A Synchronous Embedding of Antescofo,

a Domain-Specific Language for Interactive Mixed Music

Guillaume Baudart, ENS
Florent Jacquemard, IRCAM
Louis Mandel, ENS
Marc Pouzet, ENS

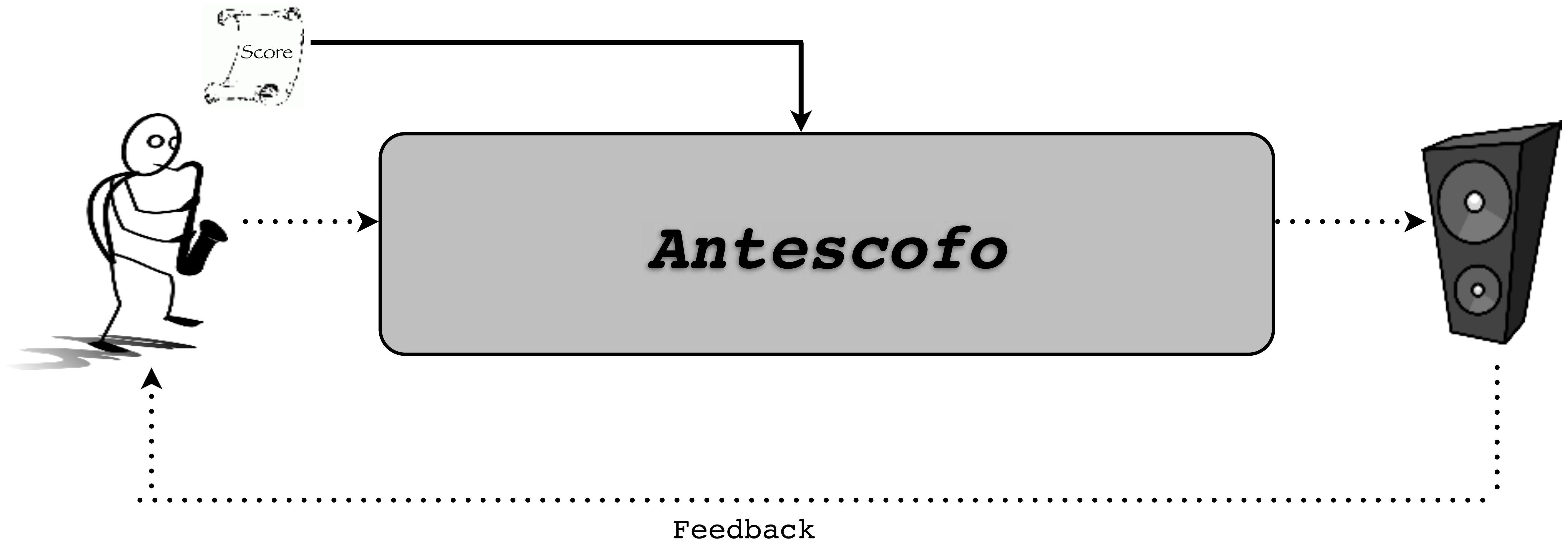
EMSOFT'13 **Montreal 09.30.2013**

Mixed Music and Antescofo

[Cont 2008]

