

The Antescofo Language in ReactiveML

Guillaume Baudart, ENS

Florent Jacquemard, IRCAM

Louis Mandel, Collège de France

Marc Pouzet, ENS

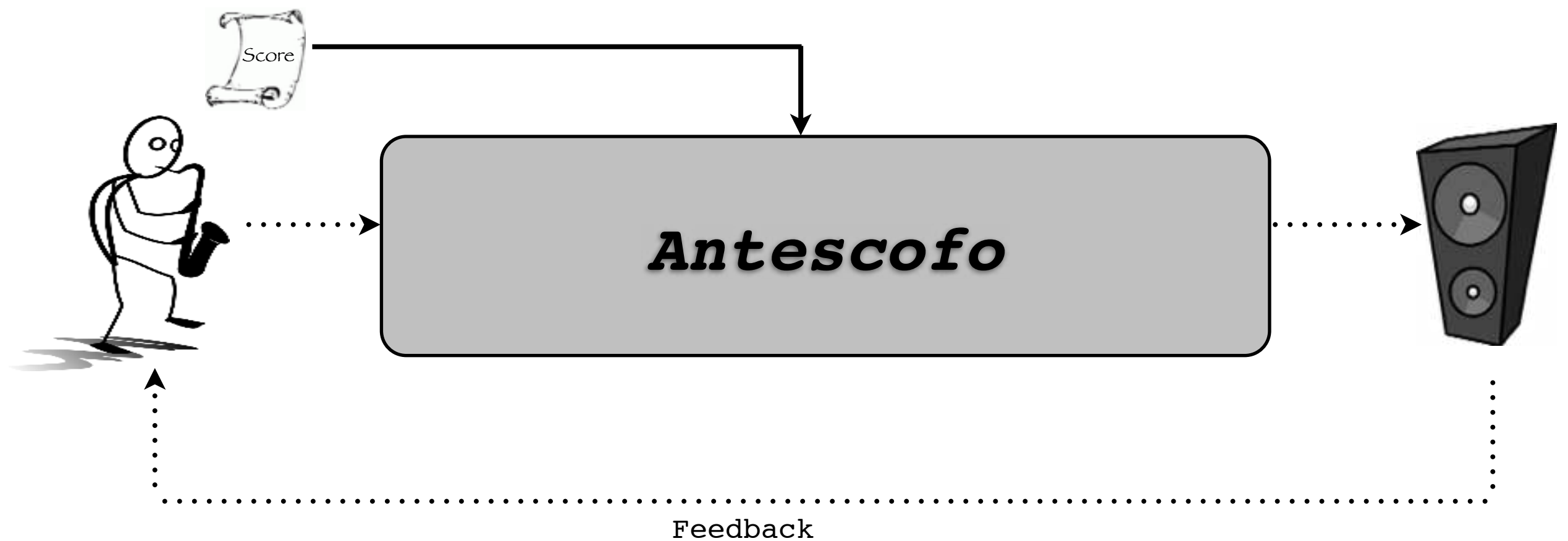
Mixed Music and Antescofo

[Cont 2008]



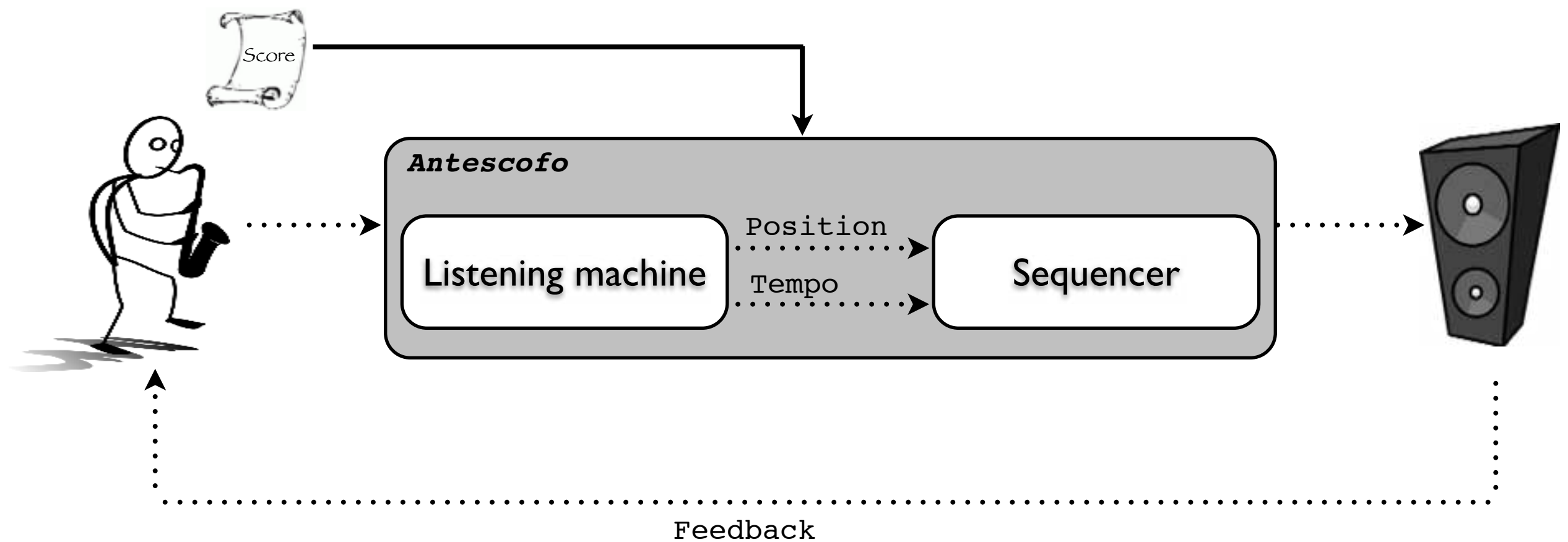
Mixed Music and Antescofo

[Cont 2008]



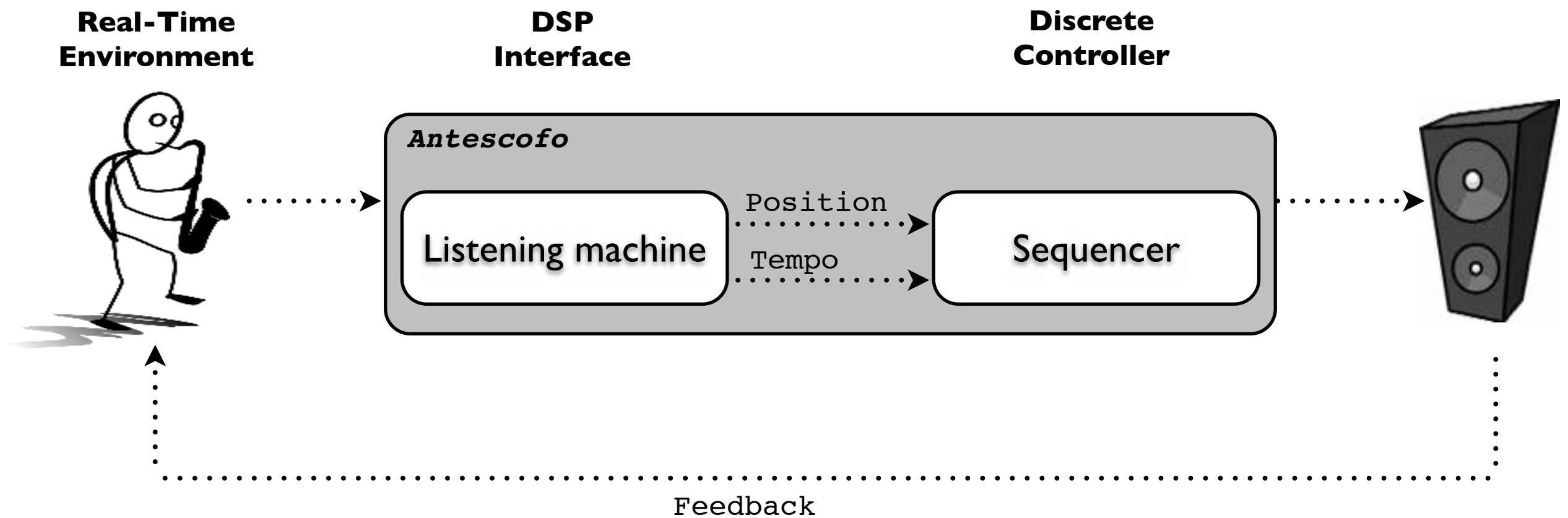
Antescofo Architecture

[Cont 2008]



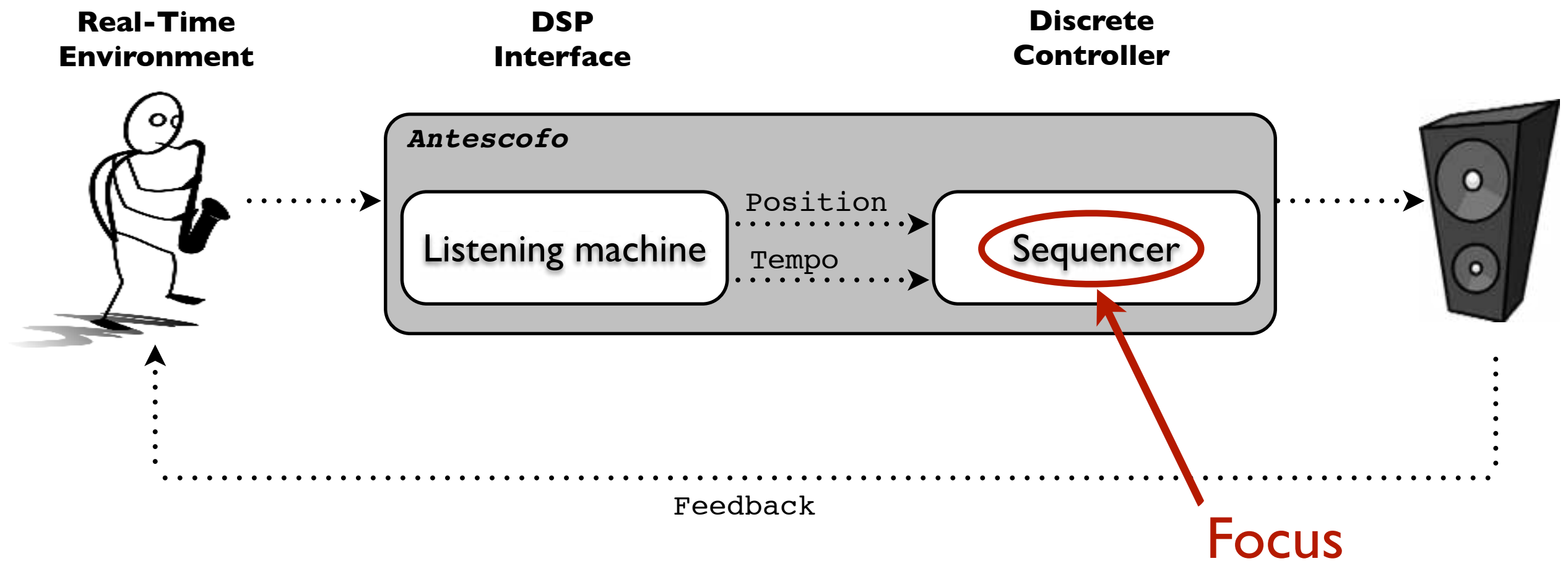
Antescofo Architecture

[Cont 2008]



Antescofo Architecture

[Cont 2008]



Motivations

- **Link with the synchronous model**
 - An executable semantics for Antescofo
 - Embedding in a synchronous reactive language
- **Benefits**
 - Live coding
 - Prototyping new features:
new attributes, reactive behaviors, ...

I. The Antescofo Language

- Description
- Synchronization and error handling strategies

II. Semantics

- Formalization
- The three predicates

III. Implementation Architecture

- Architecture
- Embedding in ReactiveML

IV. Applications

- Live Coding
- New reactive behaviors

The Antescofo Language

The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

Anthèmes II (1994)

Libre brusque
(♩ = 92)

Pierre Boulez
(*1925)

(♩ = 92) *rall.* (♩ = 66)
batt. (archet normal)

Violon
f *fff* *mf* *ff*

Spatialization: F -11/-18/-18/2.0

1 2 3 4 5 6 7 8 9

Inf. Rev.
reverb. time: 60"

Spatialization: F -11/-18/-18/2.0

Sampl. IR
MIDI: 93 90 85 84 82 80 75 77 76 75 74
reverb. time: 60"

Spatialization: F -11/-18/-18/2.0

Sampler
pizz. = 93 msec.
MIDI: [74 73 70 69 68 67 66 65]
[74 73] [74 71 70] [68 70 73 74] [74 73 72 69 68] [67 68 71 72 73 74] [63 64 67 68 69 70 71 74]
[74 73 72 71 70 67 66]

Spatialization: MR -4/-12/-24/2.0

Freq. Shift.

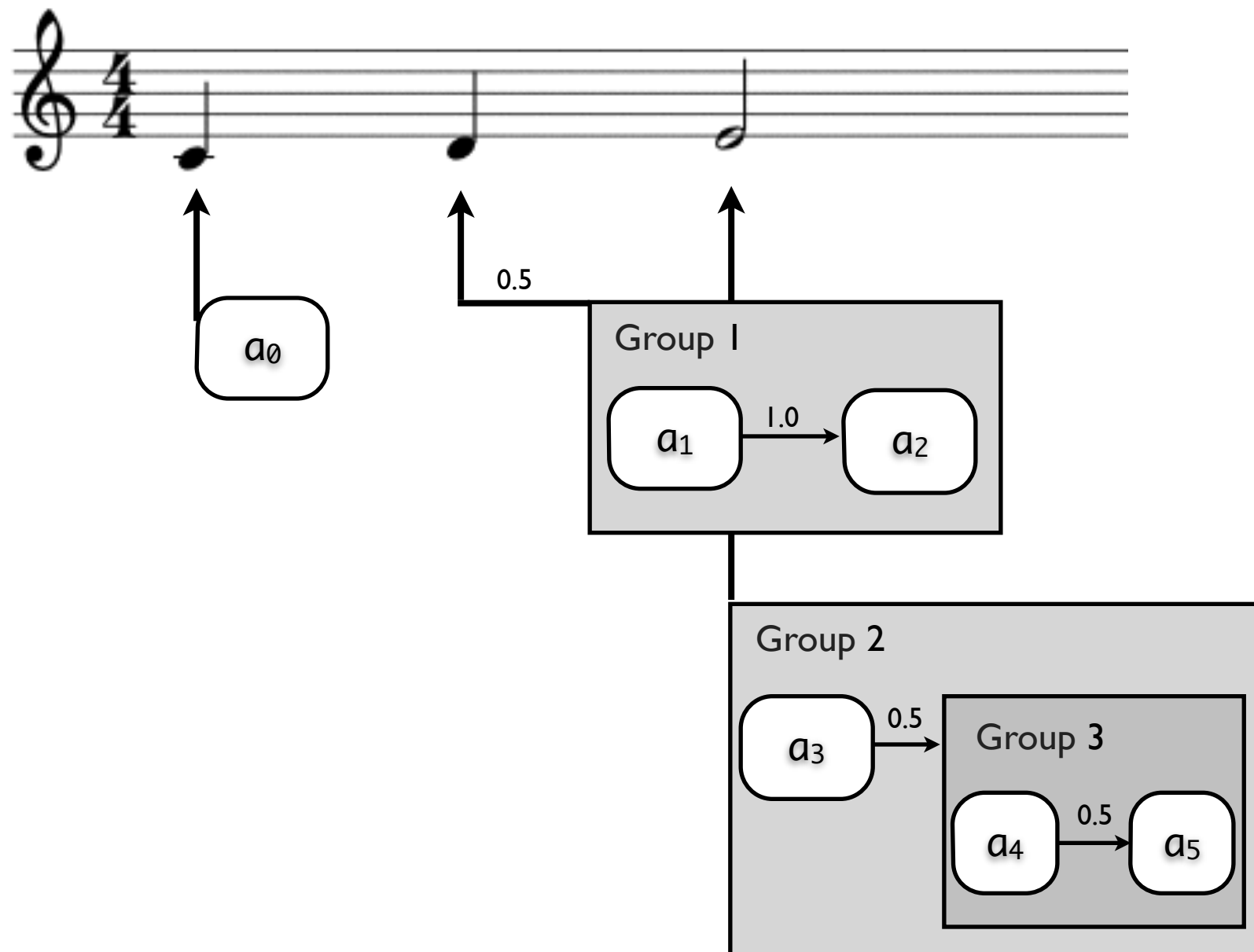
Spatialization:

