An Artificial-Chemistry Approach to Generating Polyphonic Musical Phrases

Kazuto Tominaga and Masafumi Setomoto

School of Computer Science, Tokyo University of Technology
1404-1 Katakura, Hachioji, Tokyo 192-0982 Japan
tomi@acm.org

Abstract. Many techniques in the evolutionary-computing and artificial-life fields have been applied to generating musical phrases. Recently, artificial chemistries, which are virtual models of chemical systems, have come into use for the purpose. In this paper, we attempt to produce polyphonic phrases using an artificial chemistry. The molecules and chemical equations are so designed as to implement a basic set of rules in counterpoint, a technique to compose polyphonic music. We perform a preliminary experiment using a simulator for the artificial chemistry; in the experiment, some good polyphonic phrases are generated, which confirms that the present approach is promising.

Key words: artificial chemistry, music composition, polyphony, counterpoint, trial and error.

1 Introduction

Generating music by computers is always of great interest. Many techniques in various areas have been applied to achieving the goal, including those of evolutionary computation. Specifically, genetic algorithms (GAs) are extensively used and have achieved great successes in various applications [1]. Modelling techniques in the field of artificial life, such as cellular automata and swarm, have also been used to generate musical phrases [2, 3]. Recently, *artificial chemistries* have come into use for the purpose [4]. An artificial chemistry is a model of a (virtual or real) system comprising a large number of molecules that interact with each other according to a given set of rules similar to chemical equations [5]. We have our artificial chemistry [6], and studied on generating homophonic phrases with it [7]. In this paper, we address a more challenging problem — generating *polyphonic* musical phrases — in order to investigate the applicable range of artificial chemistries in music generation.

2 Artificial Chemistry

This section briefly explains our artificial chemistry [6][1]. It has *v-atoms* and *v-molecules*, corresponding to natural atoms and molecules, respectively. V-

[1] This paper uses different terminology and notation from the ones in [6].

M. Giacobini et al. (Eds.): EvoWorkshops 2008, LNCS 4974, pp. 463–472, 2008.

https://link.springer.com/chapter/10.1007/978-3-540-78761-7_49

molecules react with each other according to *recombination rules*, which correspond to chemical equations but are expressed in terms of patterns. The artificial chemistry employs the "well-stirred tank reactor" model: the reaction pool has no spatial structure.

Atoms and Molecules. A v-atom is denoted by a capital letter followed by a sequence of lower-case letters and/or numbers. For example, `A`, `B`, `C2`, `Ix` and `Term` can be v-atoms. A v-molecule is a sequence of v-atoms, or a stack of such sequences. Example v-molecules are shown in Fig. 1, with their string representations. In the notation, lines are delimited by slashes (`/`). Each line is preceded by a number called *displacement* (the number before the hash (`#`)), which is the offset of the line relative to the top line.

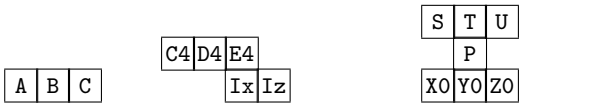


Fig. 1. Example v-molecules.

Patterns and Recombination Rules. A *pattern* matches (or does not match) a v-molecule. A pattern can include *wildcards*. There are two types of wildcards: an *atomic wildcard*, denoted by `<`*n*`>` (*n* is a number), which matches any v-atom; and a *sequence wildcard*, denoted by `<*`*n*`>` or `<`*n*`*>`, which matches a sequence of zero or more v-atoms of any kinds (the star (`*`) indicates the direction that the wildcard can extend). *Recombination rules* are defined in terms of patterns. For example, the following recombination rule

$$\texttt{0\#<*1>AB/1\#CD/} + \texttt{0\#AB<2*>} \rightarrow \texttt{0\#<*1>ABAB<2*>/1\#CD/} \tag{1}$$

recombines v-molecules, `0#ABAB/3#CD/` and `0#ABABAB/`, to produce a v-molecule, `0#ABABABABAB/3#CD/`. In this case, the wildcard `<*1>` matches the sequence of v-atoms `AB`, and `<2*>` matches `ABAB`.

Dynamics. A *system* is defined in the artificial chemistry, which consists of the set of v-atoms, the set of recombination rules, and the multiset of initial v-molecules (or *initial pool*)[2]. The system operates as follows.

1. The working multiset (called *pool*) is initialised to be the initial pool.
2. Apply a recombination rule to a collection of v-molecules in the pool.
3. Go to Step 2.

