오디오 프로그래밍 언어 ChucK을 이용한 실시간 알고리듬 작곡 도구의 개발 연구

최 홍 찬, 김 준
동국대학교 영상대학원 멀티미디어학과 컴퓨터음악전공

A Study and Implementation of Software for Real-time Algorithmic Composition with ChucK

Hongchan Choi, Jun Kim
Computer Music Lab., Department of Multimedia,
Graduate School of Image and Contents,
Dongguk University

요 약

본 논문은 오디오 프로그래밍 언어 ChucK을 사용하여 실시간 환경에서 알고리듬 작곡 및 연주를 수행할 수 있는 도구, 특히 클래스(class)의 설계 및 개발 연구를 목표로 한다. 또한, 확률적 유한 상태 오토마타(probabilistic finite-state automata)를 사용하여 무작위적 및 결정적인 방법을 동시에 사용하여 소닉 이벤트(sonic event)의 순차적 집합을 생성하는 기술을 사용하며, 이를 통해 기존의 알고리듬 작곡 기법의 사례들과는 다르게 접근한다.

I. 컴퓨터와 음악의 필연적 접점; 알고리듬 작곡

컴퓨터의 등장 이전부터 많은 작곡가들이 작곡의 과정을 형식화하기 위해 노력해왔다. 작곡 과정의 형식화는 알고리듬(algorithm)[1]에 기반을 둔 작곡과 동일한 문맥을 갖고 있다고 말할 수 있다. 어떠한 일을 수행하고 결과를 얻기 위한 단계와 순서를 명문화 또는 공식화 할 수 있다면, 컴퓨터는 정리된 논리에 따라 인간을 대신하여 작업을 수행할 수 있게 된다. 요컨대 형식화, 알고리듬, 컴퓨터는 동일 선상에 있으며, 알고리듬을 수립할 수 있다는 것은 컴퓨터의 도입이 가능함을 의미한다.

컴퓨터 음악의 역사 전반에 걸쳐 컴퓨터를 이용한 작곡은 수많은 음악가와 과학자들에 의해 개발, 연구되었다. 이와 같이 다양한 음악적 결정을 컴퓨터가 직접 수행하는 것을 기반으로 하는 형태의 작곡을 알고리듬 작곡(algorithmic composition)이라 한다. 컴퓨터의 자의적 판단과 결정을 위해 다양한 인공지능 컴퓨팅 기술들이 도입되었고, 새로운 기술들이 개발되고 있다. 그리고 기술의 발전에 의해, 현재는 컴퓨터가 알고리듬에 의한 연주 및 퍼포먼스를 실시간으로 수행하는 수준에 이르렀다.

컴퓨터 음악에 있어 알고리듬 작곡은 충분한 정통성을 지닌 학문적 흐름으로 자리매김하였다. 음악을 위한 도구로서의 컴퓨터가 아닌, 음악의 또 다른 생산자로서의 컴퓨터를 완성시키기 위한 노력이 계속되고 있으며, 다른 예술 분야에서도 유사한 맥락의 연구가 활발하게 진행중이다.

인간이 아닌 컴퓨터에 의해 생산된 결과가 예술적으로 어떠한 가치를 지니는가에 대한 논쟁은 피할 수 없는 의미심장한 것이지만, 본 논문에서는 그러한 내용은 논의하지 않기로 한다.

II. 기술적 구성요소

2.1 확률적 유한 상태 오토마타(Probabilistic Finite State Automata)를 이용한 CAAC

컴퓨터를 이용한 알고리듬 작곡(CAAC;

1) 유한한 동작의 순차적인 집합. 계산과 데이터 처리를 위하 사용된다.

Computer-Aided Algorithmic Composition)은 짧지 않은 역사를 가지고 있으며, 이를 위해 다양하고 독창적인 방법론들이 제시된 바 있다. 그 중 본 논문에서 제안하고자 하는 것은 확률적 유한 상태 오토마타(probabilistic finite-state automata)의 사용이다.