New version using antescofo (2008)

The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]



The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]

NOTE 65 1.0

0.25 GROUP tight partial

{ 1.0 'a_11'
1.0 'a_12' }

CHORD (68 54) 0.5

1.0 'a_21'

0.5 GROUP loose causal

{ 1.0 'a_22'
0.0 GROUP loose causal
{ 0.25 'a_23'
0.25 'a_24' }
1.0 'a_25' }

NOTE 52 2.0

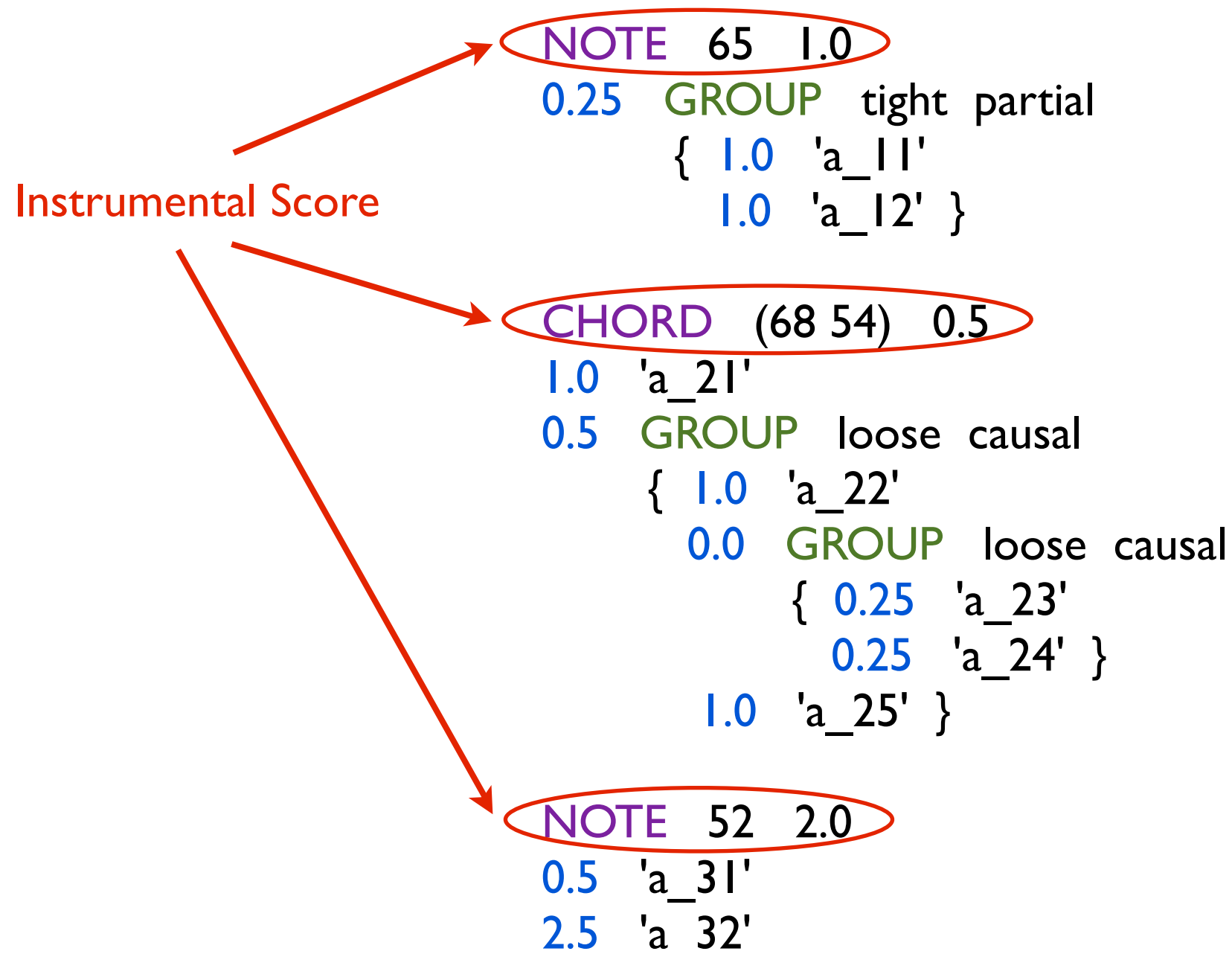
0.5 'a_31'

2.5 'a_32'

The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]



The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]

NOTE 65 1.0

0.25 GROUP tight partial
{ 1.0 'a_11'
1.0 'a_12' }

CHORD (68 54) 0.5

1.0 'a_21'
0.5 GROUP loose causal
{ 1.0 'a_22'
0.0 GROUP loose causal
{ 0.25 'a_23'
0.25 'a_24' }
1.0 'a_25' }

NOTE 52 2.0

0.5 'a_31'
2.5 'a_32'

Electronic Score

The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]

```
NOTE 65 1.0
0.25 GROUP tight partial
{ 1.0 'a_11'
  1.0 'a_12' }
```

```
CHORD (68 54) 0.5
1.0 'a_21'
0.5 GROUP loose causal
{ 1.0 'a_22'
  0.0 GROUP loose causal
    { 0.25 'a_23'
      0.25 'a_24' }
    1.0 'a_25' }
```

```
NOTE 52 2.0
0.5 'a_31'
2.5 'a_32'
```

Delay relative to the tempo

The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]

NOTE 65 1.0

0.25 GROUP **tight partial**
{ 1.0 'a_11'
1.0 'a_12' }

CHORD (68 54) 0.5

1.0 'a_21'

0.5 GROUP **loose causal**
{ 1.0 'a_22'
0.0 GROUP **loose causal**
{ 0.25 'a_23'
0.25 'a_24' }
1.0 'a_25' }

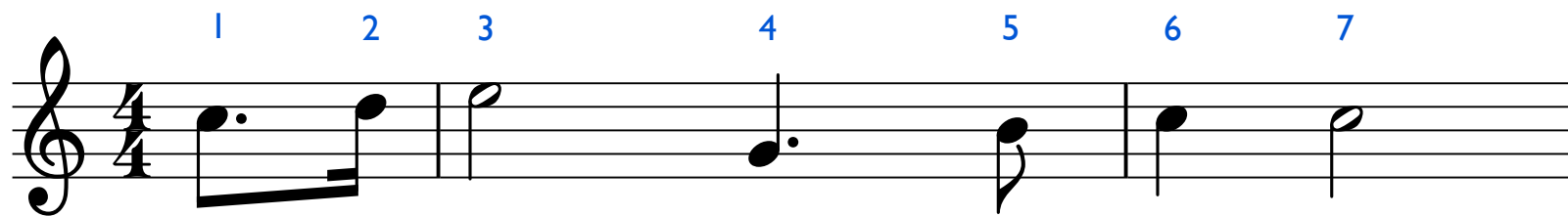
NOTE 52 2.0

0.5 'a_31'

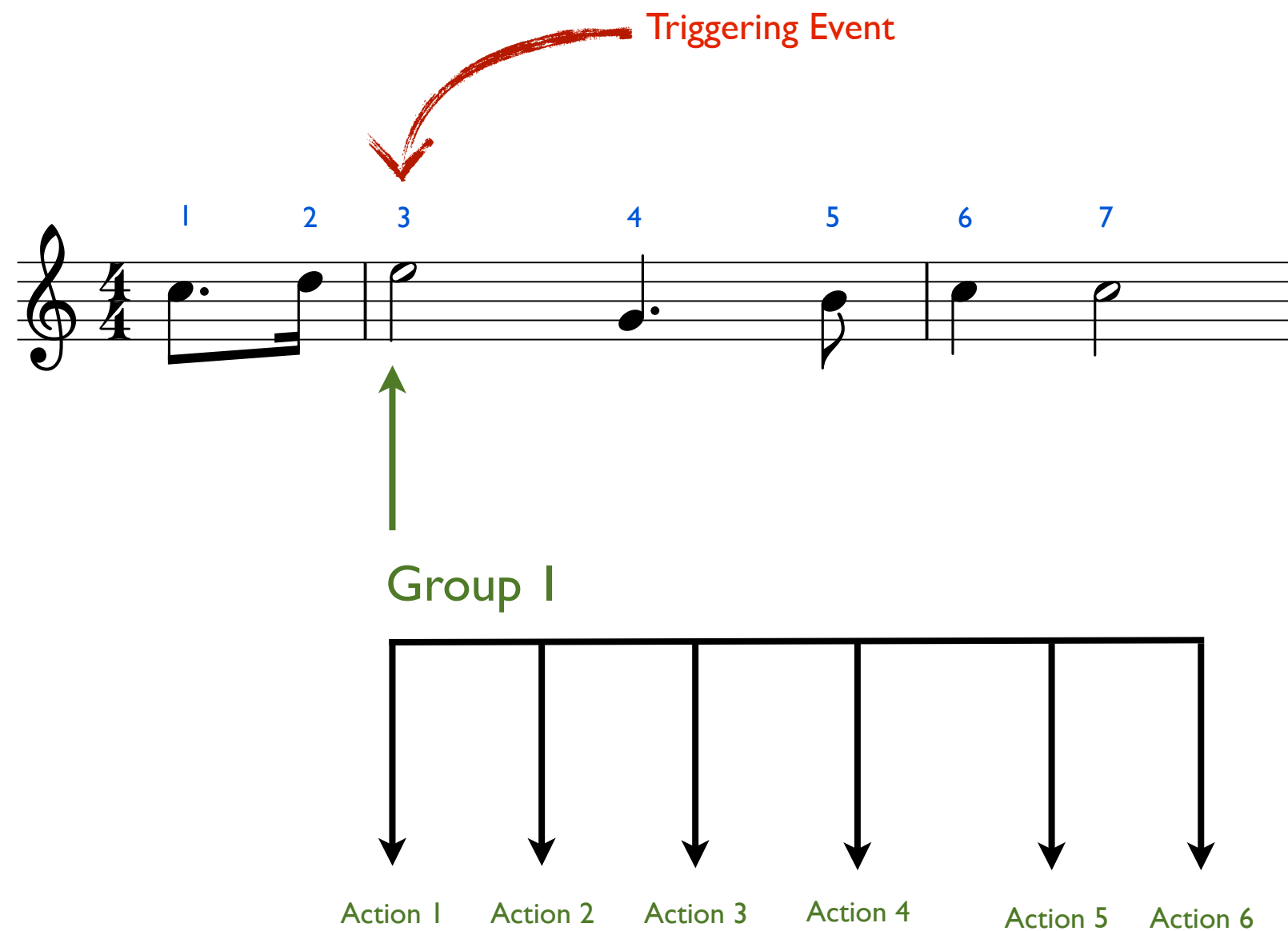
2.5 'a_32'

Group Attributes

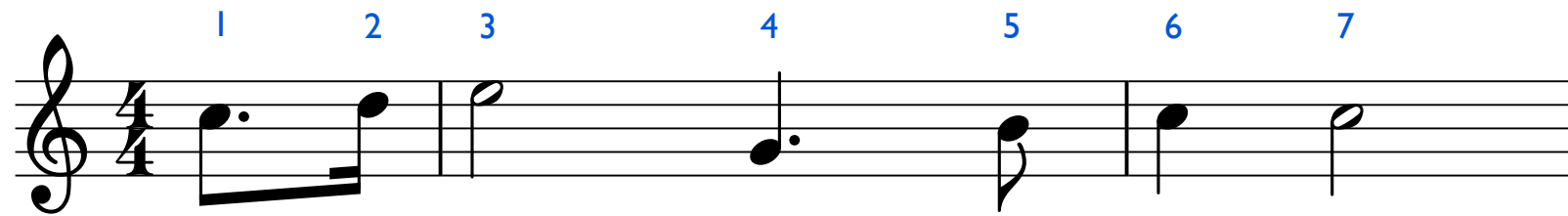
Synchronization Strategies



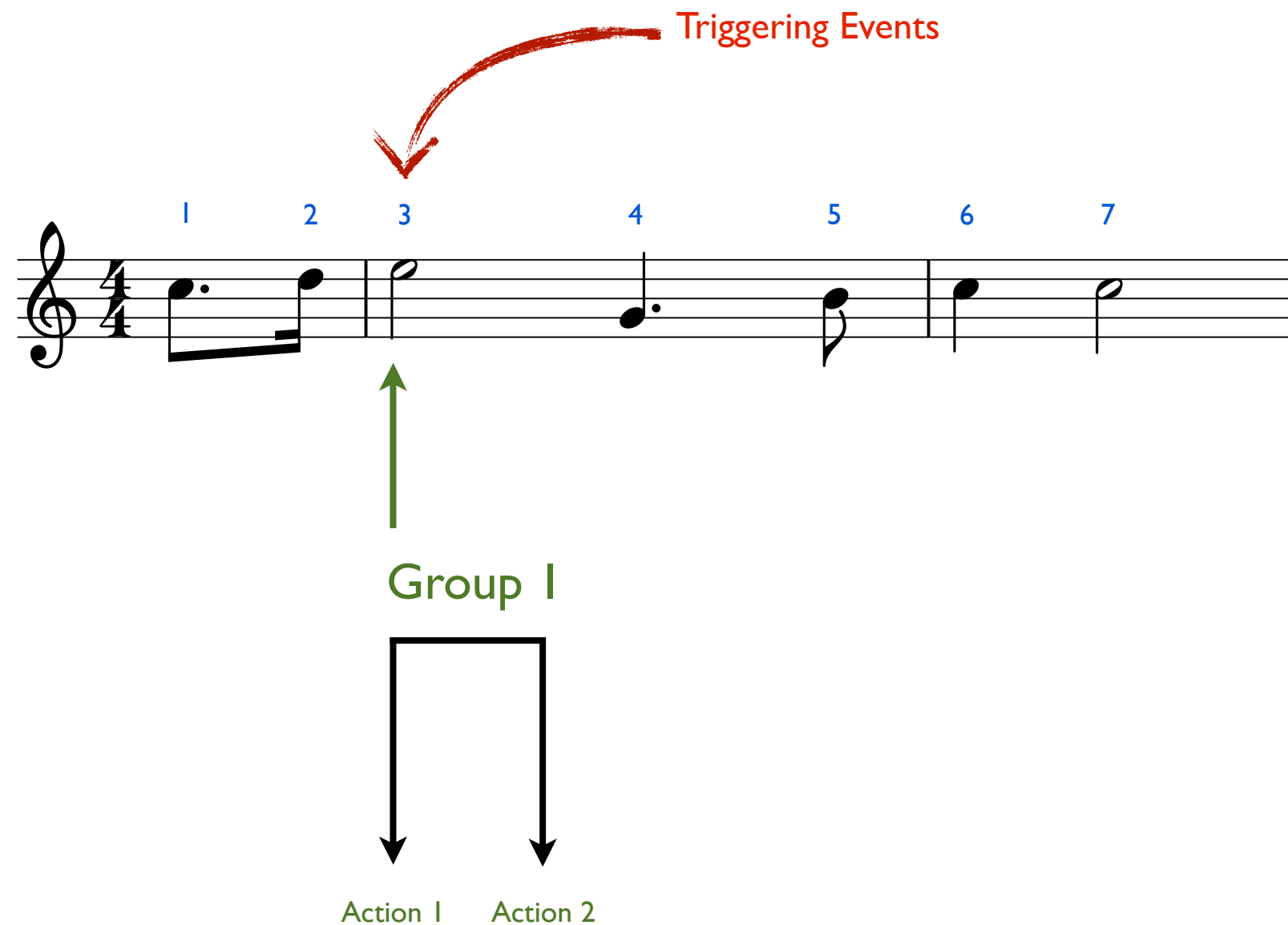
***Loose: Synchronization
with the tempo stream.***



