread out over 400 × 400 pixels, implying a resolution of 5 pixels) of a single reference agent.

The two experiments described below are discussed in more detail in (Beyls 2009). They use random activations (between -100 and +100) defined at instantiation time. The activations remain unchanged during the course of all three experiments. As an example, please refer to Figure 7.16 on page 320 of (Beyls 2009), which displays four hundred generations of sixteen interacting agents. Five values are traced: *x* and *y* position in two-dimensional space (red and blue, respectively), angle of movement (green), energy (gray), and the bonding history (orange). Bonding amplitude refers to the number of agents interacting at every moment in time. All system parameters remain unchanged throughout the simulation. Both critical distances are equal to ten. The agent's orientations are also fixed at (1100010100100100). Thus only six out of sixteen agents will exchange information with their respective immediate neighborhoods. The positions of the agents are occasionally disturbed by external action, but the affinity matrix remains unaffected. It is possible to observe the effect of the couplings expressed in the matrix. Oscar accommodates the disturbance and resettles into orbits of variable periodicity. When the system is pushed out of relative stability, it evolves from erratic to more regular oscillations, occasionally hitting a point attractor. Local stable, highly periodic oscillations occur over extended periods of time; note the push-pull activity between agents 3, 4, 5, and 7, respectively. As for the results of the second experiment, please refer to Figure 7.17 on page 321 of (Beyls 2009). The density of positive values in the affinity matrix is only 15 percent. Orientations are (0111010101011011). One can observe four behavioral phases: initial chaotic interactions, then only agent 4 engages in a steady bonding process while most of the other agents remain stationary. Next, a cycle attractor is hit with only agents 5 and 11 interacting. Finally, when the first critical distance is increased slightly, the oscillations become less periodic, and the bonding process starts peaking again, albeit in a limited number of agents. The variations in the energy profiles reveal evidence that energy dissipation and bonding density interact.

The next two experiments demonstrate the behavior of the system under specific parametric conditions; a visualization of internal structural changes and the resulting music are shown.

Figure 4.8 shows a free-running simulation over forty generations, with a snapshot of every temporal structure. The affinity matrix is filled with random numbers

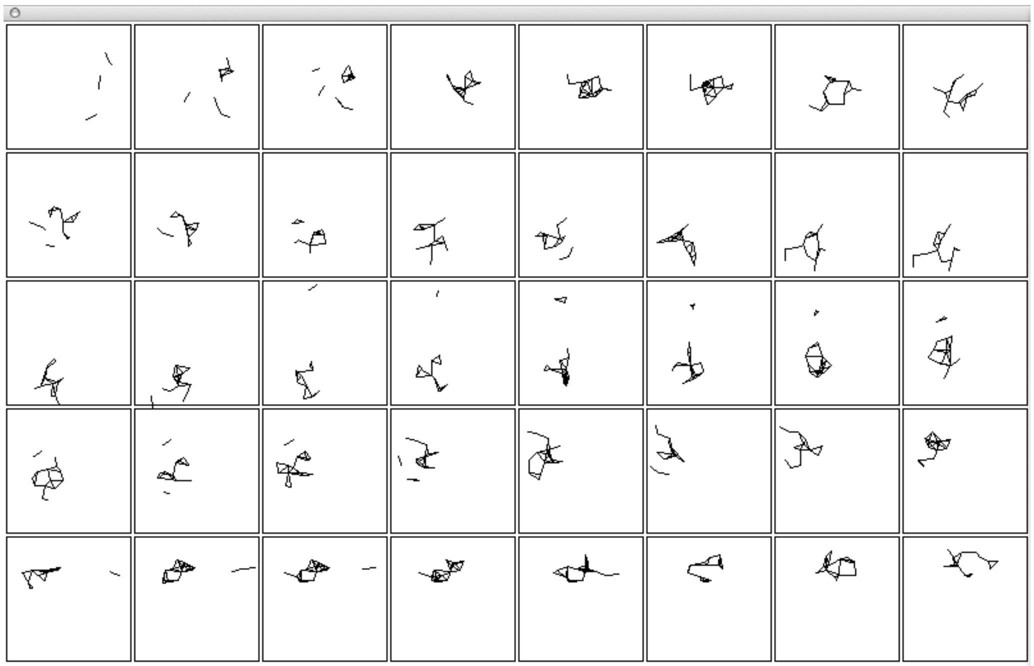

FIGURE 4.8 Structural changes with random contents of affinities matrix from experiment no. 4.