Recombination rules and v-molecules are selected stochastically on Step 2 when the system is run on a simulator.

[2] The artificial chemistry has other components called *sources* and *drains*, but we omit them since they are not used in this study.

3 Composition of Polyphonic Music

Polyphonic music consists of two or more individual melodies (called *parts*) played simultaneously; it is typified by works in the Baroque period, such as ones by J. S. Bach. Homophonic music, on the other hand, has one main melody, accompanied by chords; modern popular music is mostly homophonic.

Automatic generation of polyphonic phrases is expected to be more difficult than generating homophonic ones. In homophonic music, the melody is principal and the chords support it, so the chords can be added to the melody after the melody is complete. But in polyphonic music, each part should be a good melody, and at the same time, they should sound good when they are played simultaneously.

This study attempt to generate two-part polyphonic phrases by implementing rules in counterpoint in the artificial chemistry. Counterpoint is a technique for composing polyphonic music. Roughly speaking, counterpoint is a set of rules that decide what phrase is good and what is not. For example, a basic rule prohibits a progression of part by any augmented interval. We choose two sets of basic rules for two-part counterpoint for this study, which are shown below.

Rules for the Progression of One Part. Each part of the two parts should obey the following rules. Let *CPRuleset 1* refer to this set of rules ("CP" stands for counterpoint).

1. An augmented interval is not good.
2. A diminished one is good.
3. A disjunct motion should be 6th or less.
4. The 7th note of the scale, the leading tone, should be followed by the tonic.
5. But it can be used transitionally in the sequence of tonic→7th→6th.
6. It can also be used in an arpeggio.

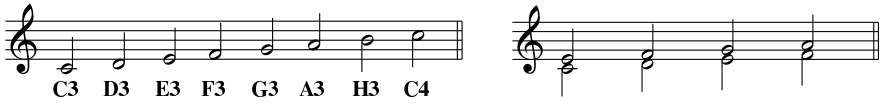
Rules for Two-Parts. The two parts must conform to the following rules about intervals between them. Let *CPRuleset 2* refer to this set of rules.

1. Vertical intervals between the two parts should be mainly 3rd (10th) or 6th.
2. A musical piece must start with one of the three chords I, V and IV; the interval between the parts should be unison, octave, 5th, 3rd (major or minor) or 6th (major or minor).
3. The end of the piece must be an authentic cadence; the lower part should end with the tonic, and the upper should end with unison, octave, 3rd or 10th to the lower.
4. The two parts must not progress in more than three notes in row with keeping the interval of 3rd (10th); same for 6th.

From now, we will use the notation "CPR m-n" to refer to the n-th rule of CPRuleset m.

4 Implementing Rules in the Artificial Chemistry

Let us have a v-atom represent a note, e.g., having a v-atom `A3` represent a note with the 440Hz pitch; a melody can be represented by a sequence of such v-atoms. In this study, we use only the C-major scale: the pitches are represented by `C`n, `D`n, `E`n, `F`n, `G`n, `A`n and `H`n, where n is a number. The pitch `C4` is higher than `C3` by the interval of an octave. `A3` is next to `G3` (not `G2`), and `C4` is next to `H3` (not `H4`) (Fig. 2(a)). A two-line v-molecule can then represent a two-part phrase. For example, the v-molecule `0#E3F3G3A3/0#C3D3E3F3/` represents the phrase shown in Fig. 2(b).



(a) V-molecules for the C major scale (b) `0#E3F3G3A3/0#C3D3E3F3/`

Fig. 2. Representing notes as v-atoms.

The set of recombination rules are so designed that the simulator generates phrases by trial and error. The overview of the designed procedure is as follows.

- Seeds of phrases are given as initial v-molecules (according to CPR 2-2).
- A note or a sequence of notes is added to the shorter part, according to the rules for horizontal intervals between notes (CPRuleset 1).
- If the vertical interval between notes is a prohibited one (as per CPR 2-1 and 2-4), one of the two notes, along with its succeeding notes, is separated from the phrase under production and is returned to the pool.
- If the phrase can connect to one of the given cadences, let it do so to finish the phrase (CPR 2-3). We use four cadences for major keys (Sec. 4.4).

The subsequent sections describe how each process is implemented.

4.1 Seeds of Phrases

CPR 2-1 requires a vertical interval between two notes of the two parts to be mainly 3rd, 6th or 10th. We use the range of pitches from G2 to C5 (approximately two and a half octaves), expecting many combinations of notes may appear in a phrase, and also in order to limit the problem size. CPR 2-2 specifies the beginning of a phrase; we choose C4 for the upper note and C3 for the lower. Let the notation P/Q denote a vertical pair of notes P and Q; the beginning of the phrase shown in Fig. 2(b) is denoted by E3/C3, for example.