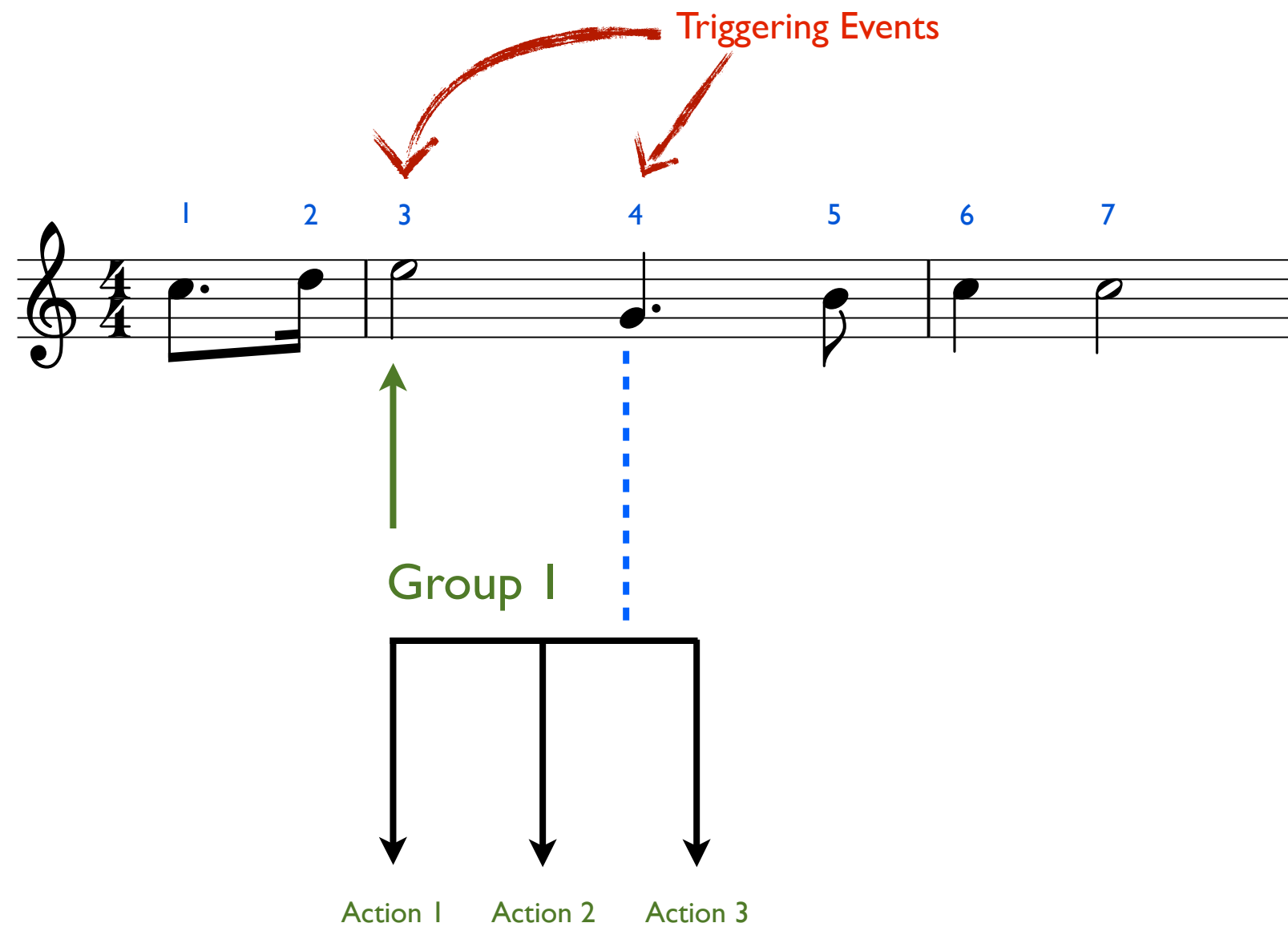
**Loose: Synchronization
with the tempo stream.**



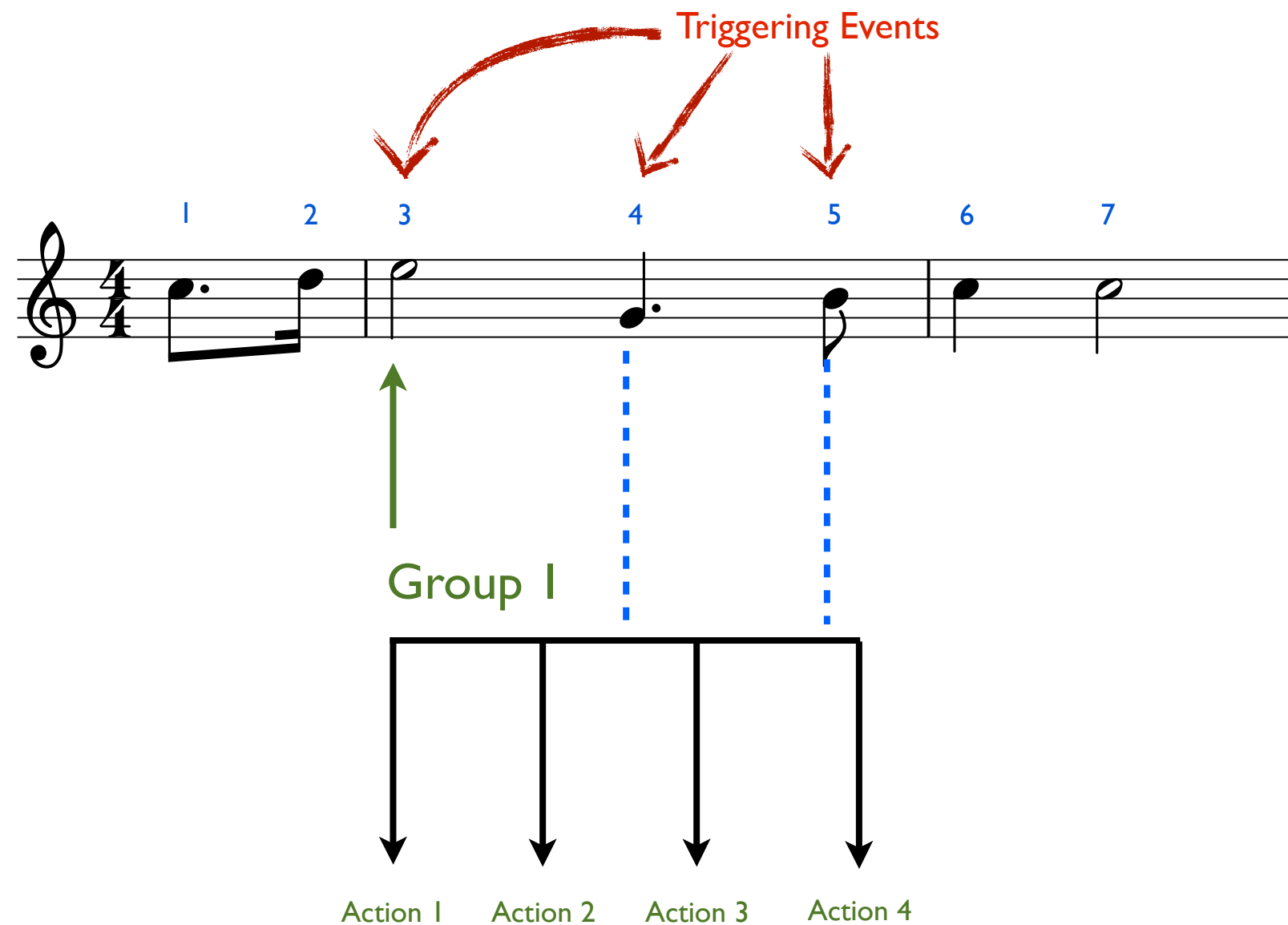
Tight: Synchronization
with tempo and events stream.



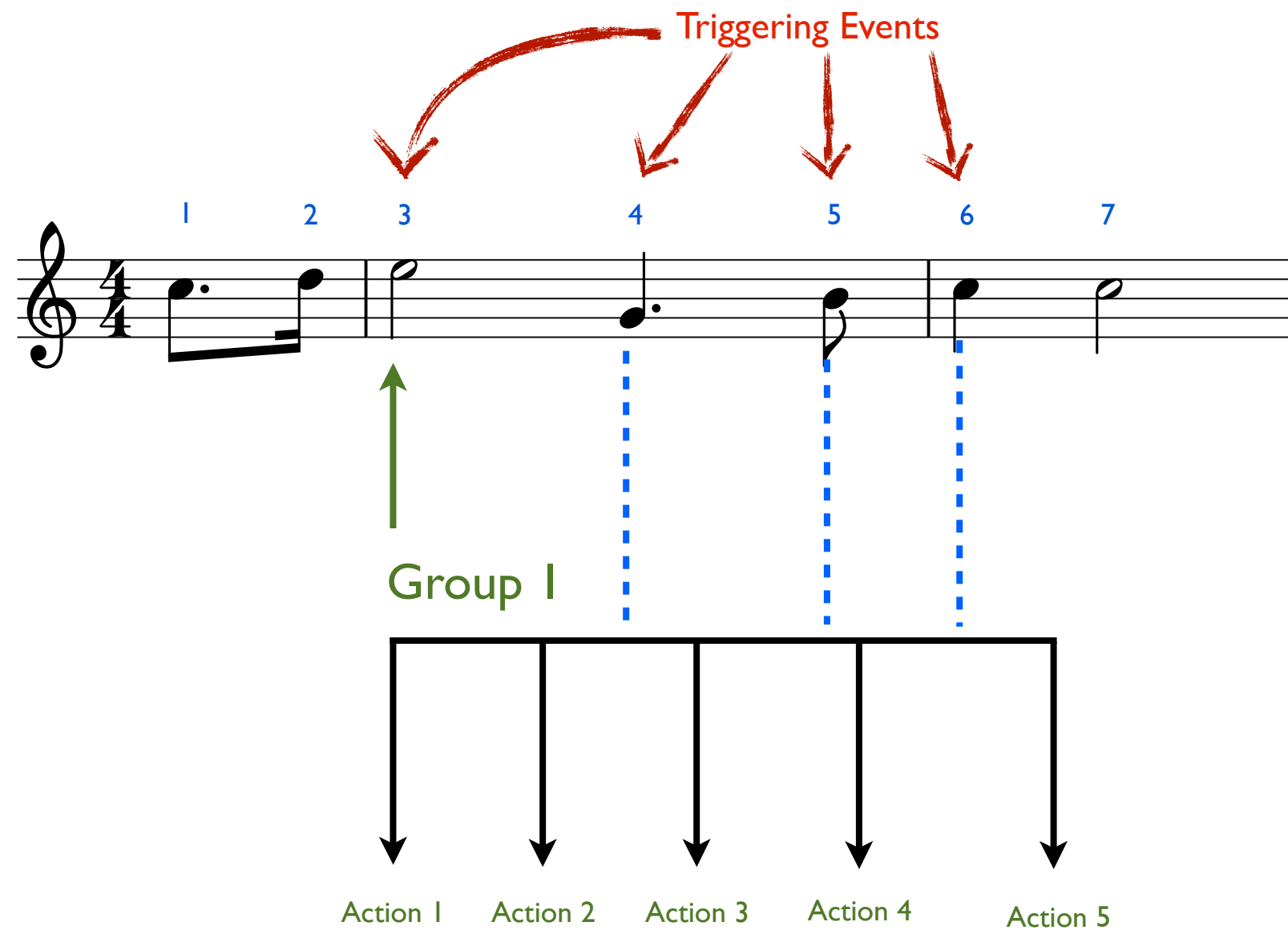
Tight: Synchronization
with tempo and events stream.



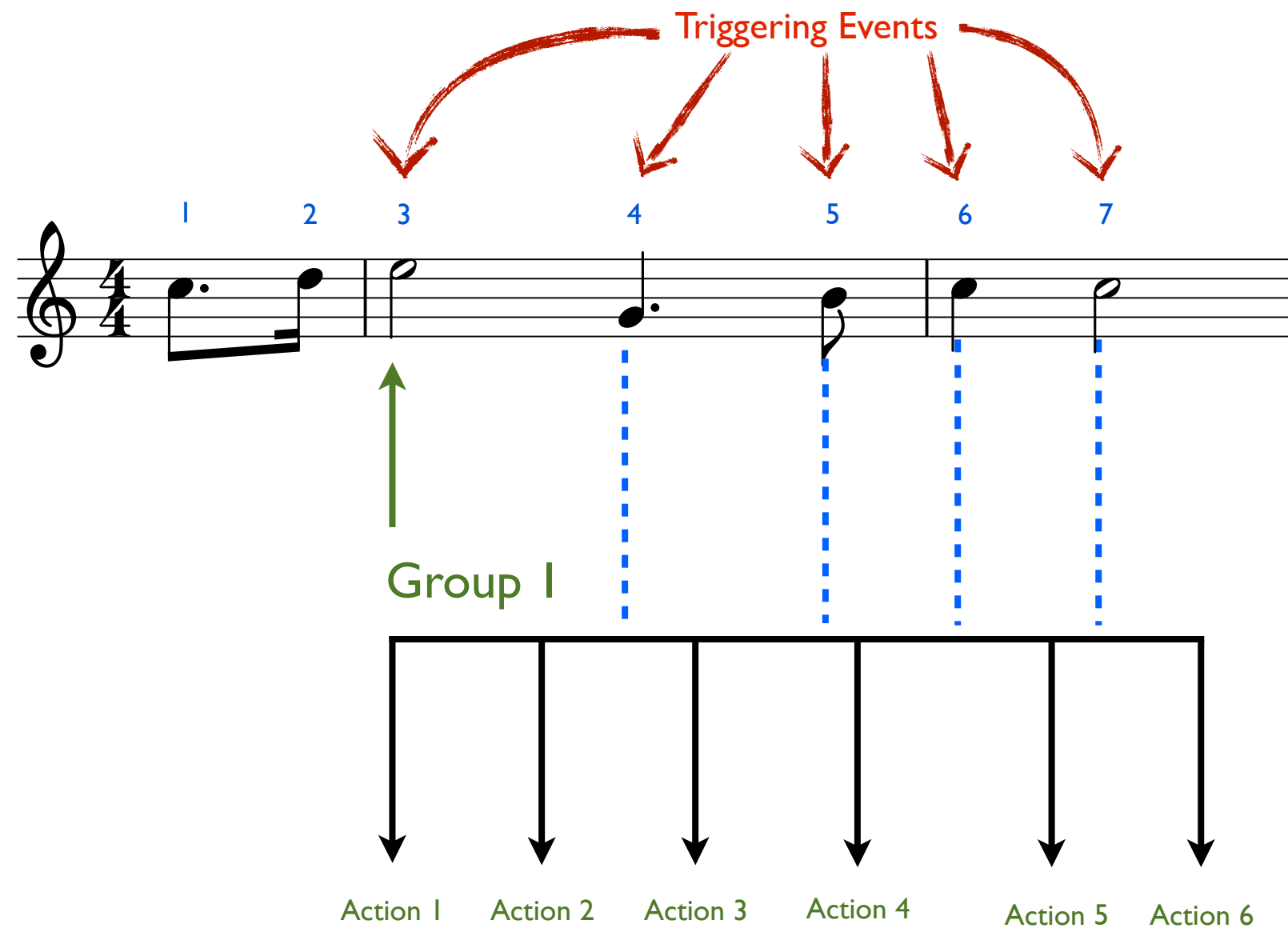
Tight: Synchronization
with tempo and events stream.



Tight: Synchronization
with tempo and events stream.

