Using concurrent constraints process calculi for music modeling

Camilo Rueda

Universidad Javeriana et IRCAM

Mamux, 2006

Working Hypothesis

- 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

Working Hypothesis(2)

- 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)

A little History (2)

Processes run concurrently performing interactions

- interaction consists in reading/writing through a process channel:
  Milner's π-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:
    PiCo (2000), NTCC (Frank Valencia, 2002)

A little History (Music applications)

- 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
  Truchet-Codognet: OM-Clouds (2002), Rueda: Situation (1996), Laurson: PW-Constraints (1998), others

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

Agenda

Partial Information in two dimensions

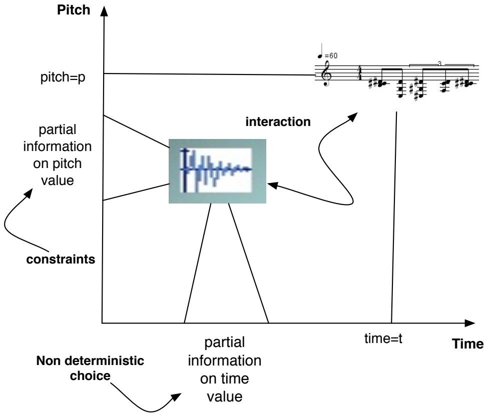


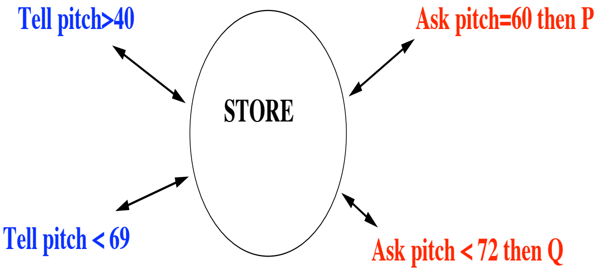
Figure: Partially Determined Processes

Computing with Partial information

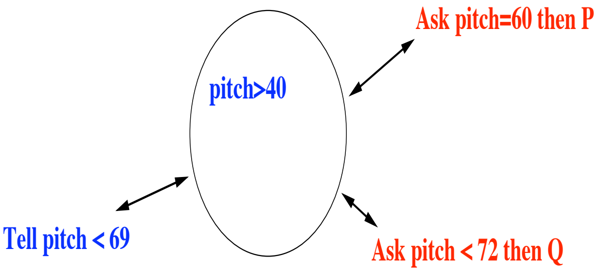
Processes,

- compute partial information in the form of constraints,
  $Pitch(Note_1) \in \{60, 64, 67\}$
- accumulate it in a global store
- synchronize by data on shared variables
  $\textbf{ask}\ Pitch(Note_1) > 64\ \textbf{then}\ tell(Duration(Note_2) < 100)$

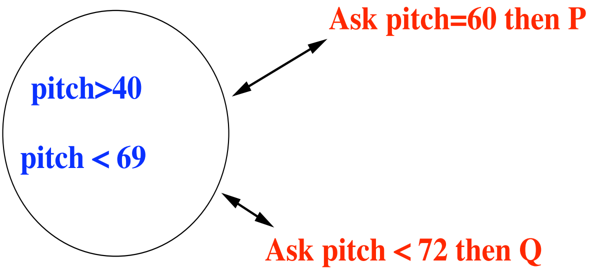
Computing with partial information



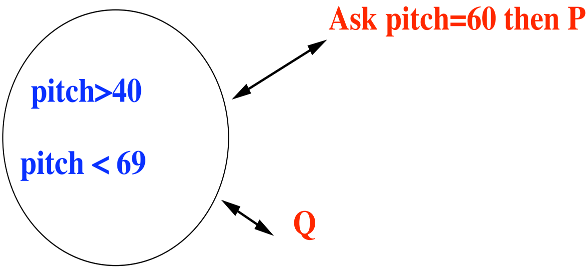
Computing with partial information



Computing with partial information



Computing with partial information

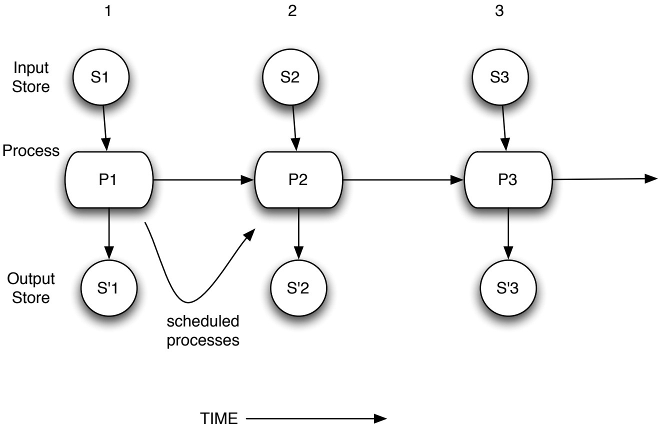


Adding the notion of time (ntcc calculus)