ranging from 50 to 200 (the upper limit is half the size of the action pane). All agents receive an initial random energy value ranging from 50 to 100. All orientation bits are set to 0, so only the critical-distance parameter is significant and no bonding can occur. The critical-distance value is equal to 27 for every agent. Delta and step parameters are equal to 5. The frames of the behavioral history are read from the top left to the bottom right.

The simulation started with random positions of all agents. Three clusters of two agents each are visible in the first generation. All agents moved one step according to the social forces expressed in the affinities matrix. The average pull is strong, and all agents gradually collided in a single cluster. From there on, the local push-pull activity on the individual agents disintegrated the cluster and caused it to shift in space.

Figure 4.9 documents an experiment in which the affinity matrix is filled uniformly with the value 150, so every agent expresses the same social-preferential distance toward every other agent. The orientations vector is (0010101100110000). Thus six out of sixteen agents may engage in a bonding process with a neighbor

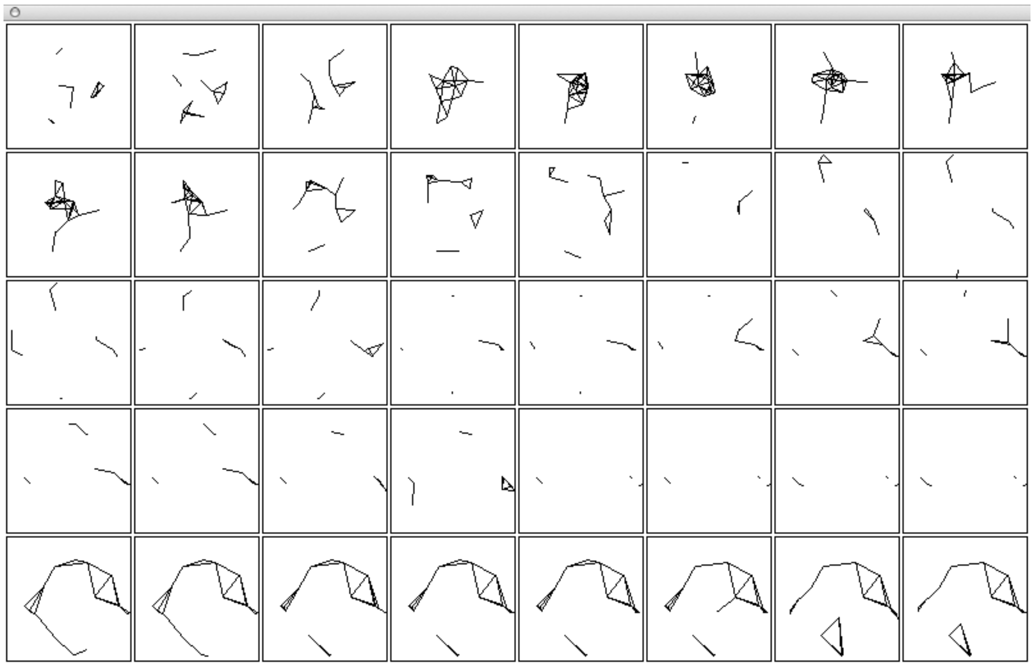

FIGURE 4.9 Structural changes with uniform contents of affinities matrix from experiment no. 5.

agent within the second critical distance value (equal to 22 in this run) from that agent. The value of the first critical distance is equal to 39 for every agent. Frame 1 shows the initial random positions of the agents. All agents express strong attraction at the start of the simulation, resulting in the creation of a single complex cluster. This cluster gradually disintegrates and transforms itself into a diamond-like shape, after which it remains relatively stable for the next few generations. The first critical distance is increased to 67 at generation 33, and a single cluster emerges. Because the affinities matrix remains uniform, the cluster does not change dramatically over the next generations.