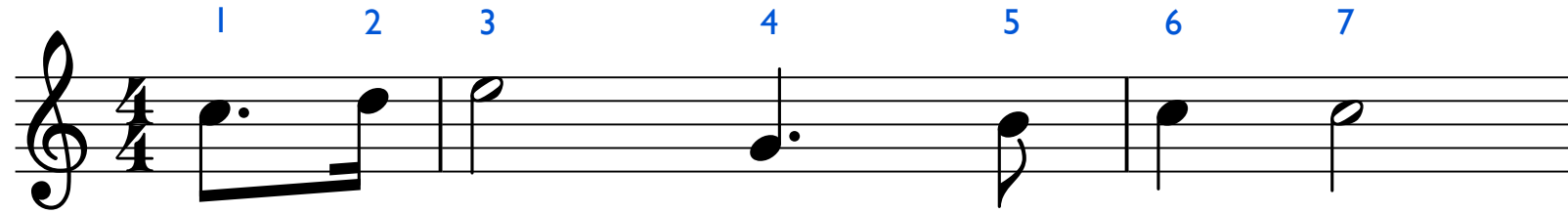


Tight: Synchronization
with tempo and events stream.

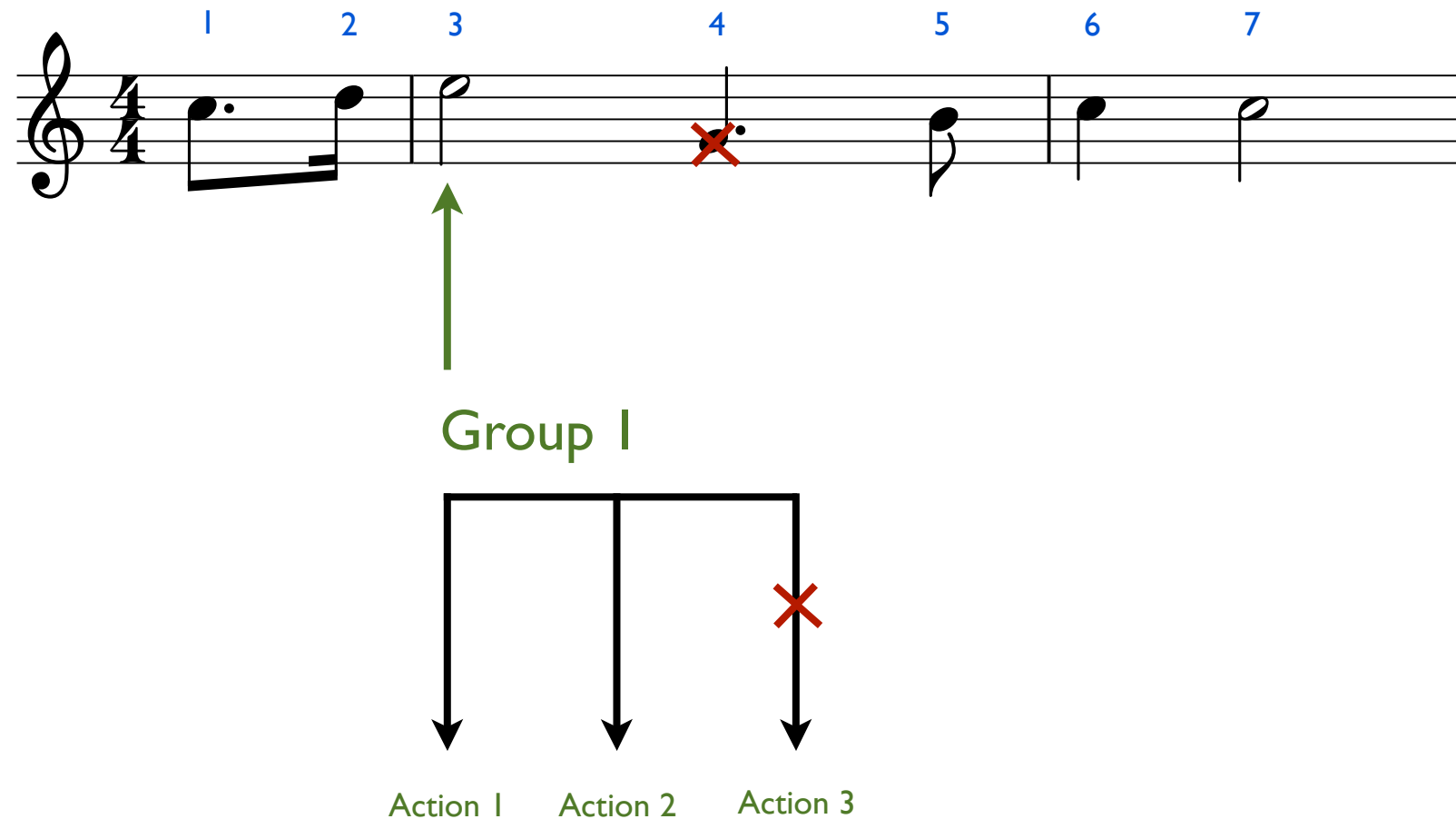


Tight: Synchronization
with tempo and events stream.

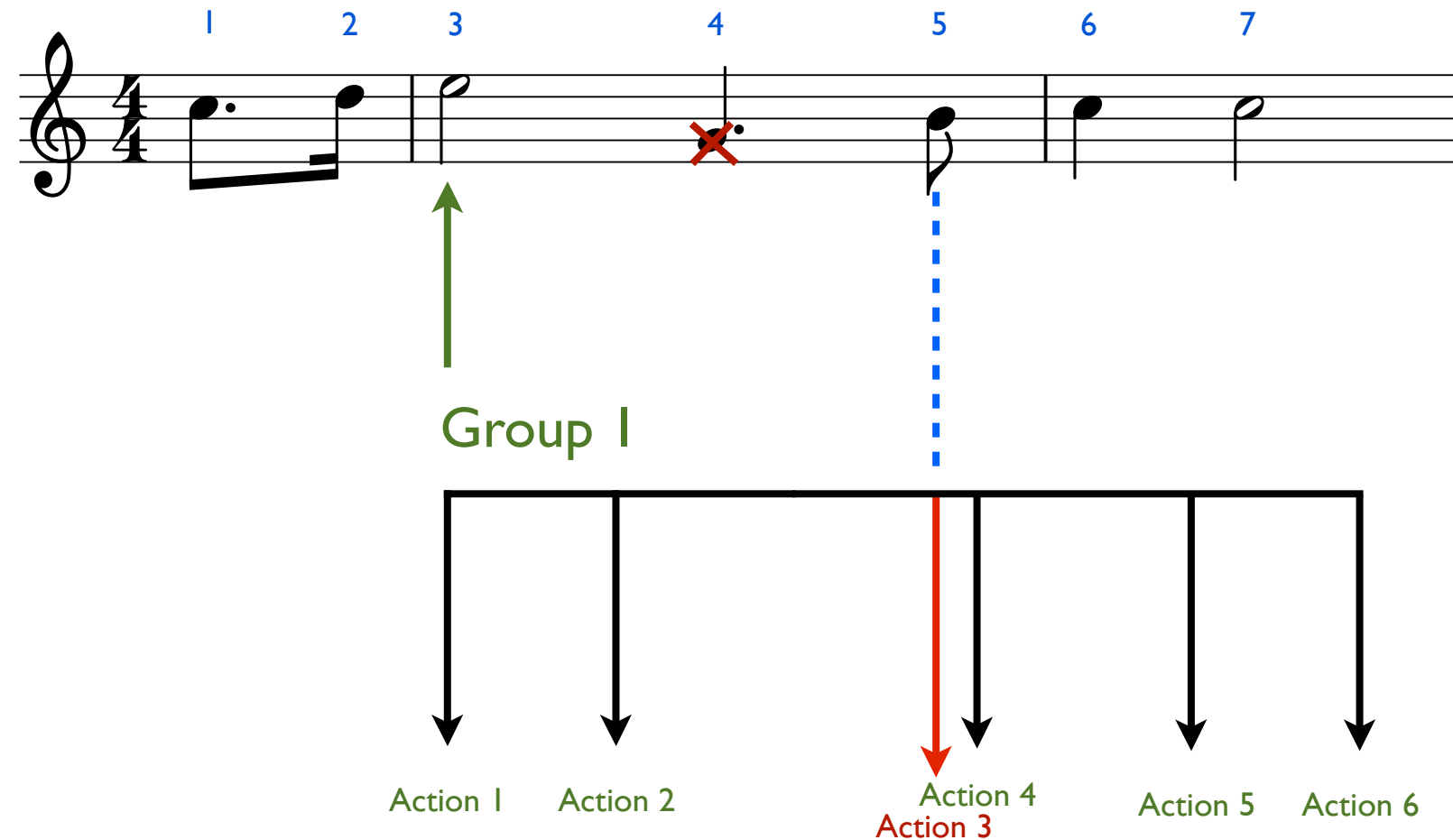
Error Handling Strategies



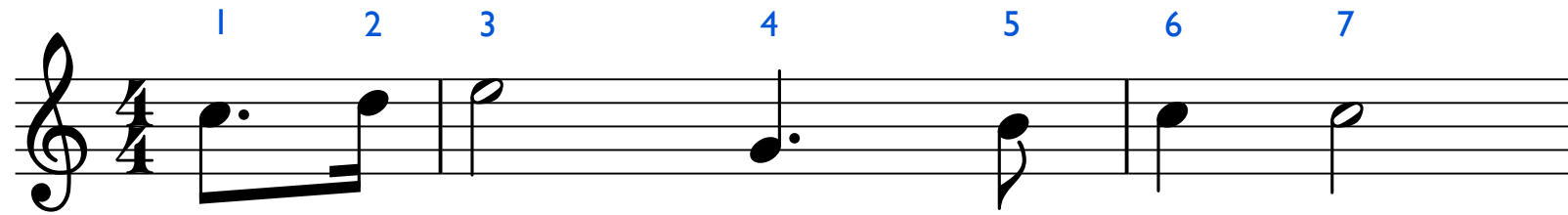
Causal: Actions should be launched immediately when the system recognizes the absence of the triggering event



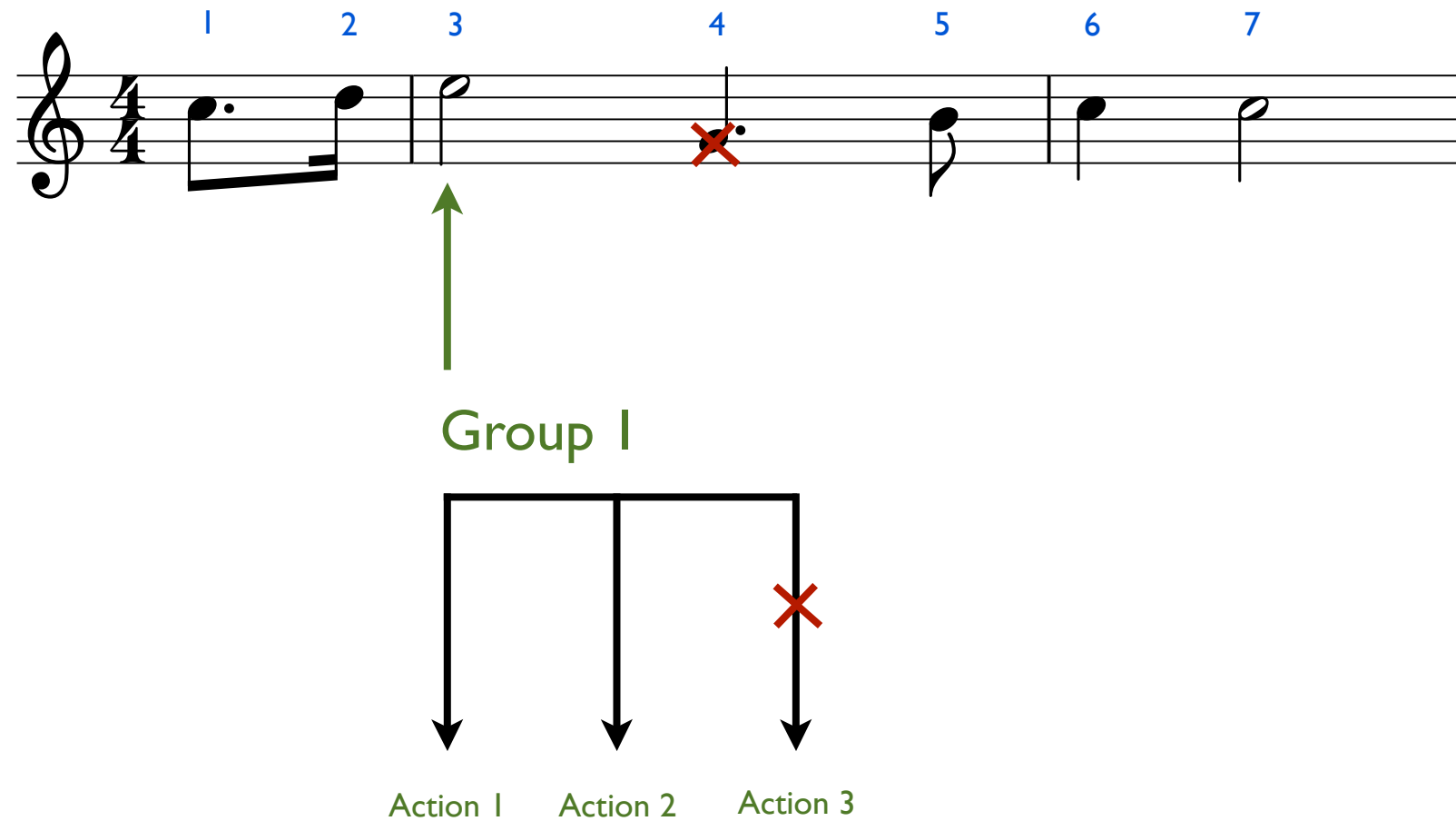
Causal: Actions should be launched immediately when the system recognizes the absence of the triggering event



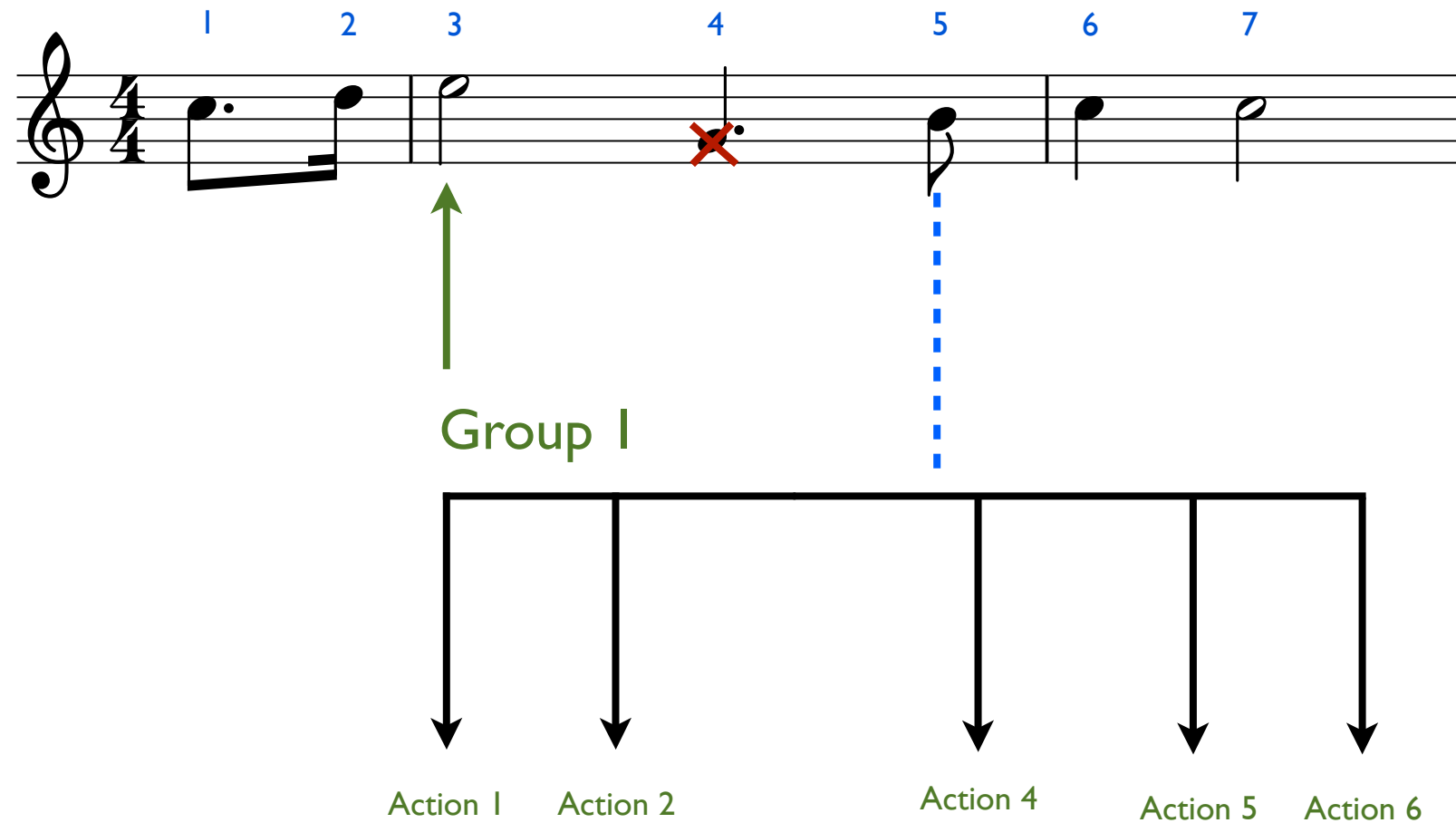
Causal: Actions should be launched immediately when the system recognizes the absence of the triggering event



