A PROCESS ALGEBRA FOR STOCHASTIC MUSIC COMPOSITION

Brian J. Ross
Brock University
Department of Computer Science
St. Catharines, Ontario, Canada L2S 3A1
ph. (416) 688-5550 ext. 4284
bross@sandbanks.cosc.brocku.ca

Abstract

The Musical Weighted Synchronous Calculus of Communicating Systems (MWSCCS) process algebra is presented. The algebra permits the design of stochastic automata which perform a variety of musically interesting behaviours, including both vertical and horizontal reactive behaviour between processes. A notable feature is the ability to formally model complex stochastic musical systems with a simple notation having a rigorous formal semantics. This permits the formal analysis of musical systems, and is a basis for deriving well-founded implementations. MWSCCS complements other music formalisms that use trace theory, Petri Nets, and Markov processes.

1 Introduction

This paper introduces a process algebra model of music called the Musical Weighted Synchronous Calculus of Communicating Systems, or MWSCCS. Process algebra are popular contemporary formalisms for modeling and analyzing concurrent systems (Hoare, 1985; Milner, 1989). They are widely accepted as effective models of concurrency largely because of the high degree of abstraction possible with them, along with their intuitive "programming language" feel. MWSCCS considers music as comprised of streams of discrete, concurrent behaviours that can be observed and reacted upon by musical processes. These streams denote the horizontal structure of music – the relative order of events with respect to each other. Work in process algebra has shown that this level of abstraction is convenient for analyzing complex concurrent behaviour, since it denotes the essence of a systems activity without being encumbered with structural details of the system itself. The vertical component of music – event simultaneity – is naturally denoted by the event domain of the synchronous algebra. A major feature of MWSCCS is its set of probabilistic primitives for representing events that adhere to stochastic event distributions. These stochastic primitives open the possibility of novel techniques for automated stochastic composition.

2 The MWSCCS Process Algebra

MWSCCS is an enhancement of Tofts' WSCCS (Tofts, 1990), which in turn is a probabilistic version of Milner's SCCS (Milner, 1989). MWSCCS adds to WSCCS a musical action domain and useful musical operators. A comprehensive discussion of MWSCCS algebra is in (Ross, 1995a).

The most basic identifiable atomic event or communication signal from a process is a *particle*. In the domain of music, a useful set of particles to include in $\mathcal{A}$ are the set of tones on a piano keyboard.

We denote this set $\mathit{Notes} \subseteq \mathcal{A}$ as $\mathit{Notes} = \{a_1,\ a_1^\sharp,\ a_1^\flat,\ b_1,\ b_1^\sharp,\ b_1^\flat,\ \cdots\}$, where each note is indexed with its octave. Each particle a has a corresponding *complement* $\overline{a}$. Informally, an overbarred particle represents an *output* communication, while the corresponding non-overbarred equivalent is an *input* communication. We let $\overline{\overline{a}} = a$. Particles are used to construct *actions*. An action is an event occurring at one moment in time. We let $\{a, b, ...\}$ range over $\mathcal{A}$, and $\{\alpha, \beta, ...\}$ range over Act. The action $\mathbf{1}$ is a moment of silence. Whenever a particle and its complement reside in α, they absorb with one another and disappear.

A major feature of MWSCCS is the ability to assign relative frequencies and priorities $\mathcal{W}$ to processes. For example, in $2\omega a\overline{b} + 3\omega c + 1\omega\overline{d}$, the term $2\omega a\overline{b}$ has a relative probability of 2/6 with respect to the other terms, and it has a priority of $\omega^1 = 1$.

The language $\mathcal{E}$ of agent expressions is defined by the grammar

$$\mathcal{E} \ ::= \ \mathbf{0} \ | \ X \ | \ \alpha.E \ | \ E\lceil A \ | \ E\backslash A \ | \ E[f] \ | \ \sum_i w_i E_i \ | \ E \times F \ | \ \mu_i \tilde{X}\tilde{E}$$

where $E, F \in \mathcal{E}$, X is a process variable, $a \in \mathcal{A}$, $w_i \in \mathcal{W}$, $\alpha \in \mathcal{A} \cup \mathcal{W}$, $A \subseteq \mathcal{A}$, and f is a particle renaming function. Tofts defines the semantics of WSCCS using natural deduction-style rules of inference. Using these rules, transitions such as $E \xrightarrow{\alpha} E'$ are derivable, which denotes the transformation of E to E', yielding the action α. This α represents a simultaneous composite event, for example, a chord. We therefore expect to hear multiple actions α_i emitted sequentially in a composition: $P \xrightarrow{\alpha_1}\xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} \cdots$. This $\rightarrow$ operative represents a *synchronous global clock*.