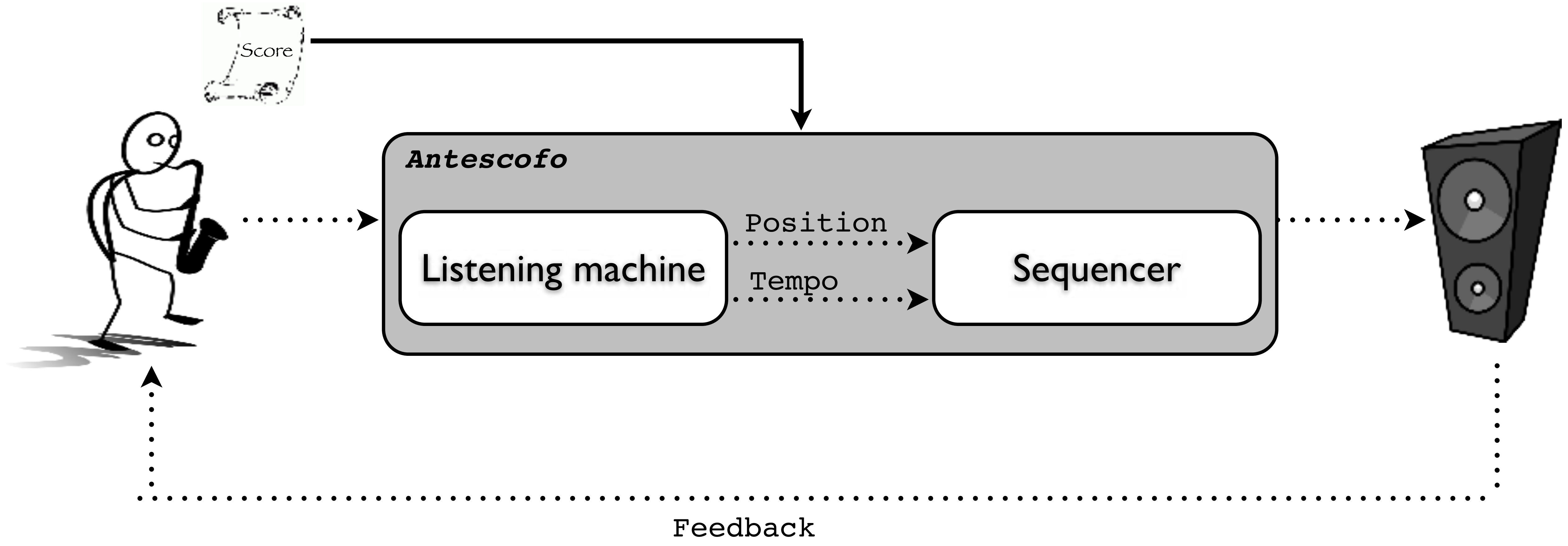
Mixed Music and Antescofo

[Cont 2008]