Partial: Actions should be dismissed
in the absence of the triggering event



Partial: Actions should be dismissed in the absence of the triggering event



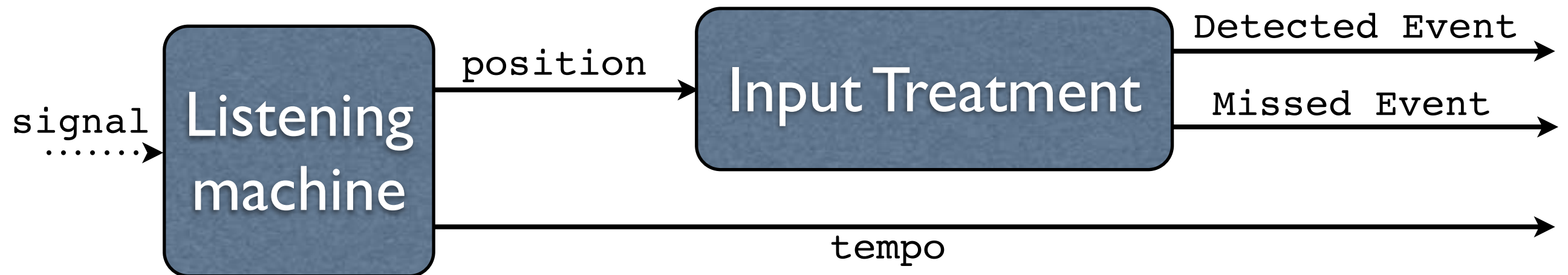
Partial: Actions should be dismissed in the absence of the triggering event

Language Characteristics

- A global logical time relative to the tempo
- Specify electronic actions with:
 - synchronization strategies
 - error handling strategies
- Composer friendly

Semantics

Detected and Missed Event

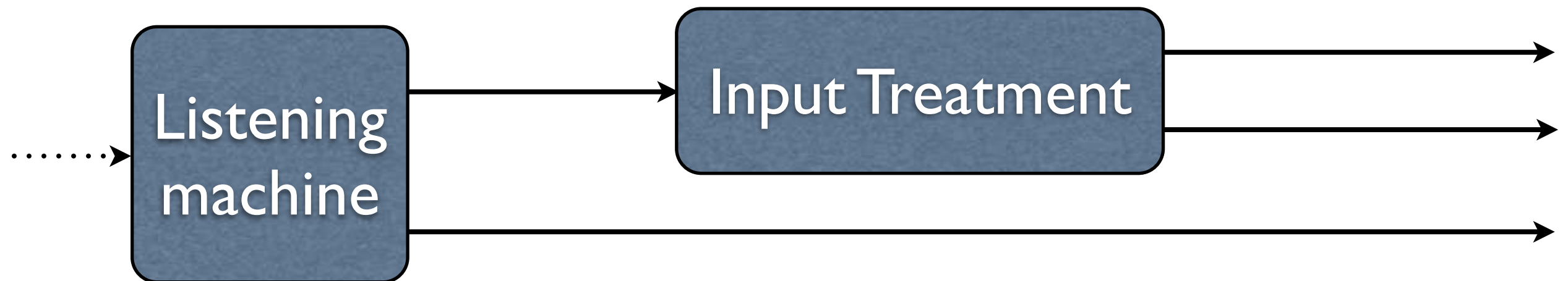


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

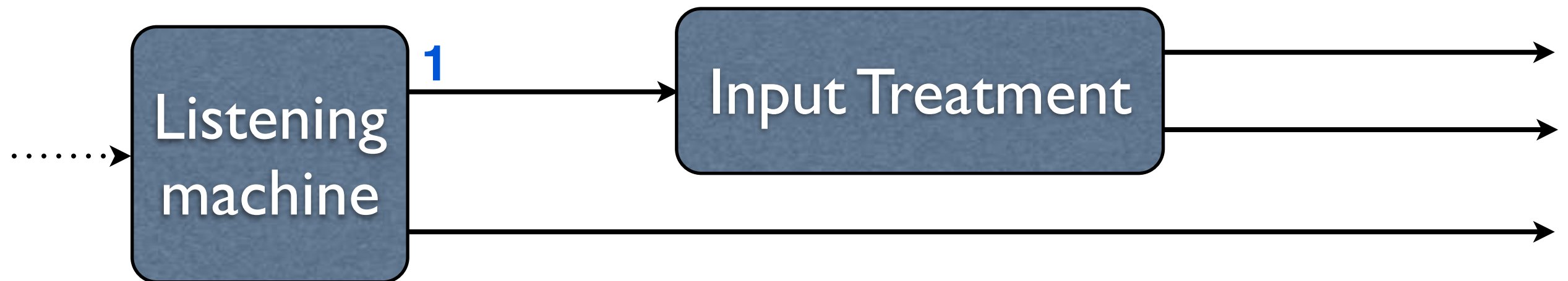


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

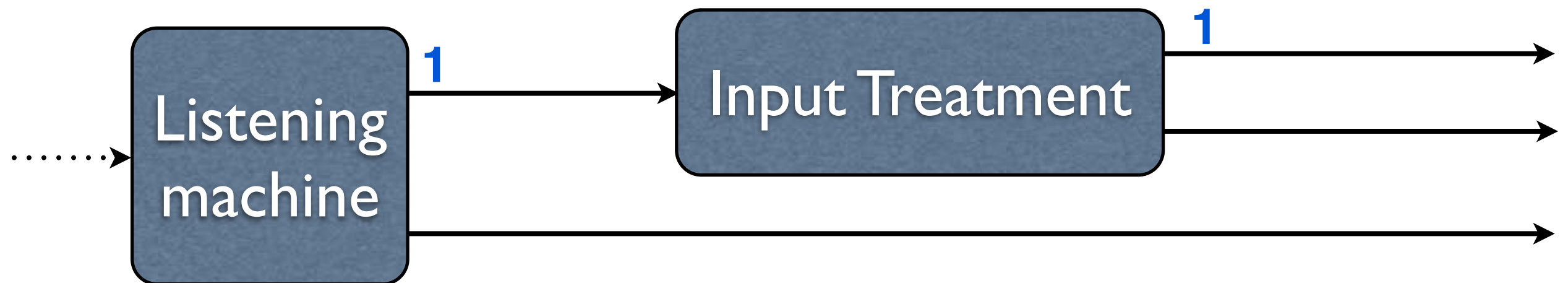


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

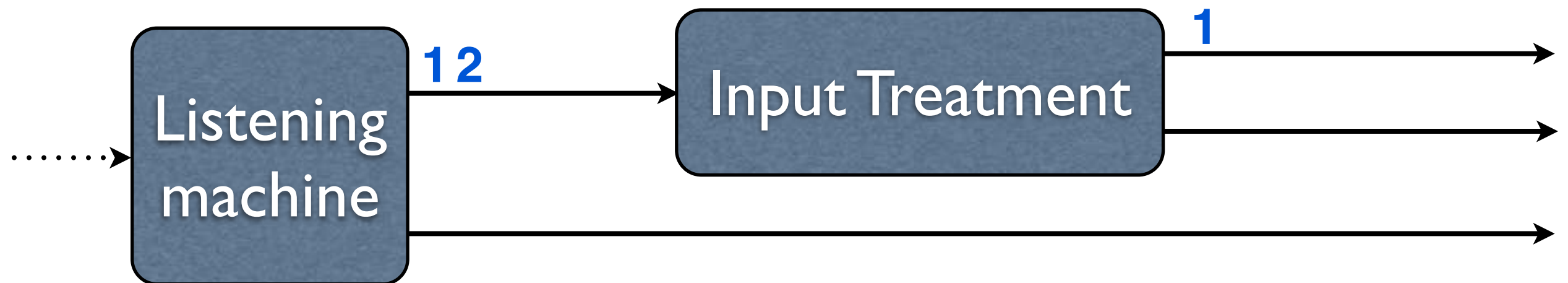


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

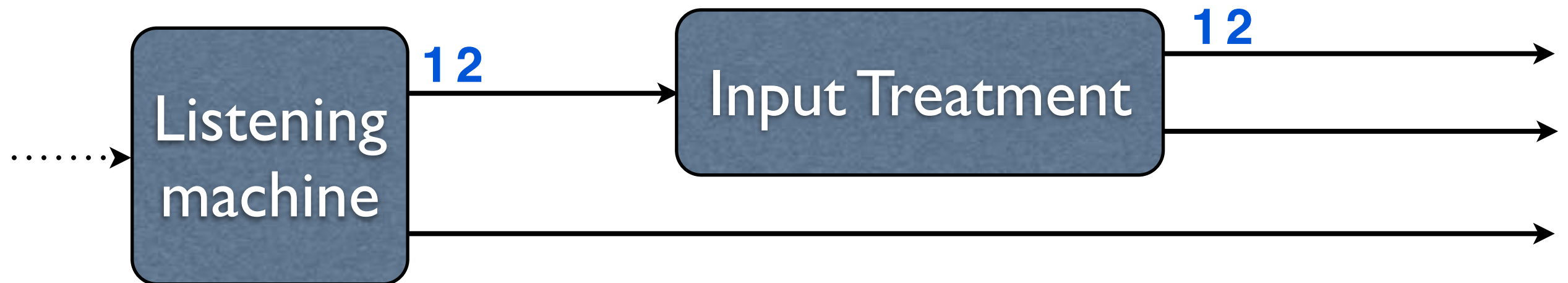


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

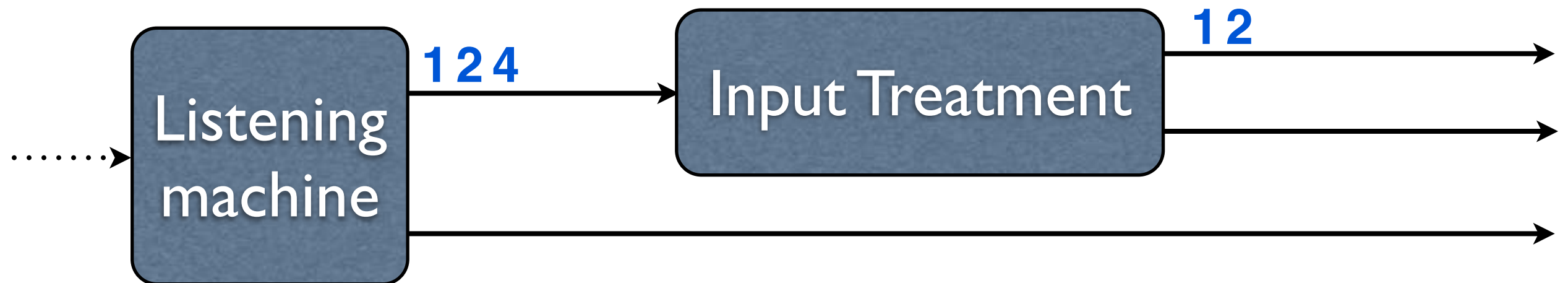


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

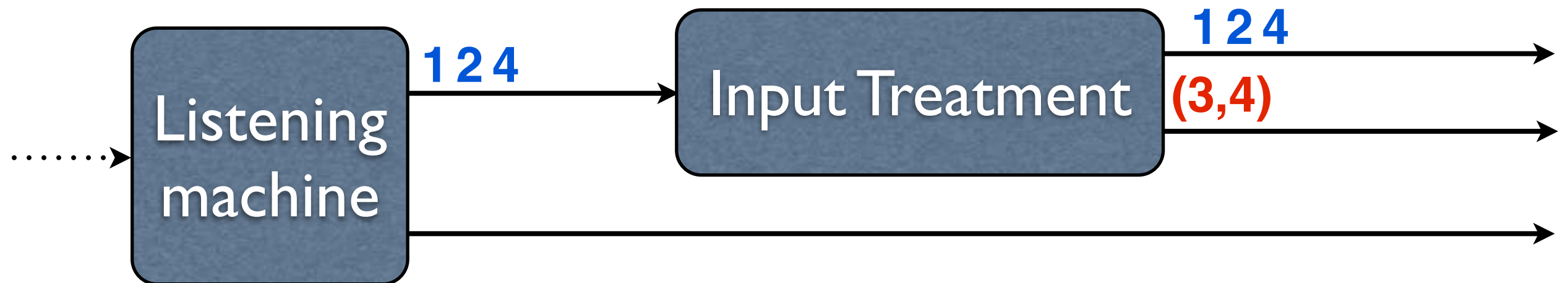


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

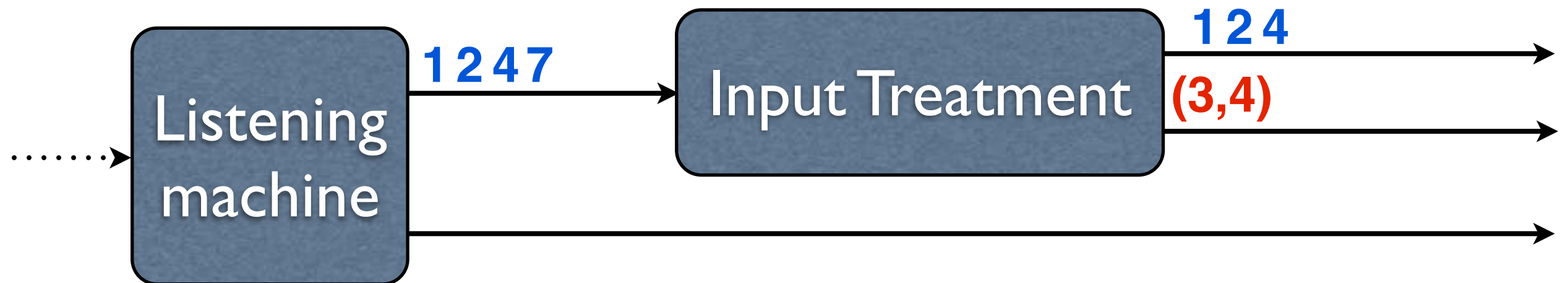


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event

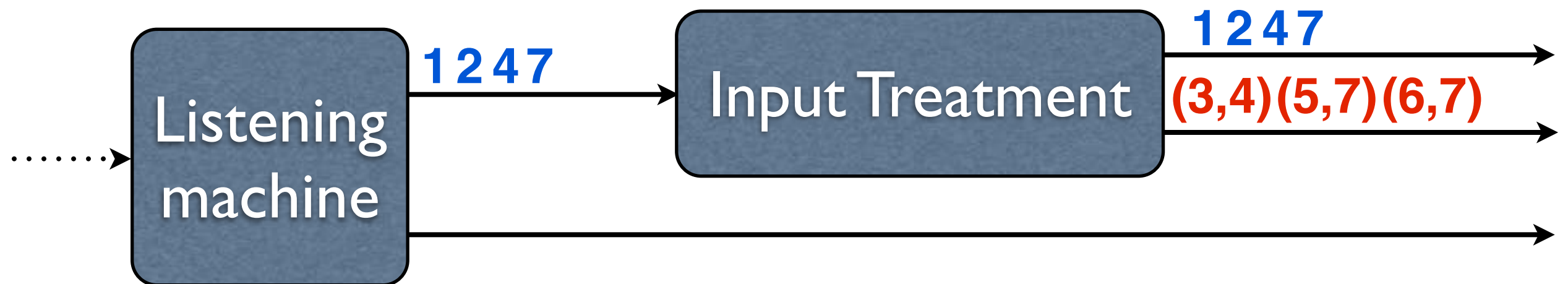


D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Detected and Missed Event



D : the set of detected instrumental events

For each missed event i we associate
the next detected event

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Formalization

<i>score</i>	$::=$	$\varepsilon \mid (event : seq) score$		
<i>event</i>	$::=$	$event\ i\ t$	$t \in \mathbb{Q}$	Duration
<i>seq</i>	$::=$	$\varepsilon \mid (\delta\ ae)\ seq$	$\delta \in \mathbb{Q}$	Delay
<i>ae</i>	$::=$	$action \mid group$	$i \in \mathbb{N}$	Label
<i>group</i>	$::=$	$group\ synchro\ error\ seq$	$a \in \mathcal{A}$	Action
<i>synchro</i>	$::=$	$tight \mid loose$		
<i>error</i>	$::=$	$local \mid global \mid partial \mid causal$		