The musical interpretations of Figures 4.8 and 4.9 were captured as MIDI files; the two resulting scores are respectively 30 and 29 measures long. Figures 4.10, 4.11, 4.12, and 4.13 show the resulting score orchestrated over eight MIDI channels, according to the contraction/expansion algorithm described earlier. The temporal articulation of the density of the events and the way in which they relate vertically reflect the structural couplings between the agents over time. Only measures 5–10 are shown, echoing the activity in the first half of the squares of the second row of

FIGURE 4.10 Music produced with experiment no. 4, mm. 5 and 6, respectively.

05_789-71_Chap4_pp53-86 3/24/11 10:39 AM Page 73

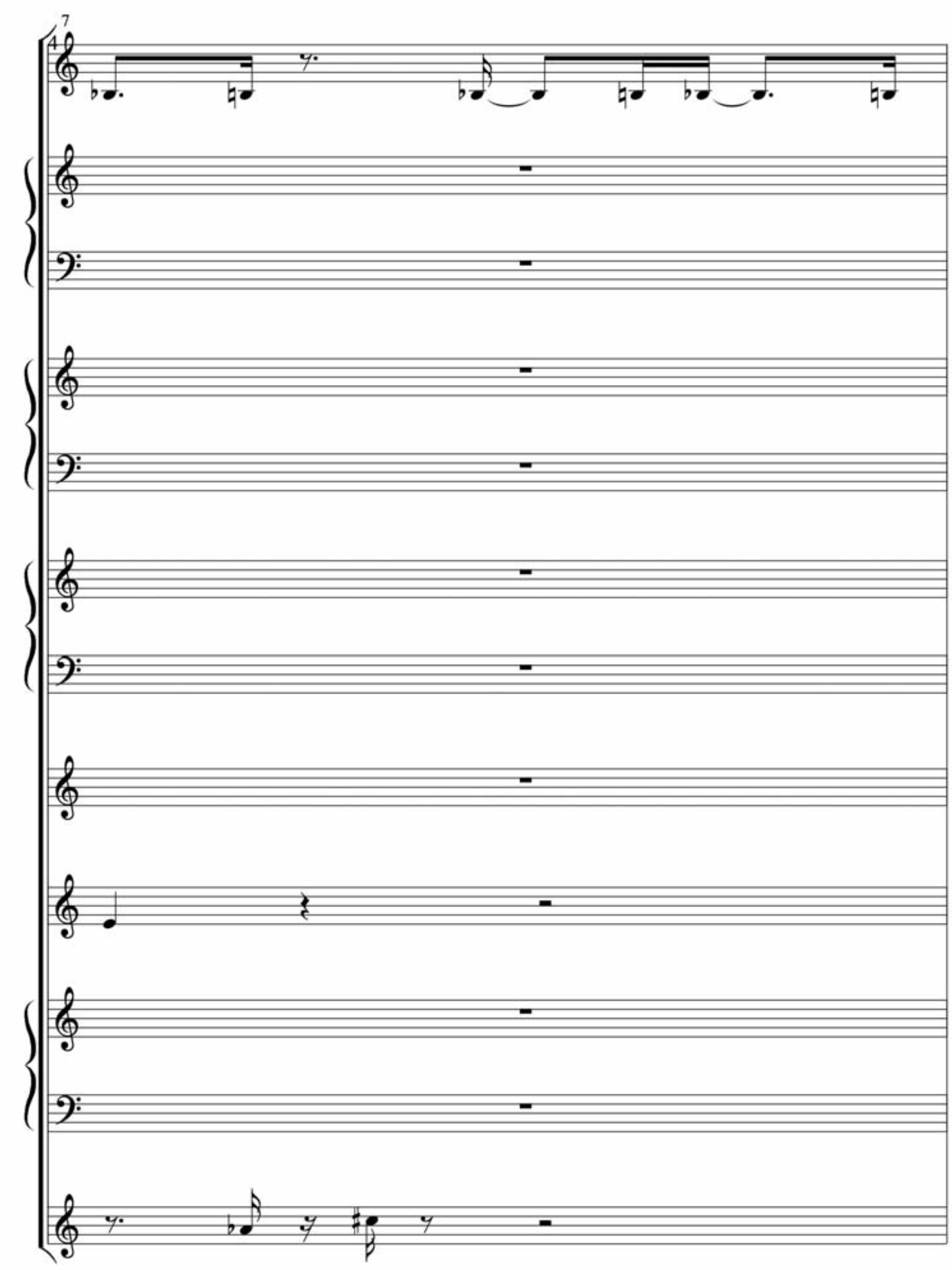

FIGURE 4.11 Music produced with experiment no. 4, m. 7; continuation of Figure 4.10.

05_789-71_Chap4_pp53-86 3/24/11 10:39 AM Page 74

FIGURE 4.12 Music produced with experiment no. 4, m. 8; continuation of Figure 4.11.

05_789-71_Chap4_pp53-86 3/24/11 10:39 AM Page 75

FIGURE 4.13 Music produced with experiment no. 4, mm. 9 and 10, respectively; continuation of Figure 4.12.

Figure 4.8. Notice that the picture gradually evolves from a fractured into a more closed cluster configuration. This is clearly reflected in the pattern of the resulting MIDI events. The incremental nature of the events follows the increments in structural coupling.

Figures 4.14, 4.15, 4.16, and 4.17 show the second score. Only measures from 5–13 are shown; they portray the musical rendering of Figure 4.9, starting halfway through the second series of squares in the simulation. The cluster in Figure 4.9 is opening up. Note the shift from activity in six channels, as shown in Figure 4.14, to activity in the two uppermost channels only, as shown in Figure 4.17.