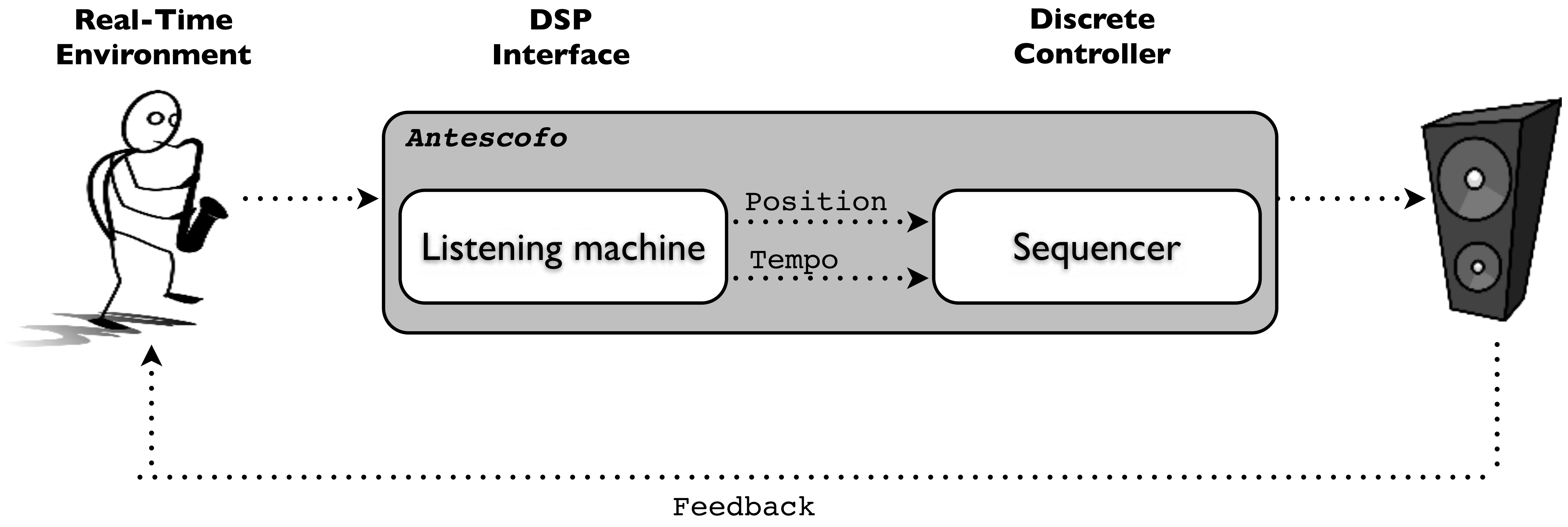
Antescofo Architecture

[Cont 2008]