A performance *perf* is a set of triplets (i, δ, a)
D is the set of detected instrumental events

Semantics

$$D \mid \frac{}{exec} score \Rightarrow perf$$

The Three Predicates

$$D \mid \frac{exec}{score} \Rightarrow perf$$

Execute a score

$$D, i, \delta \mid \frac{detected}{seq} \Rightarrow perf$$

Execute a sequence of actions
bound to a detected event i
with a delay δ

$$D, i, \delta \mid \frac{missed}{seq} \Rightarrow perf$$

Execute a sequence of actions
bound to a missed event i
with a delay δ

Execution of a score

$$\text{(Empty Score)} \quad \frac{}{D \mid \frac{exec}{\quad} \varepsilon \Rightarrow \emptyset}$$

$$\text{(Exec Score)} \quad \frac{D \mid \frac{exec}{\quad} (\text{event } i \ t : seq) \rightarrow p_1 \quad D \mid \frac{exec}{\quad} sc \Rightarrow p_2}{D \mid \frac{exec}{\quad} (\text{event } i \ t : seq) \ sc \Rightarrow p_1 \cup p_2}$$

Binding

$$\text{(Detect)} \quad \frac{i \in D \quad D, i, 0.0 \mid \frac{\text{detected}}{\quad} seq \Rightarrow p}{D \mid \frac{exec}{\quad} (\text{event } i \ t : seq) \rightarrow p}$$

$$\text{(Miss)} \quad \frac{i \notin D \quad D, i, 0.0 \mid \frac{\text{missed}}{\quad} seq \Rightarrow p}{D \mid \frac{exec}{\quad} (\text{event } i \ t : seq) \rightarrow p}$$

Execution: Atomic Actions

$$\text{(Detected Action)} \quad \frac{}{D, i, \delta \mid \frac{\text{detected}}{a \rightarrow (i, \delta, a)}}$$

$$\text{(Missed Action)} \quad \frac{\mathcal{M}(i) = j \quad \delta' = \max(0.0, \mathcal{E}(i) + \delta - \mathcal{E}(j))}{D, i, \delta \mid \frac{\text{missed}}{a \rightarrow (j, \delta', a)}}$$

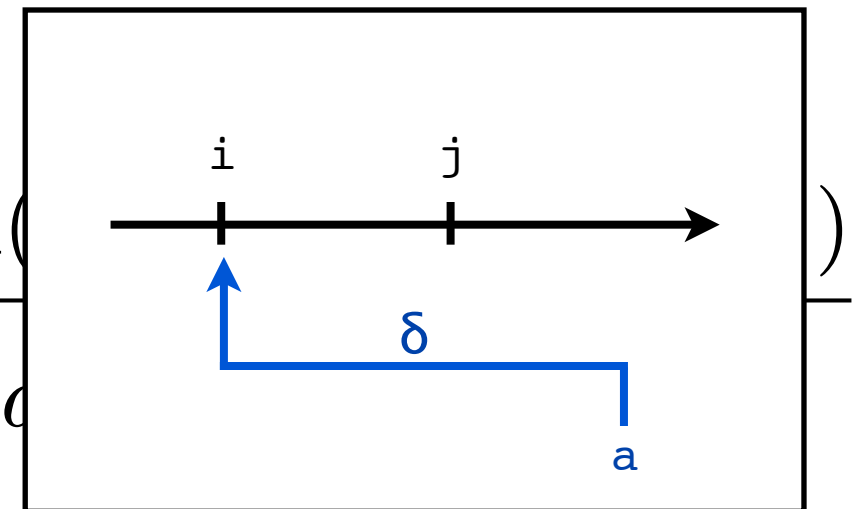
$\mathcal{E}(i)$: date of event i

$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$
Error detection: i is missed
 j is the first detection after i

Execution: Atomic Actions

$$\text{(Detected Action)} \quad \frac{}{D, i, \delta \mid \frac{\text{detected}}{a \rightarrow (i, \delta, a)}}$$

$$\text{(Missed Action)} \quad \frac{\mathcal{M}(i) = j \quad \delta' = \max(\delta, \mathcal{E}(j) - \mathcal{E}(i))}{D, i, \delta \mid \frac{\text{missed}}{a \rightarrow (j, \delta', a)}}$$



$\mathcal{E}(i)$: date of event i

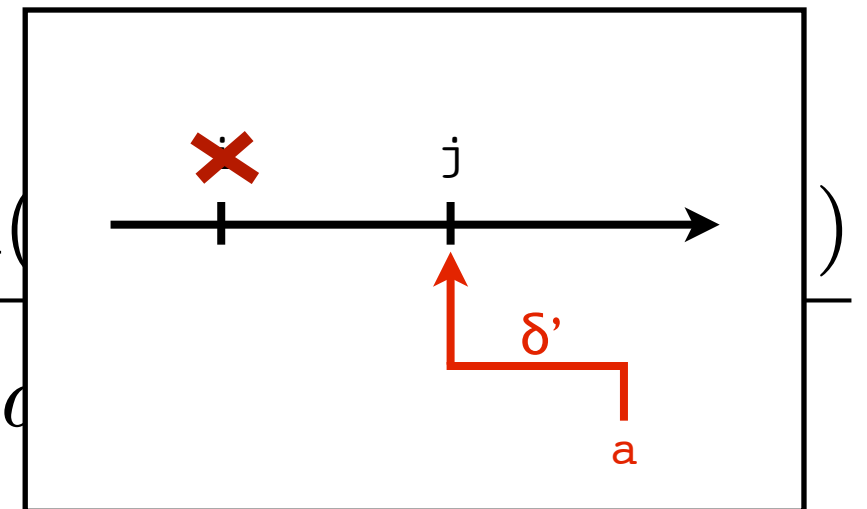
$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Error detection: i is missed
 j is the first detection after i

Execution: Atomic Actions

$$\text{(Detected Action)} \quad \frac{}{D, i, \delta \mid \frac{\text{detected}}{a \rightarrow (i, \delta, a)}}$$

$$\text{(Missed Action)} \quad \frac{\mathcal{M}(i) = j \quad \delta' = \max(\dots)}{D, i, \delta \mid \frac{\text{missed}}{a \rightarrow (i, \delta, a)}}$$



$\mathcal{E}(i)$: date of event i

$$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$$

Error detection: i is missed
 j is the first detection after i

Execution: Atomic Actions

$$\text{(Detected Action)} \quad \frac{}{D, i, \delta \mid \frac{\text{detected}}{\quad} a \rightarrow (i, \delta, a)}$$

$$\text{(Missed Action)} \quad \frac{\mathcal{M}(i) = j \quad \delta' = \max(0.0, \mathcal{E}(i) + \delta - \mathcal{E}(j))}{D, i, \delta \mid \frac{\text{missed}}{\quad} a \rightarrow (j, \delta', a)}$$

$\mathcal{E}(i)$: date of event i

$\mathcal{M}(i) = \min\{j \in D \mid j > i\}$
 Error detection: i is missed
 j is the first detection after i

Implementation

Synchronous Embedding

- **Several experiments**
Heptagon, Lucid Synchrone, ReactiveML
- **Why ReactiveML?**
 - **Functional, typed language, on top of OCaml**
recursion and higher order processes
 - **Dynamic features**
difficult to get with Lustre/Esterel/...
new interactions, live coding

ReactiveML

OCaml extended with synchronous features à la Esterel

[Mandel-Pouzet 2005]

Process

```
let process <id> {<pattern>} = <expr>
```

State machines, executed through several instants.

Simple OCaml functions are considered to be instantaneous.

Basics

Synchronization: `pause`

Execution: `run <expr>`

Composition

Sequence: `<expr> ; <expr>`

Parallelism: `<expr> || <expr>`

Signals

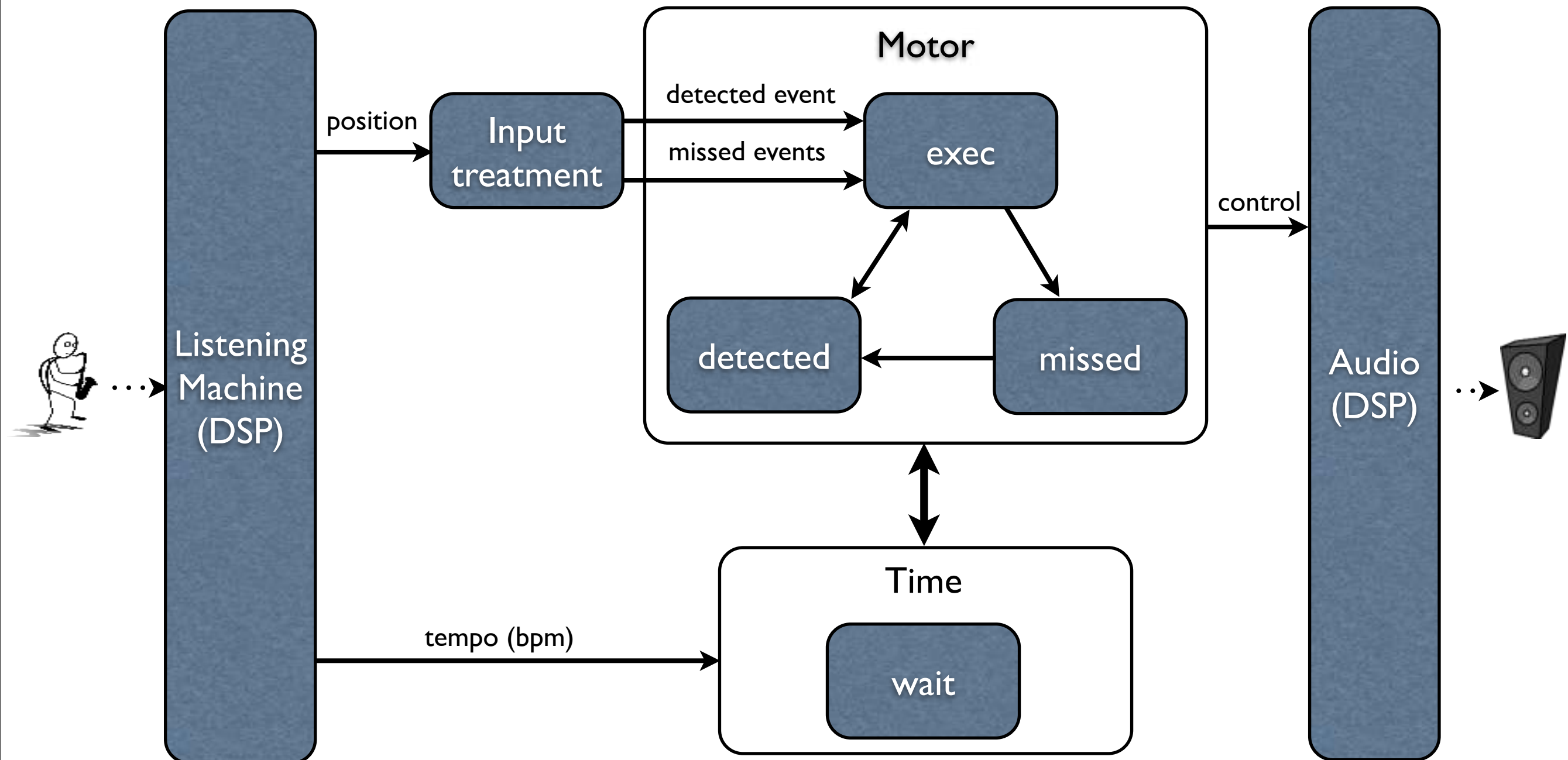
Definition: `signal <id>`

Emission: `emit <id>`

Waiting: `await <id>`

*Broadcast communication
between processes*

Architecture



Execution of a score

$$\text{(Empty Score)} \quad \frac{}{D \mid \frac{exec}{\quad} \varepsilon \Rightarrow \emptyset}$$

$$\text{(Exec Score)} \quad \frac{D \mid \frac{exec}{\quad} (\text{event } i \ t : seq) \rightarrow p_1 \quad D \mid \frac{exec}{\quad} sc \Rightarrow p_2}{D \mid \frac{exec}{\quad} (\text{event } i \ t : seq) \ sc \Rightarrow p_1 \cup p_2}$$

Execution of a score

$$\begin{array}{c} \text{(Empty Score)} \quad \frac{}{D \mid \frac{exec}{\varepsilon} \Rightarrow \emptyset} \qquad \text{(Exec Score)} \quad \frac{D \mid \frac{exec}{(\text{event } i \ t : seq) \rightarrow p_1} \quad D \mid \frac{exec}{sc \Rightarrow p_2}}{D \mid \frac{exec}{(\text{event } i \ t : seq) \ sc \Rightarrow p_1 \cup p_2}} \end{array}$$

```
let rec process exec score =  
  match score with  
  | [] -> (* rule (Empty Score) *) ()  
  | se::sc ->  
    (* rule (Exec Score) *)  
    run (exec_score_event se) ||  
    run (exec sc)
```

Execution of a score

$$\begin{array}{c}
 \text{(Empty Score)} \quad \frac{}{D \mid \xrightarrow{exec} \varepsilon \Rightarrow \emptyset} \\
 \text{(Exec Score)} \quad \frac{D \mid \xrightarrow{exec} (\text{event } i \ t : seq) \rightarrow p_1 \quad D \mid \xrightarrow{exec} sc \Rightarrow p_2}{D \mid \xrightarrow{exec} (\text{event } i \ t : seq) \ sc \Rightarrow p_1 \cup p_2}
 \end{array}$$

```
let rec process exec score =
```

```
  match score with
```

```
  | [] -> (* rule (Empty Score) *) ()
```

```
  | se::sc ->
```

```
    (* rule (Exec Score) *)
```

```
    run (exec_score_event se) ||
```

```
    run (exec sc)
```

parallel
composition



Binding

$$\text{(Miss)} \quad \frac{i \notin D \quad D, i, 0.0 \mid \text{missed} \quad seq \Rightarrow p}{D \mid \text{exec} \quad (\text{event } i \text{ } t : seq) \text{ } sc \rightarrow p}$$

$$\text{(Detect)} \quad \frac{i \in D \quad D, i, 0.0 \mid \text{detected} \quad seq \Rightarrow p}{D \mid \text{exec} \quad (\text{event } i \text{ } t : seq) \rightarrow p}$$

```
let rec process exec_score_event se =  
  let status = run (wait_event se.event) in  
  match status with  
  | Detected ->  
    (* rule (Detect) *)  
    run (exec_seq (detected i) 0.0 se.seq)  
  | Missed(j) ->  
    (* rule (Miss) *)  
    run (exec_seq (missed i j) 0.0 se.seq)
```