The operators have the following informal meanings. The *null* process $\mathbf{0}$ is the process that does nothing. The *prefix* operator in $\alpha.E$ represents the process that can perform the action α, thereafter becoming the process E. The *permission* operator in $E\lceil A$ denotes the process that performs the particles that are members of the set A, in effect pruning all actions not in A. The *restriction* operator in $E\backslash A$ is the converse of permission. The *relabeling* operator in $E[f]$ renames particles according to f. The *choice* operator $\sum_i w_i E_i$ represents the choice of execution of a set of processes E_i. The E_i chosen is influenced by the frequency and priority of any weight expressions. The *parallel composition* operator in $E \times F$ forces concurrent processes E and F to synchronously communicate with one another. If both E and F are unweighted, then a new action is formed by their combined actions:

$$a.E \times b.F \ = \ ab.(E \times F)$$

If weight terms are involved, then a number of rules are used to combine probabilities and priorities. Process definition and recursive invocation is also supported.

More complex operators can be derived with the above primitives. Whereas synchronous processes yield their actions in lock step with the clock, asynchronous processes may nondeterministically wait before eliciting their actions. We define asynchronous processes with the following meta-operator:

$$\bullet(P) \ \overset{\text{def}}{=} \ P \ + \ \mathbf{1}.\bullet(P)$$

The strict sequential execution of musical events is often required. A sequential composition operator ";" is useful for this purpose:

$$X \ ; \ Y \ \overset{\text{def}}{=} \ (X[c/\surd] \ \times \ \bullet(c.Y)) \backslash\{c\}$$

where c is a new particle not generated by X or Y.

The above considers musical events to be discrete actions running synchronously with an unspecified clock. This view of music is adequate for discussing important semantic aspects of the algebra. In reality, music is much more complex: notes have different durations, dynamics, and timbres. Enhanced musical activity can be modeled by suitably enriching the action domain. A basic enhancement that permits note duration is to use explicit *Note on* and *Note off* signals for each note, which has the advantage of mapping to MIDI. For example, let a represent an on-note, and a' an off-note. With this notation, if we presume that the global clock is running at a quarter-note duration, then the transition

$$P \xrightarrow{\overline{c}} \xrightarrow{\mathbf{1}} \xrightarrow{\mathbf{1}} \xrightarrow{\mathbf{1}} \xrightarrow{\overline{ec'}} \xrightarrow{\overline{e'g}} \xrightarrow{\mathbf{1}} \xrightarrow{\overline{g'}} P'$$

represents a sequence consisting of a whole note c, quarter note e, and half-note g. The wait action $\mathbf{1}$ is used to designate timing intervals. This process is written in MWSCCS as:

$$P \overset{\text{def}}{=} \overline{c}.\mathbf{1}.\mathbf{1}.\mathbf{1}.\overline{ec'}.\overline{e'g}.\mathbf{1}.\overline{g'}.P'$$

3 Example

Consider the expression

$$1\overline{ca} + 2\overline{da}$$

which represents a simple stochastic fragment that plays the chord $\{c, a\}$ with a probability of 1/3, and $\{d, a\}$ with a probability 2/3. We can make a nonterminating melody M by defining:

$$M \overset{\text{def}}{=} 1\overline{ca}.M + 2\overline{da}.M$$

and then invoking M. Next, we define an expression that harmonizes with M:

$$\mathit{Composition} \overset{\text{def}}{=} (\ M[x/c, y/d]\ \times\ (x\overline{cf}.H + y\overline{dg}.H)\ )\ \backslash\{x, y\}$$

The intention is that H should play f whenever it hears c, and play g whenever d is heard. In order to hear c and d, the expression relabels them so H can hear them, and then play the original notes as played by M. By convention, non-overbarred actions are input or "hearing" actions, and therefore the hearing actions x and y from H are restricted in the final output. An enhancement Composition is:

$$\mathit{Composition} \times (\mathit{Composition}[b/c, f^\sharp/a])$$

Here, we play *Composition* alongside a variation of itself that transposes c with b, and a with $f^\sharp$.

