Using concurrent constraints process calculi for music modeling

Camilo Rueda

Universidad Javeriana et IRCAM

Mamux, 2006
At a certain level of abstraction:

*Music can be seen as emerging from the interaction of many complex concurrent processes*

Some musical process

*Could be partially determined in its activity or its occurrence*
Computational/mathematical models of music:

*Could be expected to offer convenient ways to define/transform partially defined musical processes*

But

*Few pretend to present a coherent and practical view of them*
Technical Choices

- For a powerful way of expressing interacting processes
  *Concurrent process calculi*

- For dealing with partially defined musical activity
  *Constraint programming*

- For expressing choice (at composition/performance time)
  *non determinism and time awareness*

- For a coherent view
  *Non deterministic Temporal Concurrent Constraint Calculi (NTCC)*
A little History (computational)

- Computation as a **function** became awkward in interactive applications (the 80’s)
- **Process** emerged as a better abstraction
- What is fundamental in a process behavior is the way it **interacts** with others.

*Process Interaction is the fundamental unit of computation* (Milner, 1988)
Processes run **concurrently** performing interactions

- interaction consists in reading/writing through a process **channel**:
  - Milner’s $\pi$-calculus (1990), Abadi-Cardelli’s object calculus (1998)
- Interaction consists in adding/deducing information to/from a global **store**:
  - Saraswat’s **concurrent constraint** programming (1989): cc, tcc, htcc
  - Smolka et al. The Oz programming language (1996)
  - Our group’s contribution:
Most music composition environments are functional: Agon et al OM (1998), Laurson et al PW (1994), Common Lisp Music, ...

Interactive music software usually based on practical (not formal) grounds
Pucket: Max (a CCS vision of Max exists)
A little History (Music applications) (2)

- Constraints music composition environments are based on the CSP model

No attempts (that I know of) to model music using **concurrent process calculi**
Agenda

1. Working Hypothesis
   - Music as concurrent processes
   - Technical Choices

2. Background
   - Constraints as Partial Information
   - NTCC calculus

3. Music Modeling in NTCC

4. NTCC and Temporal Logic

5. Examples of Musical Models in NTCC
Partial Information in two dimensions

Figure: Partially Determined Processes
Computing with Partial information

Processes,

- compute partial information in the form of constraints, 
  \( \text{Pitch}(\text{Note}_1) \in \{60, 64, 67\} \)
- accumulate it in a global store
- synchronize by data on shared variables
  \textbf{ask} \text{Pitch}(\text{Note}_1) > 64 \textbf{then} \textit{tell}(\text{Duration}(\text{Note}_2) < 100)
Computing with partial information

Tell pitch > 40
Tell pitch < 69
Ask pitch = 60 then P
Ask pitch < 72 then Q
STORE

calcul de processus concurrents pour la musique
Computing with partial information

- Ask pitch = 60 then P
- Ask pitch < 72 then Q
- Tell pitch < 69
- pitch > 40

Calcul de processus concurrents pour la musique
Computing with partial information

pitch > 40
pitch < 69

Ask pitch < 72 then Q
Ask pitch = 60 then P
Computing with partial information

Ask pitch=60 then P

pitch > 40

pitch < 69

Q
Discrete time:

- Time is considered a sequence of (discrete) time units
- At each time unit, a CC computation takes place
- Computation in the next time unit starts with a fresh store
- A process may schedule its continuation to the next time unit.
Discrete Time Evolution

- **Input Store**: $S_1 ightarrow P_1 ightarrow S'_1$
- **Process**: $P_1 ightarrow P_2 ightarrow P_3$
- **Output Store**: $S'_2 ightarrow S_3$

**Scheduled processes**

TIME
Components of NTCC(1)

- **Adding** information:

  \[ \text{tell}(pitch \in \{60, 64, 67\}) \]

- **Asking** for information:

  \[ \text{when } (pitch \neq 70) \text{ do ACCEL} \]

- **Defining** information to be **local**:

  \[ \text{local } x \text{ in } (\text{tell}(x < pitch_3) \parallel \text{PLAY}) \]