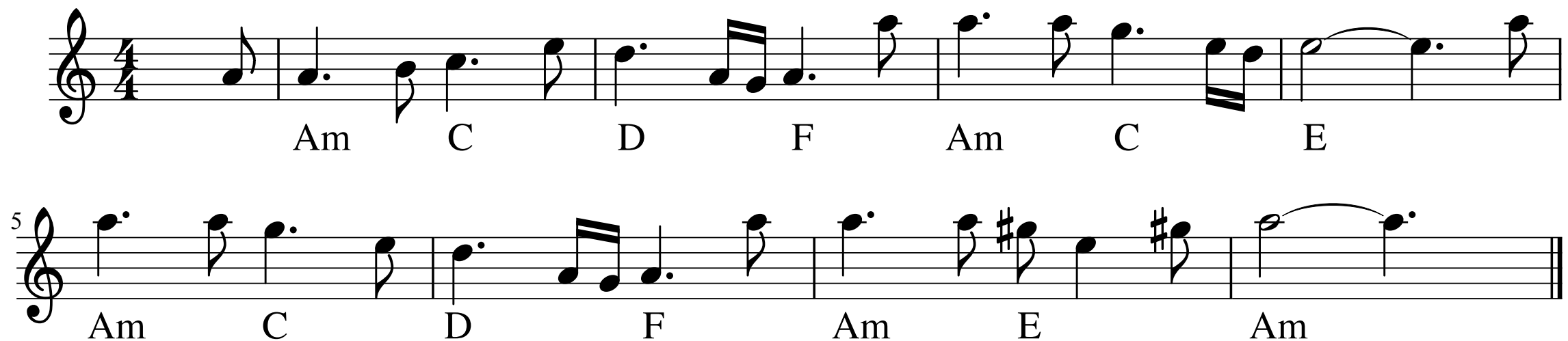
Applications

Live Coding

Modify, correct and interact with the score
during the performance

Automatic Accompaniment

The house of the rising sun



- **Functional programming**
modular definition of the accompaniment
- **Reactive programming**
interaction with the score during the performance

Definitions

1. Define the bass line

```
let bass = [0.0, (A, Min); 2.0, (C, Maj); ...]  
val bass: (delay * chord) list
```

2. Define the accompaniment style

```
let arpeggio chord =  
  ...  
  group Loose Local  
    [0.0, action_note (fond);  
     1.0, action_note (third);  
     2.0, action_note (fifth);}]  
val arpeggio: chord -> asco_event
```

3. Link with the performance

```
let process basic_accomp =  
  run (link asco 2 roots)  
val basic_accomp: unit process
```


Interactions

- **Kill a process when a signal is emitted**
allow to modify the accompaniment
- **Suspend the execution of a process**
pause and resume a process with a signal
- **Dynamically change the behavior of a process**
switch between different kinds of accompaniment

Kill a Process

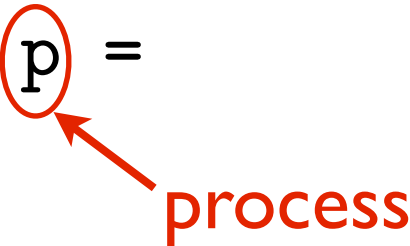
Example of a higher-order process

```
let process killable k p =  
  do  
    run p  
  until k done  
val killable:  
  (unit, unit) event -> unit process ->  
  unit process
```

Kill a Process

Example of a higher-order process

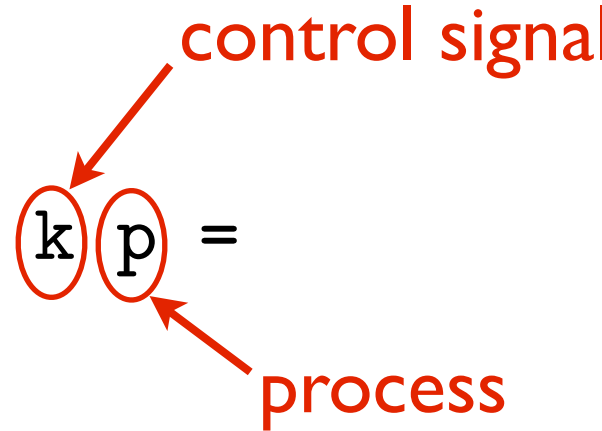
```
let process killable k (p) =  
  do  
    run p  
  until k done  
val killable:  
  (unit, unit) event -> unit process ->  
  unit process
```



Kill a Process

Example of a higher-order process

```
let process killable (k) (p) =  
  do  
    run p  
  until k done  
val killable:  
  (unit, unit) event -> unit process ->  
  unit process
```



control signal

process


Dynamic Changes

Example of a recursive higher-order process

```
let process rec replaceable replace p =  
  do  
    run p  
  until replace (q) ->  
    run (replaceable replace q)  
done  
val replaceable:  
  (unit process, unit process) event ->  
  unit process -> unit process
```

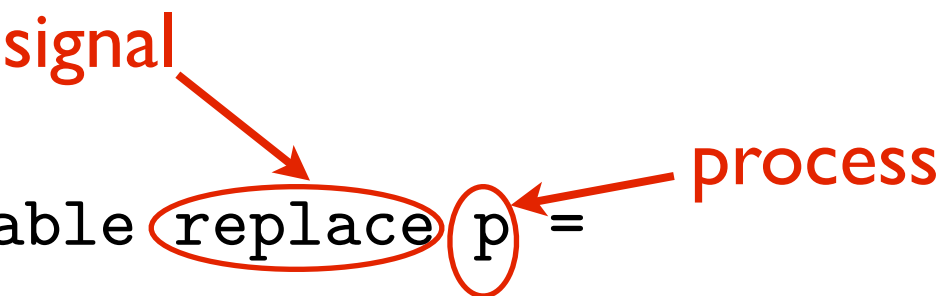
Dynamic Changes

Example of a recursive higher-order process

```
let process rec replaceable replace (p) = 
do
  run p
until replace (q) ->
  run (replaceable replace q)
done
val replaceable:
  (unit process, unit process) event ->
    unit process -> unit process
```

Dynamic Changes

Example of a recursive higher-order process



```
let process rec replaceable replace p =  
  do  
    run p  
  until replace (q) ->  
    run (replaceable replace q)  
done  
val replaceable:  
  (unit process, unit process) event ->  
  unit process -> unit process
```

Dynamic Changes

Example of a recursive higher-order process

```
let process rec replaceable replace p =  
  do  
    run p  
  until replace (q) ->  
    run (replaceable replace q)  
done  
val replaceable:  
  (unit process, unit process) event ->  
  unit process -> unit process
```

signal

process

new behavior

signal can carry processes

New Reactive Behaviors

Example: Steve Reich's Piano Phase

Piano Phase ...

Piano Phase,

pour 2 pianos ou 2 marimbas

Steve Reich

$\text{♩} = \text{ca. } 72$

Bob

Alice

The musical score for 'Piano Phase' by Steve Reich is presented for two parts: Bob (top staff) and Alice (bottom staff). The score consists of six measures, each with a specific duration in parentheses above the staff.

- Measure 1:** Bob (x4 - 8), Alice (mf non legato). Bob's part is marked 'r.h.' and 'l.h.'. Alice's part is marked 'mf' and 'non legato'.
- Measure 2:** Bob (x12 - 18), Alice (fade in, non legato). Bob's part is marked 'r.h.' and 'l.h.'. Alice's part is marked 'mf' and 'non legato'.
- Measure 3:** Bob (x4 - 16), Alice (accel very slightly, hold tempo 1). Bob's part is marked 'hold tempo 1'. Alice's part is marked 'a.v.s.' and 'hold tempo 1'.
- Measure 4:** Bob (x16 - 24), Alice (hold tempo 1). Bob's part is marked '(tempo 1)'. Alice's part is marked 'a.v.s.' and 'hold tempo 1'.
- Measure 5:** Bob (x16 - 24), Alice (hold tempo 1). Bob's part is marked '(tempo 1)'. Alice's part is marked 'a.v.s.' and 'hold tempo 1'.
- Measure 6:** Bob (x4 - 16), Alice (hold tempo 1). Bob's part is marked '(tempo 1)'. Alice's part is marked 'a.v.s.' and 'hold tempo 1'.

The score includes various performance instructions such as 'mf', 'non legato', 'fade in', 'accel very slightly', 'hold tempo 1', and 'a.v.s.' (ad libitum).

Piano Phase ...

Bob

Alice

Piano Phase,
pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

1 (x4-8) 2 (x12-18) 3 (x16-24) 4 (x16-24) 5 (x16-24) 6 (x16-24)

r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h.

mf non legato *mf* non legato *mf* non legato *mf* non legato *mf* non legato *mf* non legato

fade in non legato *mf* *mf* *mf* *mf* *mf* *mf*

hold tempo 1 hold tempo 1 hold tempo 1 hold tempo 1 hold tempo 1 hold tempo 1

accel very slightly a.v.s. a.v.s. a.v.s. a.v.s. a.v.s. a.v.s.

(tempo 1) (tempo 1) (tempo 1) (tempo 1) (tempo 1) (tempo 1)

Synchronization

Piano Phase ...

Bob

Alice

Piano Phase,
pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

1 (x4-8) 2 (x12-18) 3 (x4-16) 3 (x16-24) (x4-16)

r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h.

mf non legato mf non legato mf non legato mf non legato

fade in non legato mf hold tempo 1 accel very slightly hold tempo 1 a.v.s.

4 (x16-24) 5 (x16-24) 6 (x16-24)

(x4-16) (tempo 1) (tempo 1) (tempo 1)

hold tempo 1 a.v.s. hold tempo 1 a.v.s. hold tempo 1 a.v.s.

The image shows a musical score for 'Piano Phase' by Steve Reich. It is written for two pianos or two marimbas. The tempo is marked as approximately 72 beats per minute. The score is divided into six measures, each with a specific rhythmic pattern and duration. The first measure is marked '1 (x4-8)' and the second '2 (x12-18)'. The third measure is marked '3 (x4-16)' and '3 (x16-24)'. The fourth measure is marked '4 (x16-24)' and '5 (x16-24)'. The fifth measure is marked '6 (x16-24)' and '6 (x16-24)'. The sixth measure is marked '6 (x16-24)' and '6 (x16-24)'. The score includes various performance instructions such as 'mf non legato', 'fade in', 'hold tempo 1', 'accel very slightly', and 'a.v.s.'. A red box highlights a specific section of the score, indicating a tempo change or a specific performance instruction.

Desynchronization

Piano Phase ...

Bob

Alice

Piano Phase,
pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

1 (x4-8) r.h.
l.h. *mf* non legato

2 (x12-18) r.h.
l.h. fade in non legato

4 (x16-24) (tempo 1) a.v.s. hold tempo 1

5 (x16-24) (tempo 1) a.v.s. hold tempo 1

Steve Reich

(x4-16)
hold tempo 1

accel
very slightiy

(x4-16) (tempo 1) a.v.s.

24 (tempo 1) a.v.s.

hold tempo 1

Desynchronization

Piano Phase ...

Bob

Alice

Piano Phase,
pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

Steve Reich

The musical score for 'Piano Phase' by Steve Reich is presented for two parts: Bob (top staff) and Alice (bottom staff). The score consists of six measures, each with a specific duration in parentheses above the staff: 1 (x4-8), 2 (x12-18), 3 (x16-24), 4 (x16-24), 5 (x16-24), and 6 (x16-24). The tempo is marked as 'ca. 72' (approximately 72 beats per minute). The score includes various performance instructions such as 'mf non legato', 'fade in', 'hold tempo 1', 'accel very slightly', and 'a.v.s.'. A red arrow points to the start of measure 3 in the top staff, which is circled in red.

1 (x4-8) 2 (x12-18) 3 (x16-24) 4 (x16-24) 5 (x16-24) 6 (x16-24)

r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h.

mf non legato mf non legato mf non legato mf non legato mf non legato mf non legato

fade in hold tempo 1 accel very slightly hold tempo 1 a.v.s. hold tempo 1 a.v.s. hold tempo 1 a.v.s. hold tempo 1 a.v.s. hold tempo 1 a.v.s.

Piano Phase ...

Bob

Alice

Piano Phase,
pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

1 (x4-8) 2 (x12-18) (x4-16) 3 (x16-24) (x4-16)

r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h.

mf non legato *mf* non legato *mf* non legato *mf* non legato

fade in *mf* *mf* *mf* *mf*

hold tempo 1 hold tempo 1 hold tempo 1 hold tempo 1

accel very slightly

(tempo 1) (tempo 1) (tempo 1) (tempo 1)

a.v.s. a.v.s. a.v.s. a.v.s.

4 (x16-24) 5 (x16-24) 6 (x16-24)

(x4-16) (x4-16) (x4-16) (x4-16)

(tempo 1) (tempo 1) (tempo 1) (tempo 1)

hold tempo 1 hold tempo 1 hold tempo 1 hold tempo 1

a.v.s. a.v.s. a.v.s. a.v.s.

Piano Phase ...

Bob

Alice

Piano Phase,

pour 2 pianos ou 2 marimbas

Steve Reich

$\text{♩} = \text{ca. } 72$

The musical score for 'Piano Phase' by Steve Reich is presented for two parts: Bob (top staff) and Alice (bottom staff). The score consists of six measures, each with a measure number and a range in parentheses: 1 (x4-8), 2 (x12-18), 3 (x16-24), 4 (x16-24), 5 (x16-24), and 6 (x16-24). The tempo is marked as 'ca. 72' (quarter note). The score includes various performance instructions such as 'mf non legato', 'fade in', 'non legato', 'mf', 'hold tempo 1', 'accel very slightly', and 'a.v.s.'. A red circle highlights the first measure of the fourth measure (measure 16-24) on the Bob staff, with a red arrow pointing to it from below.

Piano Phase ...

Bob

Alice

Piano Phase,
pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

1 (x4-8) 2 (x12-18) (x4-16) 3 (x16-24) (x4-16)

r.h. l.h. r.h. l.h. r.h. l.h. r.h. l.h.

mf non legato mf non legato mf non legato mf non legato

fade in non legato mf

hold tempo 1 accel. very slightly hold tempo 1 a.v.s.

4 (x16-24) 5 (x4-16) 6 (x16-24) (x4-16) (x16-24) (x4-16)

(tempo 1) (tempo 1) (tempo 1) (tempo 1) (tempo 1)

hold tempo 1 a.v.s. hold tempo 1 a.v.s. hold tempo 1 a.v.s.

The image shows a musical score for 'Piano Phase' by Steve Reich, arranged for two pianos or two marimbas. The score is written for two parts, Bob and Alice, each with a right-hand (r.h.) and left-hand (l.h.) part. The tempo is marked as approximately 72 beats per minute. The score is divided into six measures, each with a specific range of repetitions (e.g., 1 (x4-8), 2 (x12-18), etc.). The notation includes various musical symbols such as notes, rests, and dynamic markings (mf, non legato). Performance instructions like 'fade in', 'hold tempo 1', 'accel. very slightly', and 'a.v.s.' are included. A red box highlights a specific section of the score, likely indicating a point of interest or a specific performance technique.

Piano Phase ...

Piano Phase,

pour 2 pianos ou 2 marimbas

$\text{♩} = \text{ca. } 72$

Steve Reich

Bob

Alice

The musical score is written for piano and guitar. It consists of two systems of music, each with a piano part (left staff) and a guitar part (right staff).

System 1:

- Measure 1:** Piano part starts with *mf* non legato. Guitar part has a right-hand (r.h.) melody. Rehearsal mark 1 (x4-8).
- Measure 2:** Guitar part continues. Rehearsal mark 2 (x12-18).
- Measure 3:** Piano part has a "fade in" marking. Guitar part has a "hold tempo 1" marking. Rehearsal mark 3 (x16-24).
- Measure 4:** Piano part has an "accel. very slightly" marking. Guitar part has a "hold tempo 1" marking. Rehearsal mark 4 (x16-24).
- Measure 5:** Piano part has an "a.v.s." marking. Guitar part has an "a.v.s." marking. Rehearsal mark 5 (x16-24).