The scores of both experiments four and five document the formation of intricate musical patterns. A relatively high degree of synchronization is observed between the eight individual MIDI channels. The expansion and contraction algorithms both guarantee perfect alignment of additionally generated events issuing from the backbone. Because the backbone may actually include relatively few events, when using the expansion algorithm, the additional parallel events may be of a higher density than the source channel (that is, the backbone melody).

It is acknowledged that the individual role of the backbone and the impact of the expansion/contraction algorithm are not entirely clear; all contributions mingle into a single complex outcome. But this is not a problem with which we need to concern ourselves here. The emergent result is what interests me in the first place: a complex musical organization that mixes procedural influences from a single source track into a global polyphonic musical fabric.

4.5 CONCLUSION

One may think of the agents as virtual musicians with private MIDI channels, floating in two-dimensional space. They create highly complex nonlinear temporal superstructures of great plasticity. This desirable nonlinear behavior is a consequence of the inter-agent affinities, the flux of external activation, and the dissipation of energy. Thus a simple control structure inspired by biology supports complex musical interaction. Also, we may exert influence over the quality of the internal interactions by using only a very limited number of control parameters, particularly the critical distances, the affinity matrix, and the orientations vector. In terms of musical control, there is a strong control structure implying coherent, minimal control over maximal complexity. Because all parameters are continuous rather than discrete, the implied behavioral space is virtually infinite. Yet one can easily tune the system to function within specific behavioral boundaries. Subsequently, powerful emergent musical “personalities” do surface from the interactions, both those that involve human interactors and those that do not.

FIGURE 4.14 Music produced with experiment no. 5, mm. 5 and 6, respectively.