- **Schedule** a process for the next time unit:

  \[ \text{next tell}(duration = 50) \]
Components of NTCC(2)

- Trigger an action on **absence** of information:
  
  \[
  \text{unless } start > clock + 1 \text{ next } \text{tell}(start = clock)
  \]

- Delay a process or launch a **persistent** process
  
  \[
  * \text{tell}(\text{play}(\text{stop})) \quad \text{! tell}(\text{start}(\text{Note}_1) < \text{start}(\text{Note}_2))
  \]

- Non deterministically **choose** an action
  
  \[
  \sum_{i \in \{1,2,3\}} \text{when } \text{pitch}_i > 48 \text{ do } \text{tell}(\text{duration}_i < 100)
  \]
### The ntcc calculus

<table>
<thead>
<tr>
<th>Agent</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tell(c)</code></td>
<td>Add c to the current store</td>
</tr>
<tr>
<td><code>when c do A</code></td>
<td>if c holds now, run A</td>
</tr>
<tr>
<td><code>local x in P</code></td>
<td>run P with local x</td>
</tr>
<tr>
<td>`A</td>
<td></td>
</tr>
<tr>
<td><code>next A</code></td>
<td>run A at the next instant</td>
</tr>
<tr>
<td><code>unless c next A</code></td>
<td>unless c can be inferred now, run A</td>
</tr>
<tr>
<td><code>∑_{i∈I} when c_i do P_i</code></td>
<td>choose P_i s.t. c_i holds</td>
</tr>
<tr>
<td><code>* P</code></td>
<td>delay P indefinitely (not forever)</td>
</tr>
<tr>
<td><code>! P</code></td>
<td>Execute P each time unit (from now)</td>
</tr>
</tbody>
</table>

Calcul de processus concurrents pour la musique
Musical models in NTCC

- “Conductor” processes output signals at different rates (time beating).
- Musical activity processes synchronize on these signals.
- Musical output can be partially specified (constraints).
- Default processes fill the gap between signals.
### Computing in NTCC

\[ \text{SYST} \overset{\text{def}}{=} \text{BEAT}(0) \parallel \text{CHECK} \parallel \text{PLAY} \]
\[ \parallel \ast [50,200] \text{tell}(\text{play(done)}) \]

\[ \text{PLAY} \overset{\text{def}}{=} ! \sum_{i \in \{1,2,3\}} \text{when}\ \text{play(on)}\ \text{do}\ \text{NOTE}_i \]
\[ \parallel ! \text{tell}(d = 20) \]

\[ \text{CHECK} \overset{\text{def}}{=} ! \text{when}\ \text{beat} \mod d = 0\ \text{do}\ \text{tell}(\text{play(on)}) \]

\[ \text{BEAT}(i) \overset{\text{def}}{=} \text{tell}(\text{beat} = i) \]
\[ \parallel \text{unless}\ \text{play(done)}\ \text{next}\ \text{BEAT}(i + 1) \]

**Calcul de processus concurrents pour la musique**
What is observed of a NTCC process

- **input/output behavior**

- **Output behavior**: no interactions

- **Strongest postcondition**: all possible output sequences under arbitrary inputs.
How to verify properties of a NTCC process

- There is an associated linear-time logic

\[ A ::= c \mid A \Rightarrow A \mid \neg A \mid \exists x A \mid \Diamond A \mid \Box A \mid \Diamond A. \]

- Properties of processes are expressed as formulae in this logic
  \[ \Diamond pitch(Note_1) - pitch(Note_2) = 3 \]

- Using the logic see if the observed behavior of the process satisfies the formula

- There is a proof system for the above
Two Musical Models in NTCC

- **Controlled improvisation**
  - The number of variables (notes) is not known in advance.
  - Non determinism used to express rhythmic choice.