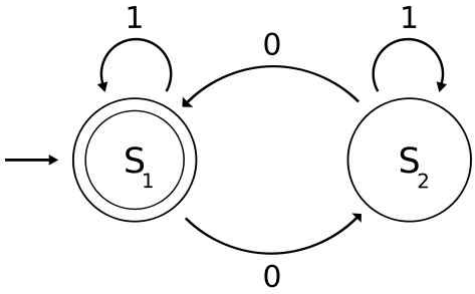


그림 1. 유한 상태 오토마타의 다이어그램

그림 1은 일반적인 유한 상태 오토마타(finite-state automata; FSA)의 도식을 보여준다. 유한 상태 오토마타는 가장 기본적인 데이터 처리의 기제라 할 수 있다. 일반적인 FSA는 결정적(deteministic) 기제이다. 다시 말해, 입력되는 데이터가 같다면 그 결과도 반드시 같다.

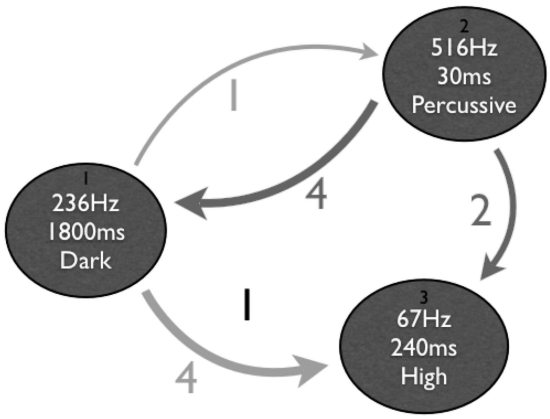
반면에 확률적 유한 상태 오토마타는 결정적 특성을 기반으로 하는 확률적 특성을 지니고 있기 때문에, 작곡가의 작곡 과정에서 나타나는 무작위성(stochastic)을 적절히 반영할 수 있다는 장점을 지닌다. 그리고 이를 통해 작곡가에 의해 창작된 동기(motive)를 컴퓨터가 확률에 기반을 두어 변주(variation)할 수 있다는 장점도 동시에 가진다.

인공지능을 표방한 학문 분야들의 주요 목표는 '사람처럼 생각하고 행동하는 컴퓨터'라 할 수 있지만, 본 논문에서는 약간 다른 관점에 서서 바라보고자 한다. 알고리듬 작곡을 위한 소프트웨어는 작곡가로 하여금 짧은 시간 내에 더 많은 예술적 시도를 할 수 있게 해주는 - 자신이 원하는 멜로디를 그대로 연주할 수도 있으며, 자신이 제작한 멜로디를 컴퓨터가 다양하게 변주하도록 할 수도 있으며, 컴퓨터가 직접 멜로디를 만들 수도 있는 - 도구라 할 수 있다.

2.2 소닉 이벤트(Sonic Event)의 도입

알고리듬 작곡을 위해 마르코프 연쇄[2]나 FSA를 사용하는 사례의 대부분은 멜로디의 생성, 특히 연주할 음의 높이를 결정하기 위함이었다. 수많은 소리의 속성 중에서 음의 높이 하나 만을 작곡의 대상으로 고려하고 있는 것이다. 이러한 단점을 극복하기 위하여 본 논문에서 제시하고자 하는 새로운 개념은 바로 '소닉 이벤트(sonic event)'라는 것이다.

이는 FSA에 의해 단순히 음의 높이만을 결정하는 것이 아니라, 음색, 길이, 시간에 따른 음량 변화의 형태, 음향



State	Properties	Destination - Probabilities	
1	236Hz, 1800ms, Dark	2	20%
		3	80%
2	516Hz, 30ms, Percussive	1	67%
		3	33%
3	67Hz, 240ms, Bright	1	100%

그림 2. 확률적 유한 상태 오토마타와 소닉 이벤트의 전이를 나타낸 표

효과의 종류와 양 등의 총체적인 음향 속성을 FSA의 상태 속성으로 정의하여 사용하는 것이다. 또한, 2.1절에서 언급한 바와 같이 이러한 상태들의 전이는 무작위적(stochastic)이면서 동시에 결정적(deteministic)인 방법으로 이루어지게 된다.