FIGURE 4.15 Music produced with experiment no. 5, mm. 7, 8, and 9, respectively; continuation of Figure 4.14.

FIGURE 4.16 Music produced with experiment no. 5, mm. 10 and 11, respectively; continuation of Figure 4.15.

FIGURE 4.17 Music produced with experiment no. 5, mm. 12 and 13, respectively; continuation of Figure 4.16.

In summary, Oscar can be characterized as follows:

- It supports a strong musical control structure by considering mutual influence rather than explicit control
- It provides musical behavior that seamlessly integrates subtle external activation in an otherwise completely autonomous system
- It supports self-organizing behavior, which is instrumental to the synthesis of both interesting internal inter-agent interaction and rewarding man-machine interaction

In the context of this work, improvisation is a process of perpetual renewal. Oscar is to a great extent inspired by the theory of autopoiesis and the fluid dynamics theory developed by Hofstadter (1995). From the musical point of view, the evolving associations between individual and group behavior are particularly interesting to me as a composer. Even without any external activation, the society of agents can sustain complex interactions for hundreds of generations.

4.6 REFERENCES

Beyls, P. (2005a). "Evolving Adaptive Sensors in a Synthetic Listener." *Proceedings of International Computer Music Conference (ICMC 2005)*, Barcelona, Spain, pp. 563–66. San Francisco: International Computer Music Association.

Beyls, P. (2005b). "A Molecular Collision Model of Musical Interaction." *Proceedings of 8th Generative Art Conference*, Milan, Italy, pp. 374–86. Milan: Alea Design.

Beyls, P. (2009). *Music As Complex Emergent Behaviour. An Approach To Interactive Music Systems*. PhD Thesis, University of Plymouth, UK. Available on-line http://neuromusic.soc.plymouth.ac.uk/Airport/Beyls_PhD230210.pdf (Assessed on 23 February 2011)

Blackwell, T., and Young, M, (2004). "Self-Organised Music." *Organised Sound* 9(2):123–36.

Cope, D. (1991). *Computers and Musical Style*. Middleton, WI: A-R Editions.

Dahlstedt, P., and McBurney, P. (2006). "Musical Agents: Toward Computer-Aided Music Composition Using Autonomous Software Agents." *Leonardo Music Journal* 39(5):469–70.

Dewdney, A. K. (1987). "Computer Recreations: Diverse Personalities Search for Social Equilibrium at a Computer Party." *Scientific American*, September:104–7.

Harth, E. (1995). *The Creative Loop: How the Brain Makes a Mind*. London: Penguin Books.

Hofstadter, D. (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* London: Harper Collins.

Maturana, H., and Varela, F. (1984). *The Tree of Knowledge: The Biological Roots of Human Understanding*. Boston, MA: Shambala.

McCormack, J. (2003). "Evolving Sonic Ecosystems." *Kybernetes: The International Journal of Systems & Cybernetics* 32(1 and 2):184–202.

Minar, N., Burkhart, R., Langton, C., and Askenazi, M. (1996). *The swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*. Working Papers 96-06-042. Santa Fe, NM: Santa Fe Institute.

Minsky, M. (1985). *The Society of Mind*. New York: Simon & Schuster.

Miranda, E. R. (2002). "Emergent Sound Repertoires in Virtual Societies." *Computer Music Journal* 26(2):77–90.

Miranda, E. R. (2004). "At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody." *Evolutionary Computation* 12(2):137–58.

Prigogine, I. (1980). *From Being to Becoming: Time and Complexity in the Physical Sciences*. New York: W. H. Freeman.

Reynolds, G. (1987). "Flocks, Herds and Schools: A Distributed Behavioural Model." *Computer Graphics* 21(4):25–34.

Thomson, J. M. T., and Stuart, H. B. (1986). *Nonlinear Dynamics and Chaos*. Chichester: John Wiley & Sons.