The v-molecule that represents the first notes is `0#IpIp/0#Y0C4/0#Y0C3/`. The second line (`Y0C4`) is the upper part, and the third line (`Y0C3`) is the lower. The v-atom `Y0` represents the beginning of a phrase, which prevents the phrase from becoming a trailing part of another phrase. The first line is allotted for v-atoms that express vertical intervals, such as `I1` (unison), `I2` (2nd), etc. (Sec. 4.3)

4.2 Adding Notes to a Part

In order to achieve the requirements stated in CPRuleset 1, we employ a simple method: providing v-molecules that represent the good progressions in the initial pool. For example, the motion from C4 to D4 is good, so v-molecules of the form `0#C4D4/` are given to the pool. V-molecules `0#H3A3/` are not provided because this motion is prohibited (CPR 1-4), but v-molecules of the form `0#C4H3A3/` are given (CPR 1-5) and so are those of `0#H3G3D3/` (CPR 1-6).

These v-molecules are added to a phrase by such recombination rules as the one shown below. This rule attaches a one-line v-molecule (a sequence of notes) that starts with `C4` to the upper part of the phrase that ends with `C4`.

$$\texttt{0\#<*1><2*>/0\#<*3>C4/1\#<*5><6*>/} + \texttt{0\#C4<4*>/} \rightarrow \texttt{0\#<*1><2*>/0\#<*3>C4<4*>/1\#<*5><6*>/} \quad (2)$$

This rule can add, for example, a note sequence `0#C4H3C4E4/` to the upper part of a phrase `0#IpIpI6I6/0#Y0C4D4C4/0#Y0C3F3E3D3/`; this results in a longer phrase `0#IpIpI6I6/0#Y0C4D4C4H3C4E4/0#Y0C3F3E3D3/`. The rule that adds notes to the lower part is described similarly.[3]

4.3 Handling Vertical Intervals

In order to deal with vertical intervals easily, we add an extra line to the two-line representation of a phrase, as we have already seen. The line consists of the following v-atoms.

- V-atoms that represent the interval from unison to 10th: `I1`, `I2`, `I3`, `I4`, `I5`, `I6`, `I7`, `I8`, `I9` and `Ix`.
- V-atom to represent other (bad) intervals: `Iz`.
- The dummy interval to protect the notes: `Ip`.
- V-atom to indicate the interval has not yet been given: `S`.

An example phrase under production is

$$\texttt{0\#IpIpI6IzS/0\#Y0C4E4G4E4C4/0\#Y0C3G3D3C3H2/}\ . \quad (3)$$

`Ip` v-atoms protect the beginning of phrase from the separation explained below. The interval between G3 and E4 is 6th, which is represented by `I6`. The interval between D3 and G4, 11th, is not used in the current study, so the v-atom for "bad" interval, `Iz`, is given. The v-atom `S` indicates that the both parts have notes so the interval must be calculated; another `S` will come next to this v-atom.

Adding `S` to a phrase is performed by the following rule.

$$\texttt{0\#<*1>/0\#<*2><3><4*>/0\#<*5><6><7*>/} \rightarrow \texttt{0\#<*1>S/0\#<*2><3><4*>/0\#<*5><6><7*>/} \quad (4)$$

[3] We relaxed the restriction that each recombination must conserve v-atoms [6], since it is not essential in this study; modifying the rules to conform to it is easy.

Then each `S` is replaced with the interval of notes. The rule below replaces `S` with a v-atom that represents the interval between C3 and A3, 6th (`I6`).

$$\texttt{0\#<*1>S<2*>/0\#<*3>A3<4*>/0\#<*5>C3<6*>/} \rightarrow \texttt{0\#<*1>I6<2*>/0\#<*3>A3<4*>/0\#<*5>C3<6*>/} \quad (5)$$

For each combination of notes, this type of rule is given.