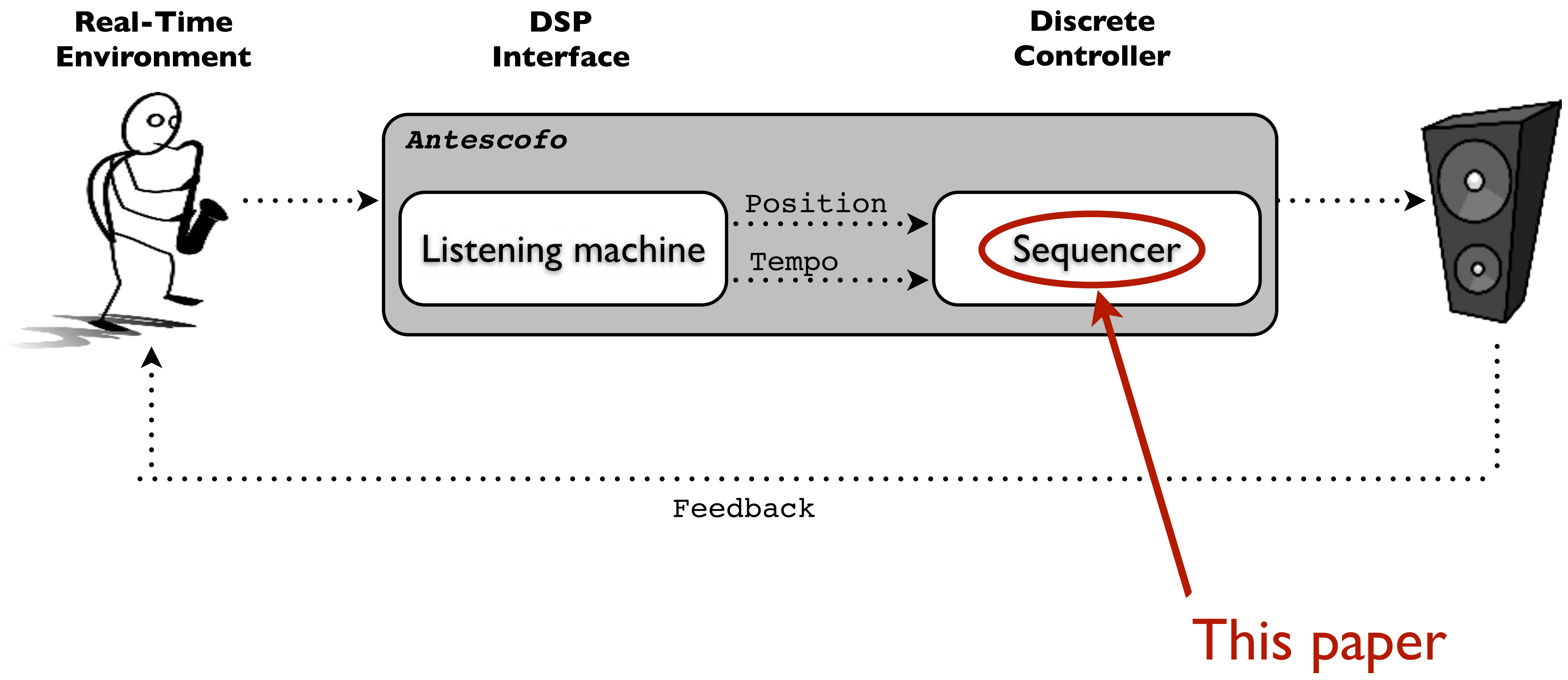
Antescofo Architecture

[Cont 2008]



Antescofo Architecture

[Cont 2008]



I. **The Antescofo Language**
   - Description
   - Synchronization and error handling strategies