따라서, 무작위에 의한 단순한 음의 나열과는 달리, 음향의 총체적인 양상을 상태 정보로 저장하고 FSA에 의해 진행되는 상태 전이(state transition)에 의하여 끊임없이 변화하는 음악으로 나타난다. 작곡가는 자신의 동기를 결정적인 형태로 입력해두고, 컴퓨터가 이를 얼마나 변주할지를 확률적으로 조절 할 수 있다.

2.3 오디오 프로그래밍 언어 ChucK의 사용

ChucK은 2002년에 출시된 오디오 프로그래밍 언어로 최근의 강력한 컴퓨팅 환경을 바탕으로 하여 기존의 오디오 프로그래밍 환경보다 우월한 프로그래밍 환경을 제공한다. 또한, 프론트-엔드(front-end)[3] 편집기인 Mini

2) 러시아 수학자인 안드레이 마르코프에 의해 완성. 시간에 따른 시스템 상태의 변화를 나타낸다. 매 시간마다 시스템은 상태를 바꾸거나 같은 상태를 유지한다.

3) 소프트웨어 시스템 중 사용자와 직접 상호작용하는 부분.

Audicle을 사용하면 라이브 코딩(live-coding)[4]을 하는 것도 가능하다.

본 논문을 통해 개발되는 도구는 ChucK의 문법에 맞게 제작된 클래스(class)[5]이며, 이 클래스를 통해 확률적 유한 상태 오토마타를 구현하였다. 또한, 오토마타로부터 생성된 상태 전이에 대한 데이터를 음향으로 발현할 수 있는 신디사이저부도 구현하였다. ChucK을 기반으로 하는 알고리듬 작곡을 위한 클래스 및 신디사이저를 통틀어 이하 AC3(Algorithmic Composition Class for ChucK)라 지칭하고자 한다.

본 논문에서 AC3의 구현을 위해 ChucK/Mini Audicle 플랫폼을 선택한 이유는 다음과 같다.

- **실시간 처리에 기초한 상호작용성**
  오디오 프로그래밍 언어 ChucK은 '강력한 시간 관리(strongly-timed)'에 대한 장점을 지니고 있으며, 이로 인해 다른 오디오 프로그래밍 언어에서는 불가능한 시간에 관련된 고려 및 처리가 가능하다. 이로 인해 순간과 시점이 중요한 실시간 퍼포먼스에서 프로그래밍 언어로써 새로운 가능성을 보여주었다. 또한, 실시간 퍼포먼스가 가능하다는 것은 다양한 미디어 간의 상호작용성(interactivity)을 가질 수 있다는 것과 같은 의미로 생각할 수 있다.

- **STK[6] 인스트루먼트와 이펙트 라이브러리의 제공**
  C언어 기반으로 제작된 STK 인스트루먼트 및 이펙트를 ChucK의 유닛 제너레이터(Unit-Generator)[7]의 형태로 자유롭게 사용할 수 있다. 물론 사용자가 신디사이저를 직접 제작하여 합성음을 만들 수도 있으며, 이를 기존의 STK 인스트루먼트와 조합하거나 이펙트를 이용하여 고품질의 음향 효과를 손쉽게 얻을 수 있다.

- **동시성이 강조된 프로그래밍 환경**
  ChucK 프로그램은 ChucK의 가상 머신(virtual machine)[8]에 여러 개의 슈레드(shred)[9]로 실행될 수 있다. 하나의 프로그램을 완성 시켰다면 완성된 내용

4) 퍼포먼스의 일부분으로서 소프트웨어를 실시간으로 작성하는 행위.
5) 프로그래밍 언어 구조. 객체를 생성할 수 있는 청사진이며, 객체의 속성과 함수 등으로 정의된다.
6) Synthesis Tool Kit. C++로 작성된 신호처리 및 소리합성을 위한 오픈 소스 클래스.
7) 신디사이저의 설계 및 신호처리 알고리듬 소프트웨어의 작성을 위한 구성 요소. 가장 기본적인 형식 단위로서 대부분 오디오 프로그래밍 언어에서 사용.
8) 실제의 기계와 유사하게 프로그램을 실행할 수 있도록 구현된 소프트웨어.
9) 가상 머신에서 실행되고 있는 ChucK 프로그램의 프로세스. 일반적인 프로그래밍 언어에서는 스레드(thread)라는 이름으로 사용됨.