System 2:

- Measure 6:** Piano part has a "hold tempo 1" marking. Guitar part has a "hold tempo 1" marking. Rehearsal mark 6 (x16-24).
- Measure 7:** Piano part has an "a.v.s." marking. Guitar part has an "a.v.s." marking. Rehearsal mark 7 (x16-24).
- Measure 8:** Piano part has a "hold tempo 1" marking. Guitar part has a "hold tempo 1" marking. Rehearsal mark 8 (x16-24).
- Measure 9:** Piano part has an "a.v.s." marking. Guitar part has an "a.v.s." marking. Rehearsal mark 9 (x16-24).
- Measure 10:** Piano part has a "hold tempo 1" marking. Guitar part has a "hold tempo 1" marking. Rehearsal mark 10 (x16-24).

A red oval highlights the first measure of the second system (Measure 6), and a red arrow points to it from below.

Piano Phase ...

Bob

Alice

Piano Phase,

pour 2 pianos ou 2 marimbas

Steve Reich

$\text{♩} = \text{ca. } 72$

The musical score for 'Piano Phase' by Steve Reich is displayed. It consists of two staves, labeled 'Bob' and 'Alice'. The score is written for two pianos or two marimbas. The tempo is marked as 'ca. 72' (approximately 72 beats per minute). The score is divided into four measures, each with a specific rhythmic pattern and duration: 1 (x4-8), 2 (x12-18), 3 (x16-24), and 4 (x16-24). The notation includes various musical symbols such as treble clefs, notes, rests, and dynamic markings like 'mf' (mezzo-forte) and 'non legato'. The score also includes instructions for 'hold tempo 1' and 'a.v.s.' (ad libitum). The score is presented in a way that highlights the phase relationships between the two staves.

Problem:

We do not want to compute a priori
when resynchronizations will occur

... in Mixed Music

Live musician

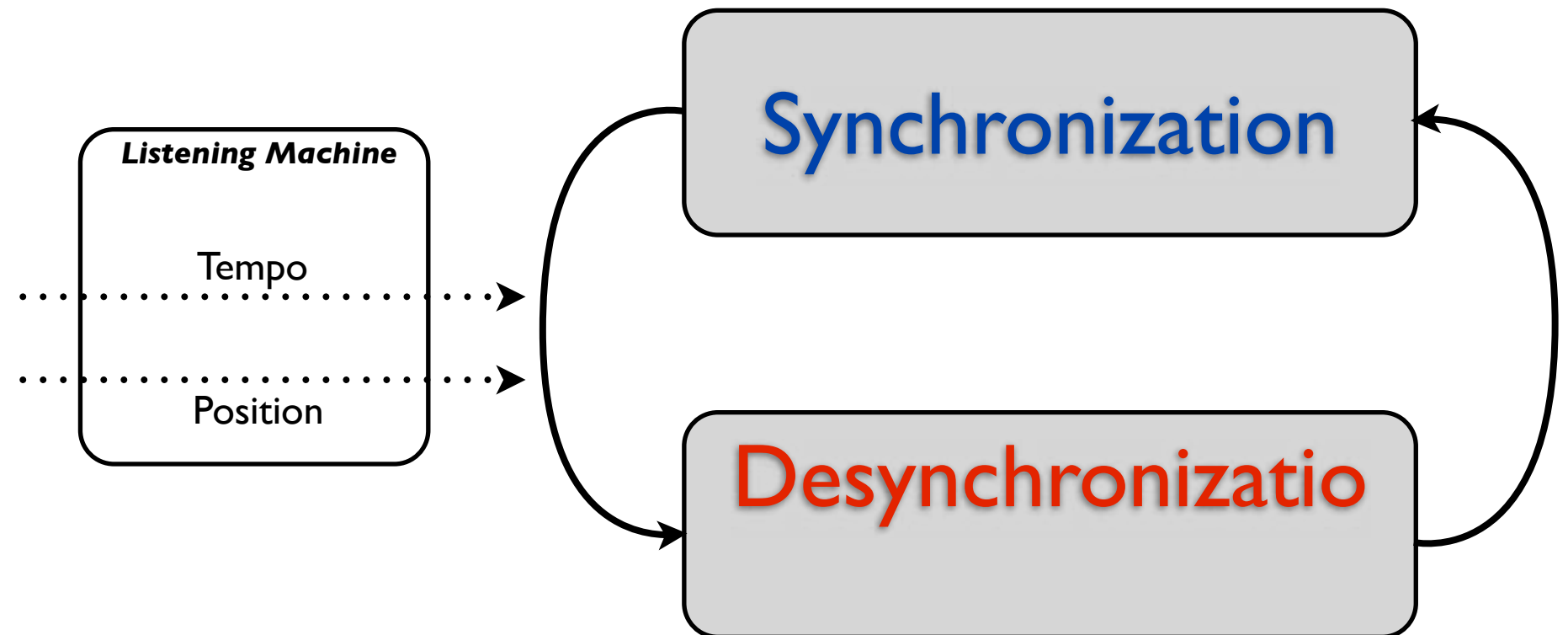
Plays the constant speed part



Bob

Electronic

Handles the desynchronization



Alice

... in Mixed Music

Live musician

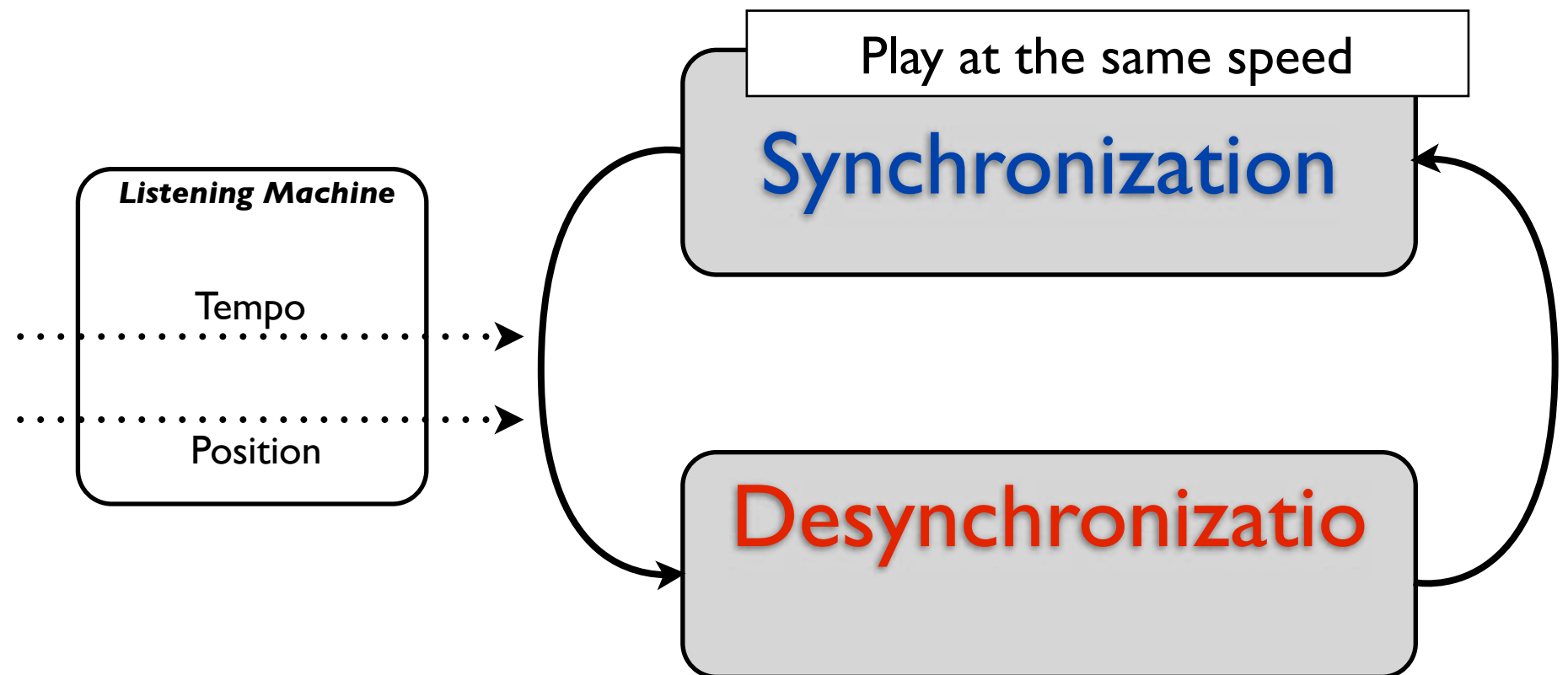
Plays the constant speed part



Bob

Electronic

Handles the desynchronization



Alice

... in Mixed Music

Live musician

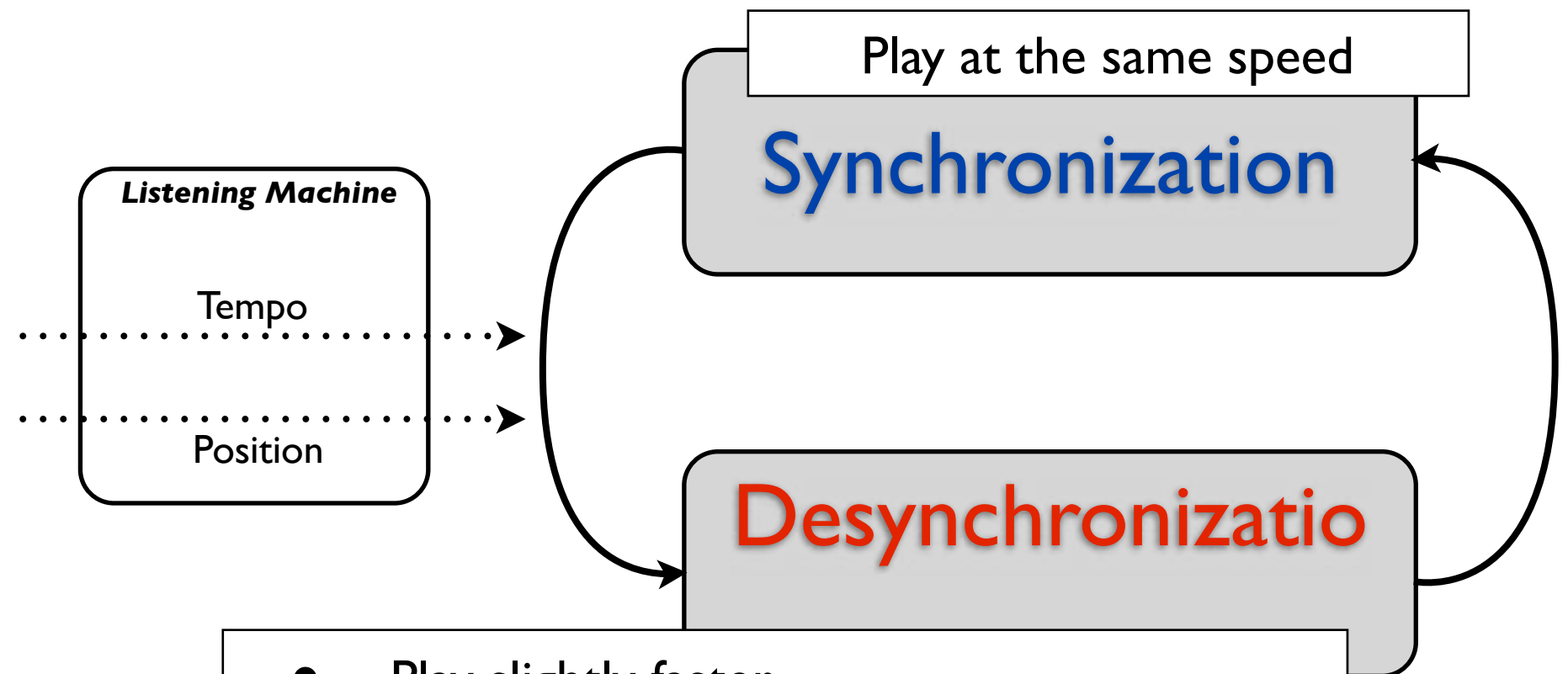
Plays the constant speed part



Bob

Electronic

Handles the desynchronization



- Play slightly faster
- Track the first note of Bob
- Resynchronize when the k-th note of Alice is close enough of the first note of Bob

Implementation

Two phases:
Synchronization
Desynchronization

```
let piano_phase sync desync first_note kth_note =  
  let rec process piano_phase k =  
    let ev = last_event asco in  
    run (melody ev 4 0.25 first_note);  
    emit desync;  
    do  
      let ev = last_event asco in  
      run (melody (ev+1) 16 0.2458 first_note) ||  
      run (track asco k kth_note) ||  
      run (compare asco first_note kth_note sync 0.05)  
    until sync done;  
    run (piano_phase ((k + 1) mod 12))  
  in  
  piano_phase 1  
in
```

Implementation

Synchronization

*Play the melody four times
and follow the tempo*

*Emit the signal `desync` after
four iterations of the melody*

```
let piano_phase sync desync first_note kth_note =  
  let rec process piano_phase k =  
    let ev = last_event asco in  
    run (melody ev 4 0.25 first_note);  
    emit desync;  
  do  
    let ev = last_event asco in  
    run (melody (ev+1) 16 0.2458 first_note) ||  
    run (track asco k kth_note) ||  
    run (compare asco first_note kth_note sync 0.05)  
  until sync done;  
  run (piano_phase ((k + 1) mod 12))  
in  
  piano_phase 1  
in
```


Implementation

Desynchronization

*Play slightly faster
and emit the signal `first_note`
whenever the first note is played*

Track the k -th note of the musician

*Compare the emission of signals
`kth_note` and `first_note` and emit
`sync` when they are close enough*

```
let piano_phase sync desync first_note kth_note =  
  let rec process piano_phase k =  
    let ev = last_event asco in  
    run (melody ev 4 0.25 first_note);  
    emit desync;  
    do  
      let ev = last_event asco in  
      run (melody (ev+1) 16 0.2458 first_note) ||  
      run (track asco k kth_note) ||  
      run (compare asco first_note kth_note sync 0.05)  
    until sync done;  
    run (piano_phase ((k + 1) mod 12))  
  in  
  piano_phase 1  
in
```

Conclusion

- **Link with the synchronous model** [EMSOFT 2013]
 - An executable semantics for Antescofo
 - A sequencer efficient enough w.r.t. human ear
 - Embedding in a synchronous reactive language
- **Applications** [FARM 2013]
 - Live coding
 - Prototyping new features:
new attributes, reactive behaviors, ...

To continue...



http://reactivem1.org/reactive_asco

References

[FARM'13] G. Baudart, L. Mandel, and M. Pouzet. Programming mixed music in ReactiveML. In Workshop on Functional Art, Music, Modeling and Design, 2013.

[EMSOF'13] G. Baudart, F. Jacquemard, L. Mandel, and M. Pouzet. A synchronous embedding of Antescofo, a domain-specific language for interactive mixed music. In Conference on Embedded Software, 2013.

[Echeveste et al 2012] J. Echeveste, A. Cont, J.-L. Giavitto, and F. Jacquemard. *Operational semantics of a domain specific language for real time musician-computer interaction*. Journal of Discrete Event Dynamic Systems, 2013.

[Cont 2008] A. Cont. *Antescofo: Anticipatory synchronization and control of interactive parameters in computer music*. In International Computer Music Conference, 2008.

[Mandel-Plateau 2008] L. Mandel and F. Plateau. *Interactive programming of reactive systems*. In Proceedings of Model-driven High-level Programming of Embedded Systems, 2008.

[Mandel-Pouzet 2005] L. Mandel and M. Pouzet. *ReactiveML: a reactive extension to ML*. In Proceedings of the International Conference on Principles and Practice of Declarative Programming, 2005.