II. **Semantics**
   - Formalization
   - The three predicates

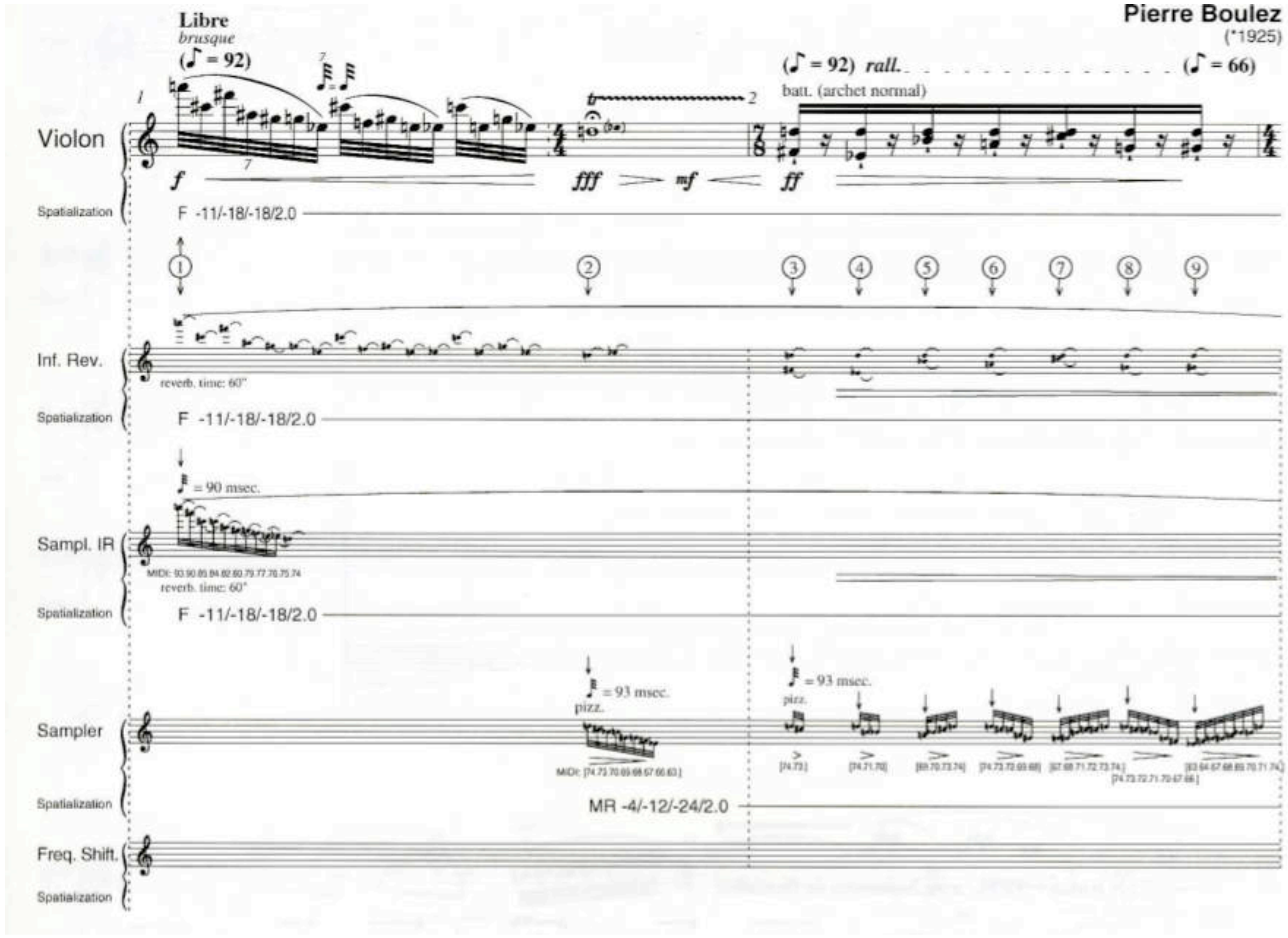
III. **Implementation**
   - Architecture
   - Embedding in ReactiveML