그대로, 혹은 몇몇 파라미터의 값을 변경하여 다수의 슈레드로 실행시킬 수 있다. 따라서 코드 및 프로그램의 재사용성이 매우 뛰어난 장점을 지닌다. 슈레드를 가상 머신에 등록하는 과정 역시 실시간으로 이루어지며, 등록을 위한 조작 역시 매우 간단하기 때문에 실시간 퍼포먼스에서도 충분히 사용할 수 있다. 이것이 ChucK을 라이브-코딩을 위한 언어로 주목을 받을 수 있게 한 점이다.

III. AC3: ChucK 클래스의 설계

AC3를 사용한 알고리듬 작곡은 크게 2개의 부분으로 나뉘어진다. 첫째로, 플레이어 모듈(player moduel)은 소닉 이벤트를 위한 FSA 정보를 담고 있으며, FSA에 의해 생성되는 시퀀스를 연주하기 위한 STK 인스트루먼트와 이펙트로 구성되어 있다. 다른 하나는 디렉터 모듈(director module)이며, 이 모듈은 다수의 플레이어 모듈을 호출하여 연주 및 정지시키는 일을 하며, 곡의 전체적인 음량과 템포를 설정할 수 있다.

3.1 플레이어 모듈

작곡가, 프로그래머는 플레이어 모듈에 명령어의 집합을 입력함으로써 FSA를 구축할 수 있다. 플레이어 모듈은 디렉터 모듈에 의해 호출되는 순간부터 자신의 내부에 구축된 FSA를 움직여서 소닉 이벤트를 위한 정보를 출력해낸다. 또한, 플레이어 모듈에 포함된 신디사이저는 출력된 정보를 바탕으로 소리를 출력한다.

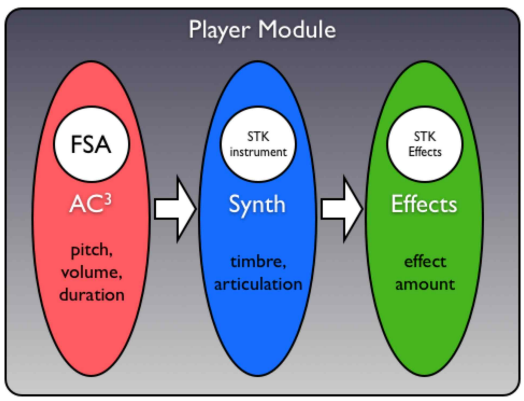


그림 3. 플레이어 모듈의 다이어그램

기본적으로 FSA를 구축하고 출력해내는 것은 모든 플레이어 모듈이 동일하나, 신디사이저 부분은 사용자가 원하는 대로 기존의 구조를 수정하거나 새롭게 작성하는 등의 작업이 가능하다.

3.2 디렉터 모듈

전술한 바와 같이 플레이어 모듈은 일련의 소닉 이벤트를 연주하는 역할을 한다. 본 클래스를 이용한 음악 작품은 다양한 소닉 이벤트가 순서대로 연주되거나, 혹은 동시에 연주되는 등의 과정을 통해 제작된다. 이렇듯 디렉터 모듈은 다양한 플레이어 모듈을 순차적 혹은 동시에 호출하기 위해 사용된다.