If a vertical pair of notes has a "bad" interval (in line with CPR 2-1), the sequence after (and including) the bad note is separated from the phrase. The following rules carry out this separation for a bad interval, 4th.

$$\texttt{0\#<*1><2>I4<3*>/0\#<*4><5><6><7*>/0\#<*8><9><10><11*>/} \rightarrow \texttt{0\#<*1><2>/0\#<*4><5><6><7*>/0\#<*8><9>/} + \texttt{0\#<9><10><11*>/} \quad (6)$$

$$\texttt{0\#<*1><2>I4<3*>/0\#<*4><5><6><7*>/0\#<*8><9><10><11*>/} \rightarrow \texttt{0\#<*1><2>/0\#<*4><5>/0\#<*8><9><10><11*>/} + \texttt{0\#<5><6><7*>/} \quad (7)$$

Rule (6) cuts the lower part, and (7) cuts the upper. The sequence separated from the phrase is returned to the pool and will be reused to construct another phrase. The interval information after `I4` is thrown away, since only one part has notes in the portion.

These types of rules are given to each of the "bad" interval v-atoms including `Iz`. No such rule is given to the "good" intervals, to leave them intact.

CPR 2-4 is also implemented using the interval information. A long sequence of 3rd (i.e., the two parts move in parallel, keeping the 3rd interval) is cut by the following rules.

$$\texttt{0\#<*1>I3I3I3<2*>/0\#<*3><4..6><7*>/0\#<*8><9..11><12*>/} + \texttt{0\#Dec/} \rightarrow \texttt{0\#<*1>I3I3/0\#<*3><4..6><7*>/0\#<*8><9><10>/} + \texttt{0\#<10><11><12*>/} + \texttt{0\#Dec/} \quad (8)$$

$$\texttt{0\#<*1>I3I3I3<2*>/0\#<*3><4..6><7*>/0\#<*8><9..11><12*>/} + \texttt{0\#Dec/} \rightarrow \texttt{0\#<*1>I3I3/0\#<*3><4><5>/0\#<*8><9..11><12*>/} + \texttt{0\#<5><6><7*>/} + \texttt{0\#Dec/} \quad (9)$$

`<4..6>` is an abbreviation of `<4><5><6>`. When a phrase that has more than three consecutive pairs of notes with the 3rd interval collides with `0#Dec/`, that portion of the phrase is broken and the trailing portion is released. We give similar rules for 6th and 10th.

4.4 Finishing a Phrase with a Cadence

In Sec. 4.2, the rules to add sequence of notes to a phrase are given. In a similar way, rules to join two phrases (each of which has the upper and lower parts) are given. The following rule joins a phrase that ends with A4/F4 and another

phrase that starts with A4/F4.

```
0#<*1><2><3>/0#<*4><5>A4/0#<*6><7>F4/+
0#<8><9><10*>/0#A4<11><12*>/0#F4<13><14*>/ →
0#<*1><2><3><9><10*>/0#<*4><5>A4<11><12*>/0#<*6><7>F4<13><14*>/
```
(10)

This type of rule is given for each possible combination of vertical pair.

Then a cadence such as `0#IpIpIpIpIp/0#A4G4F4E4Z0/0#F4H3D4C4Z0/` can connect to a phrase when the first pair of the cadence is the same as the last pair of the phrase; thus arbitrary cadences can be given in the pool. The dummy v-atom `Z0` prevents another phrase from joining to the finished phrase. We give the following four types of cadence molecules to the pool.

- `0#IpIpIpIpIp/0#A4G4F4E4Z0/0#F4H3D4C4Z0/`
- `0#IpIpIpIpIp/0#A3G3F3E3Z0/0#F3H2D3C3Z0/`
- `0#IpIpIpIpIp/0#F4E4D4C4Z0/0#A3G3H2C3Z0/`
- `0#IpIpIpIpIp/0#D4C4H3C4Z0/0#F3A3G3C3Z0/`

We give an additional rule (shown below) to finish the prolongation of a phrase when it becomes long enough.

```
0#Ip<1..17><18*>/0#Y0<19..35><36*>/0#Y0<37..53><54*>/ →
        0#Term/0#Ip<1..12>/0#Y0<19..30>/0#Y0<37..48>/
                        + 0#<30..35><36*>/ + 0#<48..53><54*>/
```
(11)

5 System Description

The designed system to generate polyphonic phrases is as follows. The initial pool comprises the following v-molecules. The number in braces ({ }) is the number of v-molecules of the specified form.