The Antescofo Language

The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

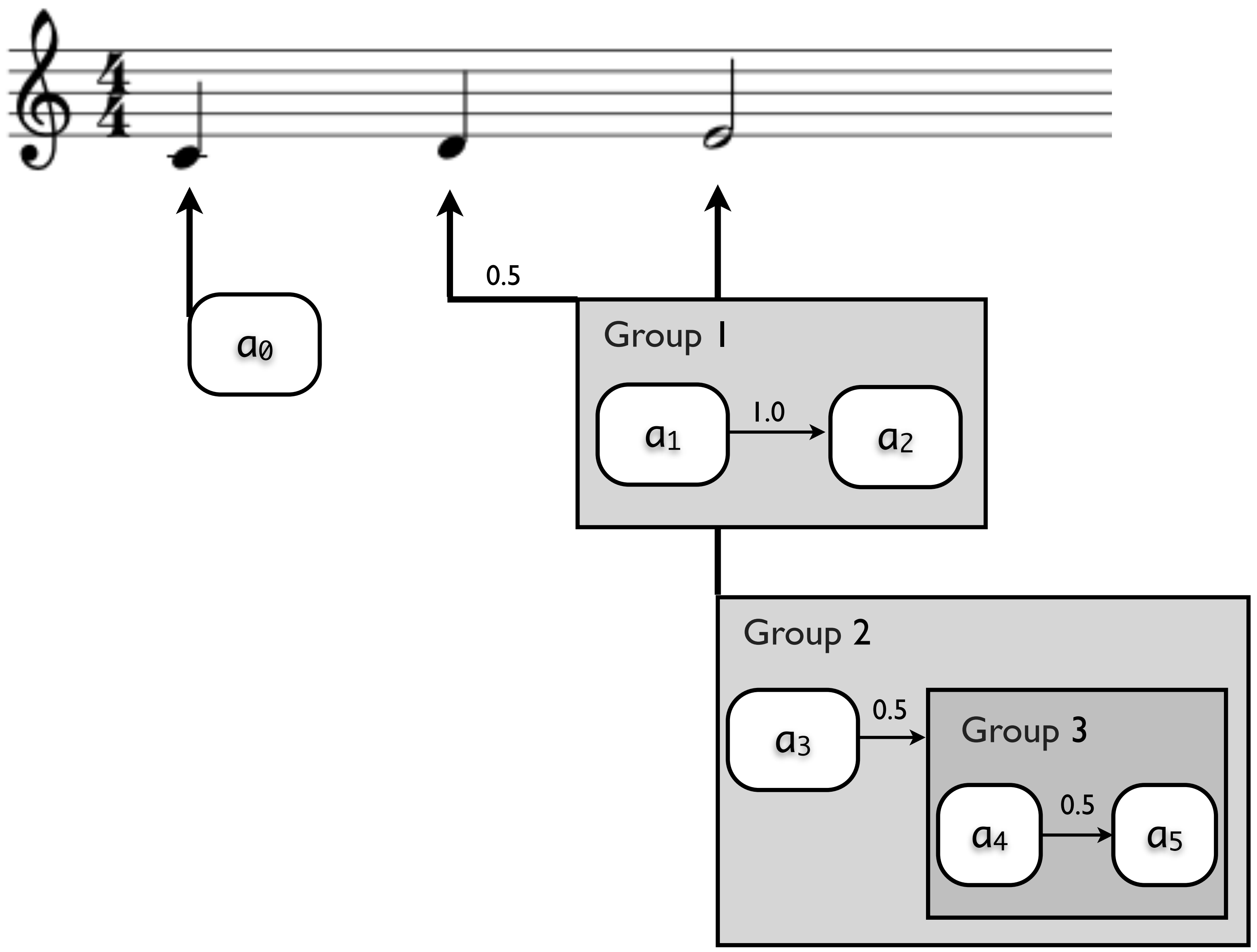
[Echeveste et al. 2012]