- **Interactive score**
  - Partial information on temporal location of musical structures
  - Structures can be controlled by the occurrence of events
Interactive score

- A collection of **temporal objects** (including discrete events) linked by temporal relations (e.g. Allen)

Diagram:
- T0
- T1
- T2
- T3
- T4
- T5
- T6
- T7
- s1
- s2
- \[\Delta_0, \Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_5, \Delta_6\]
- [\min, \max]
- overlaps

Calculus de processus concurrents pour la musique
Temporal Objects as NTCC Processes

- An interactive score process:
  - Posts Allen constraints between TO’s (persistent)
  - Launches each TO process
  - Launches a time beating process

- Each TO process
  - Decides whether it should start/stop playing (“texture” TO)
  - Synchronizes on the occurrence of an event (“event” TO)
Behavior of a TO

SCORE

Allen relations & clock

0 1 k k+1 n

TEMP. OBJECT

local object constraints

can't deduce shouldn't start can deduce duration elapsed

Play

Allen relations & clock

calcul de processus concurrents pour la musique
A TO in NTCC

\[ TO_i \overset{\text{def}}{=} ! \text{tell}(c_i) \]
\[ || ! \text{unless} \ \text{clock} + 1 < s_i \ \text{next tell}(\text{clock} \geq s_i) \]
\[ || ! \text{when} \ \text{clock} = s_i \ \text{do} \ P_i \]
\[ || ! \text{when} \ \text{clock} \geq s_i \ \text{do} (\text{Same}_i \ || \ ! \text{unless} \ \text{clock} \geq s_i + \Delta_i \ \text{next} \ P_i) \]

\[ EV_i \overset{\text{def}}{=} ! \text{tell}(c_i) \]
\[ || ! \text{when} \ \text{event}_i(\text{on}) \ \text{do} \]
\[ ( ! \text{unless} \ \text{clock} + 1 < s_i \ \text{next tell}(\text{clock} \geq s_i) \]
\[ || ! \text{when} \ \text{clock} \geq s_i \ \text{do} \ \text{next Same}_i) \]

\[ \text{Trigger}_i \overset{\text{def}}{=} *[0..n] \ \text{tell}(\text{event}_i(\text{on})) \]
Controlled Improvisation

A group of musicians each playing sequences of three notes. Each musician is given a pattern of three delays.

Each Musician,

- plays a block of three notes separated by the delays taken in some order.
- waits some time (not greater than the sum of all pattern durations)
- then goes on to play another block, provided all the other musicians have already played theirs

Playing continues until all musicians play a note at the same time.
NTCC improvisation processes

**MUSICIAN**
- start
- Go
- enabled
- start & enabled
- start
- enabled
- enabled
- Play

**CONDUCTOR**
- all done
- all played
- go
- all played
- all played
- all played
- all played
- all played
- all played
Working Hypothesis
Background
Music Modeling in NTCC
NTCC and Temporal Logic
Examples of Musical Models in NTCC

controlled improvisation in NTCC

\[ \text{Musician}_i \overset{\text{def}}{=} \]
\begin{align*}
&\lnot \text{when start}_i \land \text{enabled}_i \rightarrow ( \text{Play}_i \parallel * \text{[length,pdur]} \text{ tell}(\text{start}_i) ) \\
&\lnot \text{when start}_i \rightarrow \text{do unless enabled}_i \text{ next tell}(\text{start}_i) \\
&\lnot \text{when enabled}_i \rightarrow \text{do unless start}_i \text{ next tell}(\text{enabled}_i) \\
&\lnot \text{when GO do tell}(\text{enabled}_i) \\
\end{align*}

\[ \text{Play}_i \overset{\text{def}}{=} \]
\begin{align*}
&\sum_{\text{perm}(j,k,l)} ( \text{next}^j \text{ tell}(\text{c}_i(\text{Note}_i)) \parallel \text{next}^{J+k} \text{ tell}(\text{c}_i(\text{Note}_i)) \\
&\parallel \text{next}^{j+k+l} (\text{tell}(\text{c}_i(\text{Note}_i)) \parallel \text{tell}(\text{done}_i) ) ) \\
\end{align*}

\[ \text{Conductor} \overset{\text{def}}{=} ! \text{when } \bigwedge_{i \in 1..m} \text{note}_i > 0 \rightarrow \text{do tell}(\text{end}) \\
! \text{when } \bigwedge_{i \in 1..m} \text{done}_i \rightarrow \text{do unless end next tell}(\text{GO}) \\
\Pi_{i \in 1..m} ! \text{when done}_i \rightarrow \text{do unless Alldone next done}_i \\
! \text{unless end next tell}(\text{noEnd}) \\
\]
let