We now rewrite M so that it can be terminated if desired:

$$M' \overset{\text{def}}{=} 1\omega\overline{ca}.M' + 2\omega\overline{da}.M' + \omega^2 \mathit{fini}.\overline{da}.\overline{ceg}$$

fini is a control action we introduce to force M to quit. The priority of this term is set to 2, which makes it a higher priority than the other priority 1 terms. We next define a *conductor*,

$$\mathit{Conductor} \overset{\text{def}}{=} \mathbf{1}.\mathbf{1}.\mathbf{1}.\mathbf{1}.\mathbf{1}.\mathbf{1}.\omega^2\overline{\mathit{fini}}$$

Then, in the expression,

$$(M'\ \times\ \mathit{Conductor}) \setminus \{\mathit{fini}, \overline{\mathit{fini}}\}$$

the conductor permits M' only 6 clock ticks, after which it forces termination by expressing a prioritized $\omega^2\overline{\mathit{fini}}$ action. On the other hand, if we run M' without a conductor, it behaves as before:

$$M' \setminus \{\mathit{fini}\}$$

4 Discussion

As with other process algebra, constructing musical systems with MWSCCS benefits from its abstract view of observable behaviour. A primary influence of its design is its use of a small set of powerful process construction operators, combined with an intuitive and flexible model of probability. A working implementation of a music programming language based on MWSCCS has been done (Ross, 1995b). Particular attention was needed to ensure that the $\times$ operator was tractable, as a exponential amount of space and execution time can easily occur. We are also discovering new operators useful in a musical context. Compositions using MWSCCS are being undertaken that exploit interesting chaotic and self-organizing behaviours.

Modeling music as a concurrent activity is well established. There are three main formal models of concurrency in the literature: Petri nets, trace theory, and process algebra. All three approaches share a common mathematical basis, which is reflected by the fact that they essentially define different varieties of finite automata. Both Petri nets (Haus & Sametti, 1992) and trace algebra (Chemillier & Timis, 1988) have been applied successfully to music. The differences between Petri net and MWSCCS models of music result from the inherent differences between PNs and process algebra (Nielsen, 1987). PNs are said to model "true concurrency", in that they make explicit all the simultaneous causal relationships amongst components of a concurrent system. Process algebra view concurrency more abstractly, and causal relationships amongst components are not explicit denoted. The control mechanisms used by PNs and process algebra are also quite different in flavour. Process algebra are considered to be more specification-oriented than PNs, which is advantageous in a music composition setting.

Acknowledgement: Support through NSERC Operating Grant 0138467 is gratefully acknowledged.

References

Chemillier, M., & Timis, D. 1988. Toward a theory of formal musical languages. *Pages 175–183 of: ICMC 95.*

Haus, G., & Sametti, A. 1992. ScoreSynth: A System for the Synthesis of Music Scores Based on Petri Nets and a Music Algebra. *Pages 53–77 of:* Baggi, D. (ed), *Computer-Generated Music.* IEEE Computer Society Press.

Hoare, C. A. R. 1985. *Communicating Sequential Processes.* Prentice–Hall.

Milner, R. 1989. *Communication and Concurrency.* Prentice Hall.

Nielsen, M. 1987. CCS - and its Relationship to Net Theory. *Pages 393–415 of:* Brauer, W. (ed), *Petri Nets: Application and Relationship to Other Models of Concurrency (LNCS 255).* Springer-Verlag.

Ross, B.J. 1995a (February). *A Process Algebra for Stochastic Music Composition.* Tech. rept. CS-95-02. Brock University, Dept. of Computer Science.

Ross, B.J. 1995b. MWSCCS: A Stochastic Concurrent Music Language. *In: Proc. II Brazilian Symposium on Computer Music.*

Tofts, C. 1990. A Synchronous Calculus of Relative Frequency. *In:* Baeten, J.C.M., & J.W.Klop (eds), *CONCUR 90.* Amsterdam, The Netherlands: Springer-Verlag LNCS 458.