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



Components of NTCC(1)

- Adding information:

$$\textit{tell}(pitch \in \{60, 64, 67\})$$

- Asking for information:

$$\textit{when}\ (pitch \neq 70)\ \textit{do}\ ACCEL$$

- Defining information to be local:

$$\textit{local}\ x\ \textit{in}\ (\ \textit{tell}(x < pitch_3)\ \parallel\ PLAY\ )$$

- schedule a process for the next time unit:

$$\textit{next}\ \textit{tell}(duration = 50)$$

Components of NTCC(2)

- Trigger an action on absence of information:

$$\textit{unless}\ start > clock + 1\ \textit{next tell}(start = clock)$$

- Delay a process or launch a persistent process

$$*\ \textit{tell}(play(stop)) \qquad !\ \textit{tell}(start(Note_1) < start(Note_2))$$

- non deterministically choose an action

$$\sum_{i \in \{1,2,3\}} \textit{when}\ pitch_i > 48\ \textit{do tell}(duration_i < 100)$$

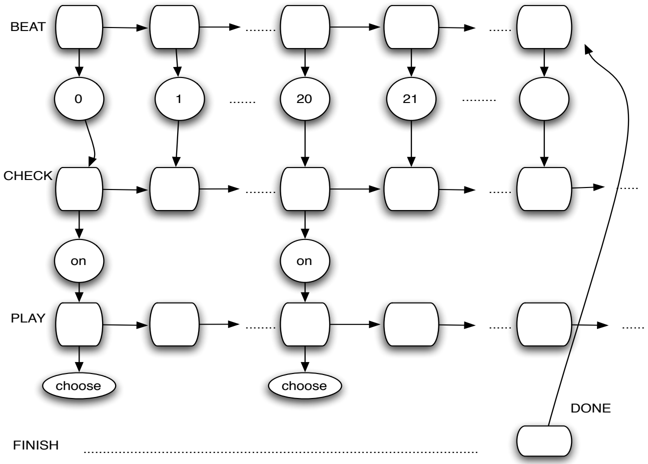
The ntcc calculus

Agent	meaning
tell(c)	Add c to the current store
when c *do* A	if c holds now, run A
local x *in* P	run P with local x
$A \parallel B$	Parallel composition
next A	run A at the next instant
unless c *next* A	unless c can be inferred now, run A
$\sum_{i \in I}$ *when* c_i *do* P_i	choose P_i s.t. c_i holds
$* P$	delay P undefinitely (not forever)
$! P$	Execute P each time unit (from now)

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

Musical Processes Sync

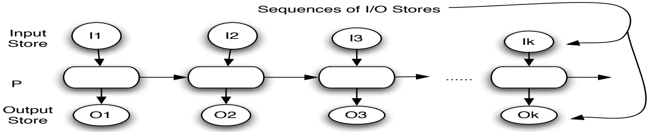


Computing in NTCC

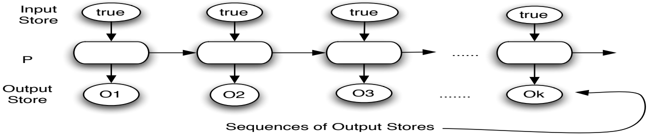
$SYST \stackrel{\text{def}}{=} BEAT(0) \parallel CHECK \parallel PLAY$ $\parallel *_{[50,200]}\ \textbf{tell}(play(done))$
$PLAY \stackrel{\text{def}}{=}\ !\sum_{i \in \{1,2,3\}} \textbf{when}\ play(on)\ \textbf{do}\ NOTE_i$ $\parallel\ !\ \textbf{tell}(d = 20)$
$CHECK \stackrel{\text{def}}{=}\ !\ \textbf{when}\ beat\ \mathbf{mod}\ d = 0\ \textbf{do}\ \textbf{tell}(play(on))$
$BEAT(i) \stackrel{\text{def}}{=} \textbf{tell}(beat = i)$ $\parallel\ \textbf{unless}\ play(done)\ \textbf{next}\ BEAT(i+1)$

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

$$A ::= c \mid A \Rightarrow A \mid \neg A \mid \exists_x A \mid \circ A \mid \Box A \mid \Diamond A.$$