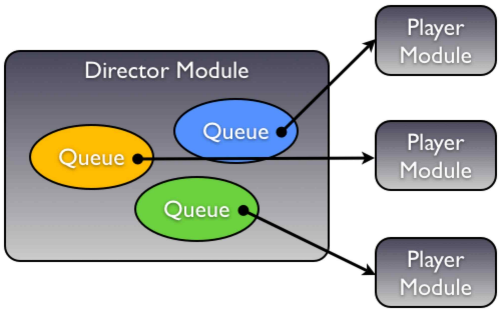


그림 4. 디렉터 모듈의 다이어그램

Ⅳ. 작품의 연주

본 시스템을 이용한 음악 작품의 연주 형태는 크게 2가지로 나뉜다. 하나는 미리 정의된 조합과 순서에 의한 연주이며, 다른 하나는 작품 실연 도중에 실시간, 즉흥적으로 플레이어 모듈을 호출하여 연주하는 것이다.

4.1 디렉터 모듈에 의한 연주

전통적인 의미의 작곡은 곡의 시작과 끝이 있으며, 전체적인 흐름을 갖는 것이 일반적이다. 본 시스템을 통해 작품의 총체적인 서사성을 얻기 위해서는 디렉터 모듈이 플레이어 모듈들 어떤 순서로 혹은 어떠한 조합으로 연주시킬 것인가를 결정하는 것으로 가능하다. 이는 다수의 전자음악작품에서 사용되고 있는 큐 리스트(q-list)와 같은 방법이다.

이처럼 디렉터 모듈만을 사용하여 작품을 제작하는 경우, 선형적인 테잎음악(tape music)형태의 작품이 된다. 하지만 디렉터 모듈 자체도 가상 머신에 의해 실행되는 하나의 프로그램이기 때문에 순차적으로 혹은, 동시에 여러 슈레드의 호출이 가능하며, 이로 인해 디렉터 모듈 만으로도 실시간으로 인터랙티브하게 작품을 제작하는 것이 가능해진다.

4.2 라이브 코딩(live-coding)에 의한 연주

ChucK/Mini Audicle 플랫폼이 제공하는 실시간 프로그래밍 환경을 사용하면 슈레드를 원하는 순간에 추가하거나 제거할 수 있기 때문에, 프로그래밍에 의해 출력되는 음향을 들으며 즉시 수정하여 다시 출력하는 등의 작업이 가능하다.

또한, 그림 6과 같이 하나의 ChucK 프로그램의 파라미터를 변화 시키면서 동시에 여러 개의 슈레드를 호출할 수 있다. 다시 말해, 하나의 FSA를 가지고 있는 플레이어 모듈을 신디사이저의 파라미터나 연주 속도, 전체적인 음량 등을 변경한 다른 슈레드들을 순차적으로 혹은, 동

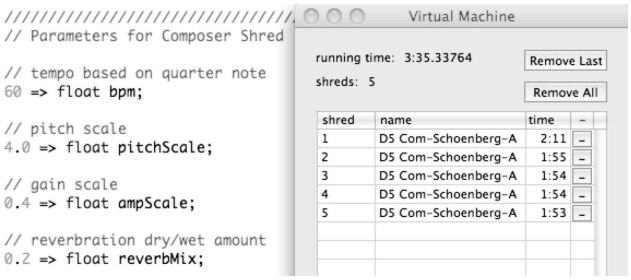


그림 6. 하나의 ChucK 프로그램에서 생성된 다수의 슈레드

시에 호출할 수 있다는 것이다.

Ⅴ. 문제점 및 개선방안

5.1 독창성 및 다양성의 결여

FSA를 이용한 알고리듬 작곡은 기존에 사례들에서 충분히 찾아볼 수 있는 방법이다. 하지만 본 논문에 제시된 소닉 이벤트라는 개념의 도입을 통해 기존 사례들과의 차별화를 시도하였다. 또 다른 문제는 FSA 부분과 신디사이저 부분이 플레이어 모듈에 하나로 통합되어 있다는 것이다. 이 부분을 개선하여 시퀀스를 생성하는 부분과 소리 합성 및 음향 처리를 수행하는 부분을 독립적으로 분리하여 다양성을 확보할 계획이다.