\( \text{SYSTEM} = \text{Musician}_i \parallel \text{Play}_i \parallel \text{Conductor} \)

- Regardless of choices performance ends
  \( \text{SYSTEM} \vdash \Diamond \text{end} \)
- There are choices leading to performance ending. No proof of:
  \( \text{SYSTEM} \vdash \Box \text{noEnd} \)
Research directions

- modeling more complex musical system in ntcc:
  - Improvisation systems based on bounded history of musical events (ref. Assayag & Chemillier)
  - Tackling real-time performance of interactive scores (ongoing)

- Fully modeling an audio streaming architecture

once ntcc is extended to be better equipped with:

- stochastic notions (maintaining its clean semantics),
- suitable constraint systems,
- a much better compiler and supporting tools
Merci !
Operational Semantics

**Internal Transitions:**

\[ RT \langle \text{tell}(c), a \rangle \rightarrow \langle \text{skip}, a \land c \rangle \]

\[ RG \langle \sum_{i \in I} \text{when } c_i \text{ do } P_i, a \rangle \rightarrow \langle P_j, a \rangle \]

\[ RB \langle ! P, a \rangle \rightarrow \langle P \parallel \text{next} ! P, a \rangle \]

\[ RS \langle \star P, a \rangle \rightarrow \langle \text{next}^n P, a \rangle^{(n \geq 0)} \]

**Observable Transition**

\[ RO \langle P, a \rangle \rightarrow^* \langle Q, a' \rangle \rightarrow \]

\[ P \xrightarrow{(a,a')} F(Q) = \begin{cases} 
Q' & \text{if } Q = \text{next } Q' \\
Q' & \text{if } Q = \text{unless } (c) \text{ next } Q' \\
F(Q_1) \parallel F(Q_2) & \text{if } Q = Q_1 \parallel Q_2 \\
\text{local } x \text{ in } F(Q') & \text{if } Q = \text{local } x \text{ in } Q' \\
\text{skip} & \text{otherwise}
\end{cases} \]
The Associated Logic

An LTL Process Logic

**Syntax.** $A := c | A \land A | \neg A | \exists x A | \diamond A | \diamond A | \square A$

- $c$ means “$c$ holds in the current time unit”.
- $\square A$ means “$A$ holds always”.
- $\diamond A$ means “$A$ eventually holds”.
- $\Diamond A$ means “$A$ holds in the next time unit”

**Semantics.** Say $\alpha = c_1.c_2.\ldots \models A$ iff $\langle \alpha, 1 \rangle \models A$ where

- $\langle \alpha, i \rangle \models c$ iff $c_i \vdash c$
- $\langle \alpha, i \rangle \models \neg A$ iff $\langle \alpha, i \rangle \not\models A$
- $\langle \alpha, i \rangle \models A_1 \land A_2$ iff $\langle \alpha, i \rangle \models A_1$ and $\langle \alpha, i \rangle \models A_2$
- $\langle \alpha, i \rangle \models \Diamond A$ iff $\langle \alpha, i + 1 \rangle \models A$
- $\langle \alpha, i \rangle \models \square A$ iff for all $j \geq i$ $\langle \alpha, j \rangle \models A$
- $\langle \alpha, i \rangle \models \diamond A$ iff there exists $j \geq i$ s.t. $\langle \alpha, j \rangle \models A$
- $\langle \alpha, i \rangle \models \exists x A$ iff there is $\alpha'$ variant of $\alpha$ s.t. $\langle \alpha', i \rangle \models A$. 