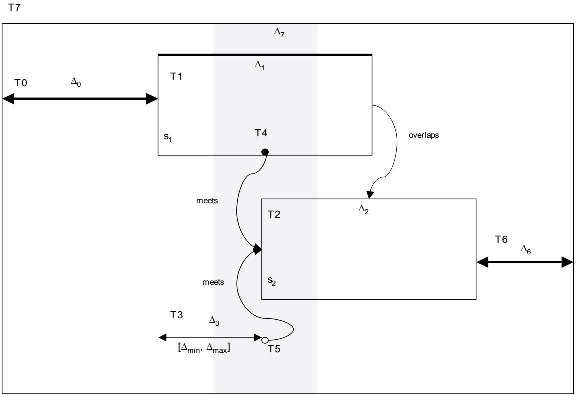
- Properties of processes are expressed as formulae in this logic
  $\Diamond pitch(Note_1) - pitch(Note_2) = 3$
- Using the logic see if the observed behavior of the process satisfies the formula
- There is a proof system for the above

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

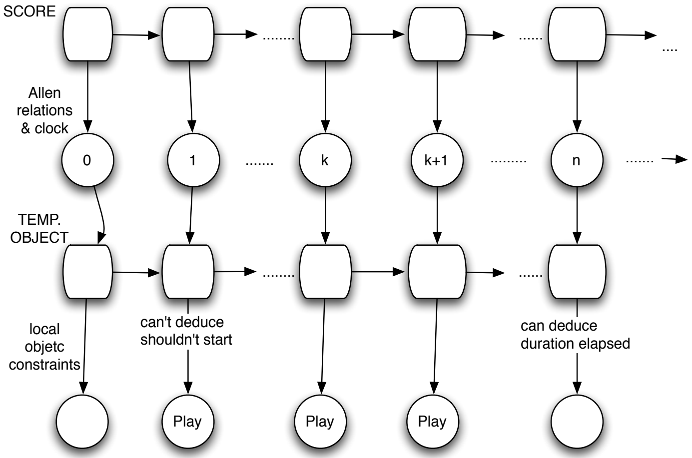
- A collection of temporal objects (including discrete events) linked by temporal relations (e.g. Allen)



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



A TO in NTCC

$$
\begin{aligned}
TO_i \stackrel{\text{def}}{=}\ & !\ \textit{tell}(c_i) \\
\parallel\ & !\ \textit{unless}\ clock + 1 < s_i\ \textit{next}\ \textit{tell}(clock \geq s_i) \\
\parallel\ & !\ \textit{when}\ clock = s_i\ \textit{do}\ P_i \\
\parallel\ & !\ \textit{when}\ clock \geq s_i\ \textit{do}\ (Same_i\ \parallel\ \textit{unless}\ clock \geq s_i + \Delta_i\ \textit{next}\ P_i\ )
\end{aligned}
$$

$$
\begin{aligned}
EV_i \stackrel{\text{def}}{=}\ & !\ \textit{tell}(c_i) \\
\parallel\ & !\ \textit{when}\ event_i(on)\ \textit{do} \\
& \quad (\ !\ \textit{unless}\ clock + 1 < s_i\ \textit{next}\ \textit{tell}(clock \geq s_i) \\
& \quad \parallel\ !\ \textit{when}\ clock \geq s_i\ \textit{do}\ \ \textit{next}\ Same_i\ )
\end{aligned}
$$

$$
Trigger_i \stackrel{\text{def}}{=}\ *_{[0..n]}\ \textit{tell}(event_i(on))
$$

Controlled Improvisation

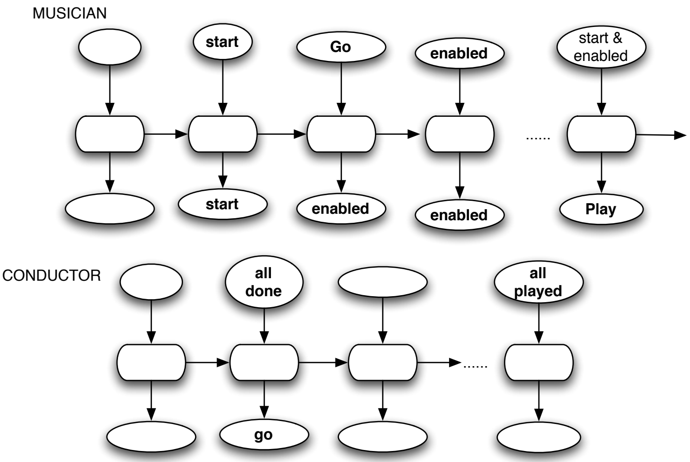
A group of musicians each playing sequences of three notes. Each musician is given a pattern of three delays.