Unemi, T., and Bisig, D. (2005). "Playing Music by Conducting Boid Agents: A Style of Interaction in the Life with A-Life." *Proceedings of A-Life IX*, Boston, MA, pp. 546–50. Cambridge, MA: MIT Press.

Ventrella, J. (1993). "Attractions and Repulsions: A Gravity-Based Animated Microworld for Exploration of Dynamical Systems." http://www.ventrella.com/Ideas/attractions.html (accessed 16 June 2009).

4.7 APPENDIX: THE ALGORITHMS OF THE SYSTEM

```
;; Parameters

energy-parameter          = 100   ;; typical value, set in GUI
activation-thresh-percent = 20    ;; typical value
activation-thresh-value   = 0     ;; initial value

;; Animation procedure

for every agent a in society s
   if status = active ;; update status according to energy
        if energy < 50
                status = asleep then
   else
         energy = energy * 1.25
        if energy > energy-parameter
                status = active then
   then

   if orientation-bit = 0   ;; consider orientation bit
        ;; consider stress, physical XY location and delta-parameter (set in GUI)
        compute-stress for 8 XY-locations around the current XY at a distance delta
        move agent to XY location with lowest stress
   else
        ;; consider proximity
        for every agent b in society s
                    if distance a, b < critical-distance-2 a
                        ;; local interaction
                        n = neighbours of b = all agents within critical-distance-1
                        unless n = nil
                                ;; visually connect all n
                                c = neighbour with highest energy
                                exchange angle of b and c
                        then
                    then
        loop
   then

   if position a = previous-position a  ;; check if agent is stationary
        energy = energy * 0.97  ;; decay when stationary
        else
        energy = energy * 0.83  ;; decay more when moving
   then
loop
```

FIGURE 4.18 Code part 1.

```
;; Musical interpretation of current clusters

ls = compute list of stationary agents
unless ls = nil
       reposition one randomly selected agent in 2d space then

;; draw agents and update history

if player-finished
        cl = compute-clusters
        sc = cluster with max average energy
        a1 = first agent in sc    ;; compute new melody for agents pane
        c1 = MIDI channel of a1   ;; c1 is reference-channel

        ;; create backbone melody: use energy and the personality dataset
        make-reference-melody (see details below)
        for every agent ax in sc except a1
            if status ax = active
               if activation = positive ;; 1 to 100
                  add-grouping-events   ;; apply contraction
               then
               if activation = negative ;; -100 to -1
                  add-ungrouping-events ;; expansion
               then
            then
        play melody
        unless cl = nil  ;; adjust agent activations
            if random 100 < activation-thresh-percent
               invert activation-thresh-status
            then
            for every agent a in cl ;; use threshold algorithm to control agents' activation
                     if activation-status = 0
                        increment activation of a by 10 %
                        increment 1
                        else
                        decrement activation of a by 10 %
                        decrement 1
                     then
            loop
        then
then
```

FIGURE 4.19 Code part 2.

```
;; Creation of a reference melody

selectors = '((0 0 1 1) (0 1 1 0) (1 1 0 0) (1 0 0 1) (1 0 1 0) (0 1 0 1))
selector  = select any bit-pattern from selectors
ra (reference-agent) = first agent in cluster
mc        = MIDI channel is set to ID of ra
nr-events = ra energy / 5
density   = ra energy
start-pitch = 48 + choose any private-interval of reference-agent
neighbour = select any neighbour agent of reference agent
repeat nr-events
        ;; add-events using 4 circular argument lists:
        1: pitch    = if first bit = 0 use private-intervals of ra
                      else use private-intervals of neighbour
        2: velocity = if second bit = 0 use private-velocities of ra
                      else use private-velocities of neighbour
        3: duration = if third bit = 0 use private-durations of ra
                      else use private-durations of neighbour
        4: IOT      = if fourth bit = 0 use private-IOTs of ra
                      else use private-IOTs of neighbour
        all events added on MIDI channel mc.
```

FIGURE 4.20 Code part 3.