- Phrase seeds: `0#IpIp/0#Y0C4/0#Y0C3/` {20}
- Authentic cadences: {20 for each}
  - `0#IpIpIpIpIp/0#A4G4F4E4Z0/0#F4H3D4C4Z0/`
  - `0#IpIpIpIpIp/0#A3G3F3E3Z0/0#F3H2D3C3Z0/`
  - `0#IpIpIpIpIp/0#F4E4D4C4Z0/0#A3G3H2C3Z0/`
  - `0#IpIpIpIpIp/0#D4C4H3C4Z0/0#F3A3G3C3Z0/`
- Sequences for conjunct motion (2nd-interval motion): `0#C3D3/`, `0#C3H2/`, etc.; 20 v-molecules for every motion within the range from G2 to C5, except for the motion prohibited by CPR 1-4.
- Sequences for disjunct motion (non-2nd-interval motion): `0#C3E3/`, `0#C3F3/`, etc.; 4 v-molecules for every motion allowed by CPR 1-2 and 1-3 within the range from G2 to C5, except for the motion prohibited by CPR 1-1 and 1-4.
- Motion by CPR 1-5: `0#C5H4A4/`, `0#C4H3A3/` and `0#C3H2A2/` {4 for each}
- Motion by CPR 1-6: `0#H4G4D4/` and `0#H3G3D3/` {4 for each}

– V-molecules to cut a long parallel motion: `0#Dec/` {50}

The following recombination rules are given to the system.

– Rules to add notes: for each note in the range from G2 to C5, a recombination rule similar to Rule (2) is given.
– Rules to join two phrases: for each possible vertical pair of notes, a recombination rules similar to Rule (10) is given.
– Rule (4): the rule to add `S` in the line for interval information.
– Rules to provide interval information: rules to replace `S` with an interval v-atom (`I1`, `I2`, etc.), such as Rule (5).
– Rules to destroy bad vertical intervals: for each undesirable vertical interval, rules similar to Rules (6) and (7) are given.
– Rules to cut a long parallel motion of the two parts: in addition to Rules (8) and (9), similar rules are given for 6th and 10th.
– Rule (11): the rule to terminate a long generation.

6 Preliminary Experiment

The description given in the previous section is executed on a simulator of the artificial chemistry. The total number of v-molecules in the initial pool is 1178, and that of recombination rules is 708. The numbers may seem big, but most part of the description is mechanically generated by a script program. V-molecule species for the same purpose, such as conjunct motion, have the same form; they are different only in their pitches. Same applies to the recombination rules.

The simulator was implemented with the Cocoa Framework and Objective-C, and was run in Mac OS X 10.4 on Intel Core 2 Duo 2 GHz with 2 GB memory. During the run, we noticed that v-molecules of the types `0#H2C3/`, `0#H3C4/` and `0#H4C5/` were lacking (because the conjunct motion C→H and the disjunct motion from the leading tone are prohibited), so we supplied 40 v-molecules of each type while the simulator was running. Except this, there was no human intervention. The run took about nine hours, and produced ten phrases. Fig. 3 shows obtained good phrases (chosen by human ears). The phrase (a) has a cadence at the end, while (b) and (c) do not (i.e., prolongation terminated). Neither of them violates the CPRulesets. Some of the phrases not shown here violate CPR 1-4; they seem to be fragments from v-molecules given for CPR 1-5 and 1-6, produced by cutting. The description should be improved to fix this flaw.

7 Discussion

In this approach, notes are encoded as v-atoms, and then phrases are encoded as v-molecules. These mappings are straightforward. The computing process directly deals with the v-molecules. This approach does not need a complex mapping scheme, such as interpreting a geometric position as sound, which is used in some music-generating systems [3, 4].

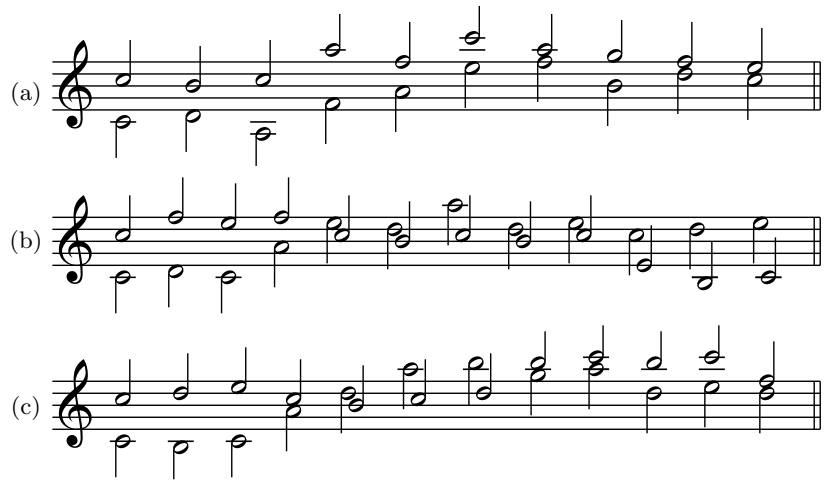


Fig. 3. Generated two-part polyphonic phrases.