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



controlled improvisation in NTCC

$$
\begin{aligned}
&\mathit{Musician}_i \stackrel{\text{def}}{=} \\
&\quad !\ \textbf{when}\ \mathit{start}_i \wedge \mathit{enabled}_i\ \textbf{do}\ (\ \mathit{Play}_i\ \parallel\ *_{[\mathit{length},\mathit{pdur}]}\ \textbf{tell}(\mathit{start}_i)\ ) \\
&\quad !\ \textbf{when}\ \mathit{start}_i\ \textbf{do unless}\ \mathit{enabled}_i\ \textbf{next tell}(\mathit{start}_i) \\
&\quad !\ \textbf{when}\ \mathit{enabled}_i\ \textbf{do unless}\ \mathit{start}_i\ \textbf{next tell}(\mathit{enabled}_i) \\
&\quad !\ \textbf{when}\ \mathit{GO}\ \textbf{do tell}(\mathit{enabled}_i) \\
&\mathit{Play}_i \stackrel{\text{def}}{=} \\
&\ \textstyle\sum_{\mathit{perm}(j,k,l)}\ (\ \textbf{next}^{j}\ \textbf{tell}(c_i(\mathit{Note}_i))\ \parallel \textbf{next}^{J+k}\ \textbf{tell}(c_i(\mathit{Note}_i)) \\
&\qquad\qquad \parallel\ \textbf{next}^{j+k+l}\ (\textbf{tell}(c_i(\mathit{Note}_i))\ \parallel\ \textbf{tell}(\mathit{done}_i)\ )) \\
&\mathit{Conductor} \stackrel{\text{def}}{=}\ !\ \textbf{when}\ \textstyle\bigwedge_{i\in 1..m} \mathit{note}_i > 0\ \textbf{do tell}(\mathit{end}) \\
&\qquad\qquad !\ \textbf{when}\ \textstyle\bigwedge_{i\in 1..m} \mathit{done}_i\ \textbf{do unless}\ \mathit{end}\ \textbf{next tell}(\mathit{GO}) \\
&\qquad\quad \textstyle\prod_{i\in 1..m}\ !\ \textbf{when}\ \mathit{done}_i\ \textbf{do unless}\ \mathit{Alldone}\ \textbf{next}\ \mathit{done}_i \\
&\qquad\quad !\ \textbf{unless}\ \mathit{end}\ \textbf{next tell}(\mathit{noEnd})
\end{aligned}
$$

Proving Properties

let

$SYSTEM = Musician_i \parallel Play_i \parallel Conductor$

- Regardless of choices performance ends
  $SYSTEM \vdash \Diamond end$
- There are choices leading to performance ending. No proof of:
  $SYSTEM \vdash \Box noEnd$

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 (on going)
- 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

Operational Semantics

- **Internal Transitions:**

$$RT \quad \langle \mathbf{tell}(c), a\rangle \longrightarrow \langle \mathbf{skip}, a \wedge c\rangle \qquad RG \quad \frac{a \vdash c_j}{\langle \sum_{i \in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i, a\rangle \longrightarrow \langle P_j, a}$$

$$RB \quad \frac{}{\langle !\, P, a\rangle \longrightarrow \langle P \parallel \mathbf{next}\, !\, P, a\rangle} \qquad RS \quad \frac{}{\langle \star\, P, a\rangle \longrightarrow \langle \mathbf{next}^n P, a\rangle}\,(n \geq 0)$$

- **Observable Transition**

$$RO \quad \frac{\langle P, a\rangle \longrightarrow^* \langle Q, a'\rangle \not\longrightarrow}{P \xRightarrow{(a,a')} \mathbf{F}(Q)} \quad = \begin{cases} Q' & \text{if } Q = \mathbf{next}\ Q' \\ Q' & \text{if } Q = \mathbf{unless}\ (c)\ \mathbf{next}\ Q' \\ \mathbf{F}(Q_1) \parallel \mathbf{F}(Q_2) & \text{if } Q = Q_1 \parallel Q_2 \\ \mathbf{local}\ x\ \mathbf{in}\ \mathbf{F}(Q') & \text{if } Q = \mathbf{local}\ x\ \mathbf{in}\ Q' \\ \mathbf{skip} & \text{otherwise} \end{cases}$$

The Associated Logic

An LTL Process Logic

Syntax. $A := c \mid A \wedge A \mid \neg A \mid \exists_x A \mid \circ A \mid \Diamond A \mid \Box A$

- c means "c holds in the current time unit".
- $\Box A$ means "A holds always".
- $\Diamond A$ means "A eventually holds"
- $\circ A$ 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	$\langle \alpha, i \rangle \not\models A$
$\langle \alpha, i \rangle \models A_1 \wedge A_2$	iff	$\langle \alpha, i \rangle \models A_1$ and $\langle \alpha, i \rangle \models A_2$
$\langle \alpha, i \rangle \models \circ A$	iff	$\langle \alpha, i+1 \rangle \models A$
$\langle \alpha, i \rangle \models \Box A$	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 α' $x$variant of α s.t. $\langle \alpha', i \rangle \models A$.