5.2 다양한 기술의 도입

역사적으로 시도되었던 다양한 알고리듬 작곡 기술을 추가로 도입한다. 이것은 AC3라는 이름의 클래스가 본격적인 CAAC를 위한 통합 패키지로 완성되는데 큰 반석이 될 것이다. 그 예로, 인공지능 분야에서 언급되는 Fractal, Neural Network 등이 그것이며, 기존에 제작된 알고리듬을 ChucK으로 이식하거나, 존재하지 않는 것들은 새로이 알고리듬을 작성하여 추가함으로서 가능하리라 생각한다.

5.3 세분화, 전문화 된 모듈의 구축으로 실시간 알고리듬 퍼포먼스를 위한 통합 패키지 개발

2개로 나뉘어져 있는 비교적 단순한 시스템을 보다 세분화, 전문화 된 모듈의 구축을 통해 개선한다. 디렉터 모듈은 악곡의 전체적인 형식 등을 제어하는 고수준의 사고를 할 수 있도록 제작하며, 컴포저 모듈(composer moduel)는 FSA, 신경망(neural network)[10]등을 통한 시

```
// AC3 Demo 5: "The Day in the Life"
// Simple Director Shred

Machine.add("D5 Com-Beethoven I") => int voiceBowed;
25::second => now;
Machine.add("D5 Com-Schoenberg-A I") => int voiceFlute;
25::second => now;
Machine.remove(voiceFlute);
Machine.add("D5 Com-Schoenberg-B I") => int voiceCla;
25::second => now;
Machine.remove(voiceBowed);
25::second => now;
Machine.remove(voiceCla);
```

그림 5. 디렉터 모듈의 예

퀀스를 생성하도록 제작한다.

또한, 퍼포머 모듈(performer module)은 시퀀스를 해석하여 소닉 이벤트(소리 합성 방식, 음고, 음색, 변조 등)로 생성하는 작업을 수행한다. 그 밖에도 음향 효과를 담당하는 블랙박스 모듈(blackbox module), 다른 어플리케이션, 다른 컴퓨터와의 OSC(open-sound control) 통신을 위한 네트워커 모듈(networker module)등을 추가할 계획

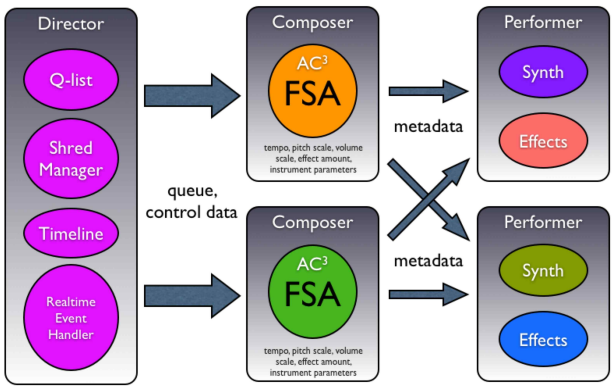


그림 6. 디렉터, 컴포저, 퍼포머 모듈로 구성된 알고리듬 작곡 통합 패키지의 다이어그램

이다.

VI. 참고문헌

[1] Curtis Roads, *"Computer Music Tutorial"*, pp.819-909, 1996

[2] Robert Rowe, *"Machine Musicianship"*, pp.6-14, 2001

[3] Ge Wang, Perry R. Cook, *"ChucK Manual"*, 2007

[4] Ge Wang, *"The ChucK Audio Programming Language <A Strongly-timed and On-the-fly Environ/mentality>"*, Ph.D Dissertation, 2007

[5] Tomasz Oliwa and Markus Wagner, *"Composing Music with Neural Networks and Probabilistic Finite-State Machines"*, 2008

[6] George Papadopoulos, Geraint Wiggins, *"AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects"*

[7] Nigel Gwee, *"Complexity and Heuristics in Rule-based Algorithmic Composition"*, 2002

[8] Jon McCormack, *"Grammar Based Music Composition"*

[9] Christopher Ariza, *http://www.flexatone.net/*

[10] Official Chuck User Forum, *http://electro-music.com/forum/forum-140.html*

10) 생물학적 뉴런으로 구성된 회로 및 네트워크를 지칭하나, 현재는 인공 뉴런이나 노드로 구성된 인공 신경망을 가리킴.