In the traditional programming, a procedure is usually so designed as to take correct steps of computation, avoiding errors. In contrast, this approach allows temporary errors, or utilises it to search for a better solution. Some rules try to make as good phrases as possible; some others modify (or destroy partly) a phrase if it is not good. Trial and error are useful when a deterministic algorithm is not available, such as the case of composing music.

Genetic algorithms usually use randomness for selection, crossover and mutation, and have been used in studies for generating musical phrases [1]. A crucial part of GA is fitness evaluation: the fitness function should be designed carefully to have the system work in the desired way. Once the fitness of an individual is evaluated, it is selected as a parent for the next generation (or simply discarded), and crossover and/or mutation are performed. But designing crossover and mutation for a specific application is not an easy task. In contrast, our approach enables programming in modifying a v-molecule to make a better phrase. For example, if a phrase has a bad vertical interval, the system can remove only the problematic portion, or even can replace one of the notes with a good one in a programmed way. We think this facilitates more flexible control in computation.

The simulator employs stochastic collision as the reactor algorithm, and the number of v-molecules affects the probability of reaction in which the v-molecule species is involved. Therefore, increasing or decreasing the number of v-molecules controls the behaviour of the system. For example, if a phrase consisting mainly of conjunct motion is desired, giving many v-molecules that represent conjunct motion will increase the probability of generating such phrases.

In our approach, good components can be given in the initial pool, such as arpeggios and cadences. Furthermore, as we did in our experiment, it is easy to add v-molecules to the pool during the execution. This is not a typical way to control a computation, but it is similar to adding insufficient reagent during

a chemical experiment. Moreover, removing v-molecules and adding/removing recombination rules are easy, since it is an "artificial" chemistry.

Nevertheless, the current approach has some difficulties. One is that it is hard to numerically evaluate how good a phrase is: although phrases that comply with the implemented Rulesets are generated, it is difficult to tell which of them is better than others. One way to achieve such evaluation is to combine this approach with GAs: the initial population generated by the artificial chemistry are fed to GA to evolve. Another difficulty is the low performance of the current implementation. Since we put the understandability of description before the performance, the description of the present system has not been refined for efficiency. Refining recombination rules, as well as adjusting the numbers of v-molecules, would improve the performance.

8 Concluding Remarks

This study implemented a basic set of rules in counterpoint in the artificial chemistry, and generated polyphonic phrases using the simulator. Each rule in counterpoint is realised as steps of recombination, which is similar to programming in procedural language. At the same time, the artificial chemistry deals with multiple v-molecules (i.e., phrases) and decomposes or modifies bad phrases; this realises a kind of generate-and-test method like GAs. Since a process of composing music should include trial and error, we think this dual nature of artificial chemistry will be useful in implementing music-generating systems.

Some generated phrases (including those generated in our previous works) can be listened to in our web site: `http://www.tomilab.net/alife/music/`.

References

1. Burton, A., Vladimirova, T.: Generation of musical sequences with genetic techniques. Computer Music Journal **23**(4) (1999) 59–73
2. Blackwell, T., Bentley, P.: Improvised music with swarms. In: Proceedings of the 2002 Congress on Evolutionary Computation. Volume 2., Piscataway, NJ, IEEE Press (2002) 1462–1467
3. Miranda, E.: On the music of emergent behavior: What can evolutionary computation bring to the musician? Leonardo **36**(1) (2003) 55–59
4. Beyls, P.: A molecular collision model of musical interaction. In Soddu, C., ed.: Proceedings of the 8th International Conference on Generative Art (GA2005), Milan, AleaDesign (2005) 375–386
5. Dittrich, P., Ziegler, J., Banzhaf, W.: Artificial chemistries — a review. Artificial Life **7**(3) (2001) 225–275
6. Tominaga, K., Watanabe, T., Kobayashi, K., Nakamura, M., Kishi, K., Kazuno, M.: Modeling molecular computing systems by an artificial chemistry — its expressive power and application. Artificial Life **13**(3) (2007) 223–247
7. Miura, T., Tominaga, K.: An approach to algorithmic music composition with an artificial chemistry. In: Explorations in the Complexity of Possible Life (Proceedings of the 7th German Workshop on Artificial Life (GWAL-7)), Berlin, Germany, Akademische Verlagsgesellschaft Aka GmbH (2006) 21–30