The Antescofo Language

Goal: Jointly specify electronic and instrumental parts

[Echeveste et al. 2012]



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]

Instrumental Score

```
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]

```
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]

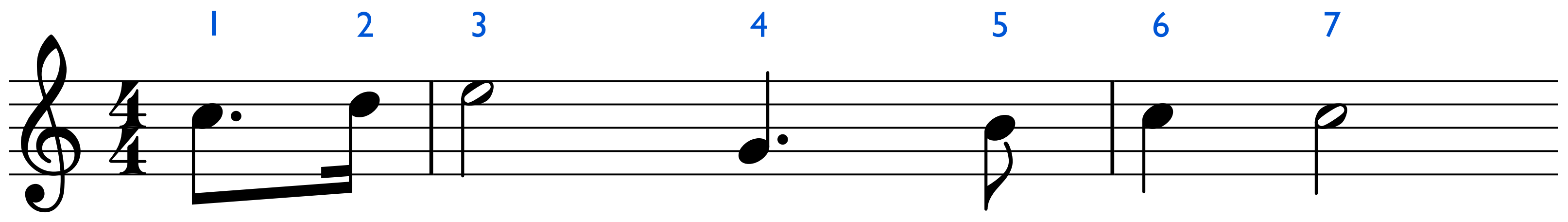
```
NOTE   65   1.0
0.25   GROUP  tight  partial
        {  1.0   'a_11'
           1.0   'a_12'  }

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

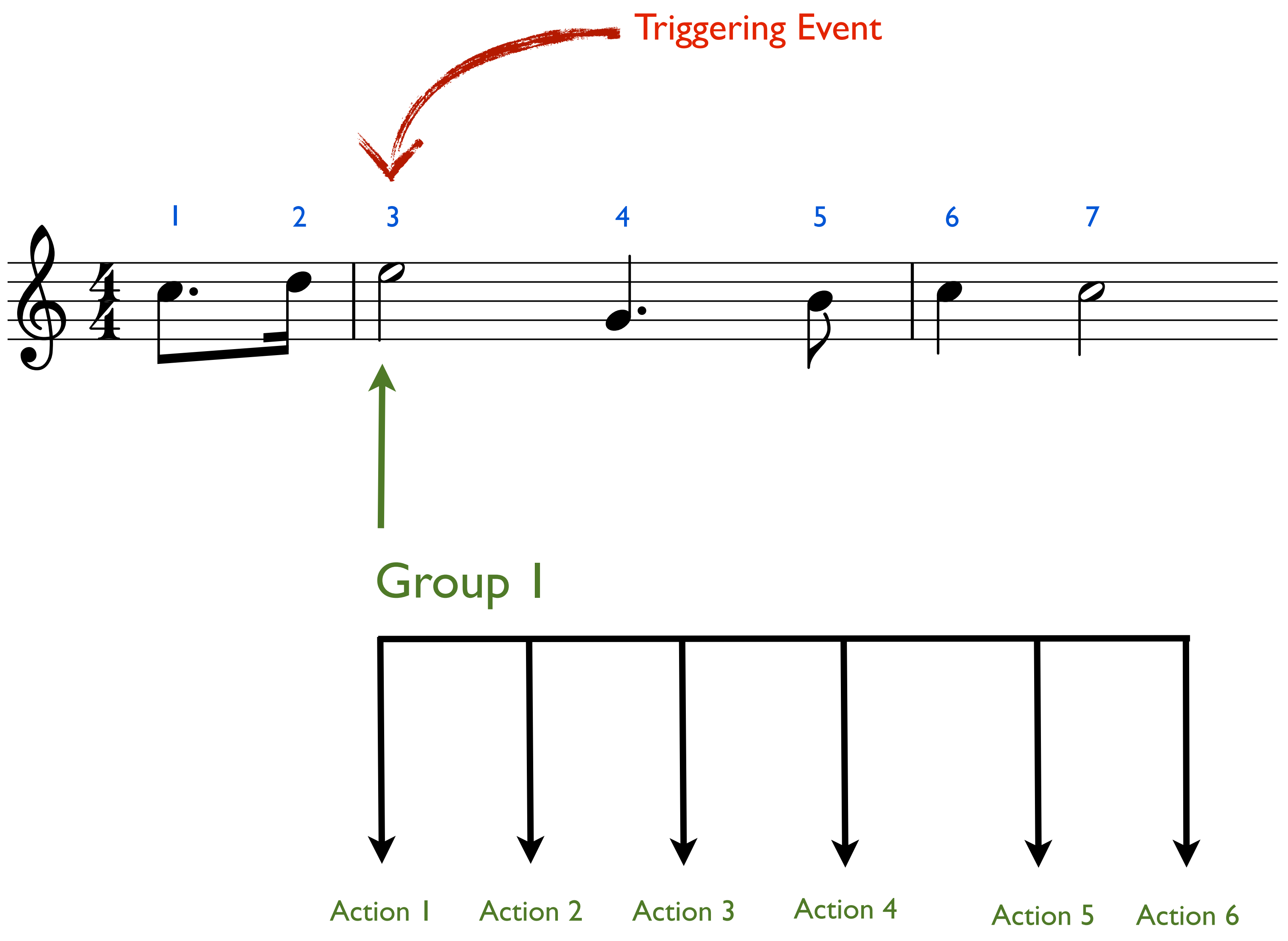
Language Characteristics

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

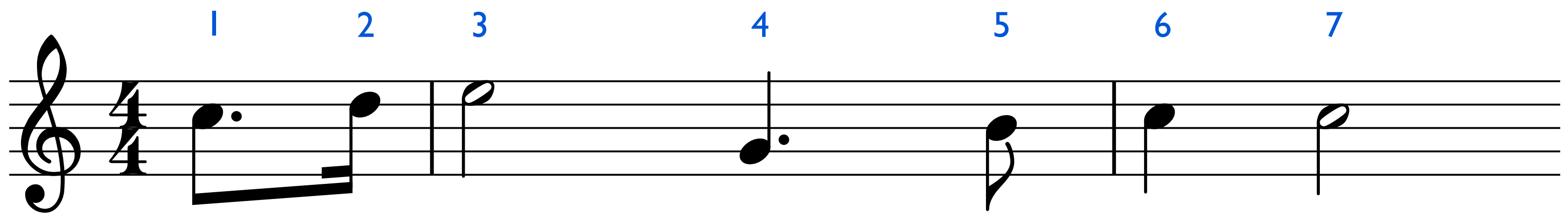
Synchronization Strategies



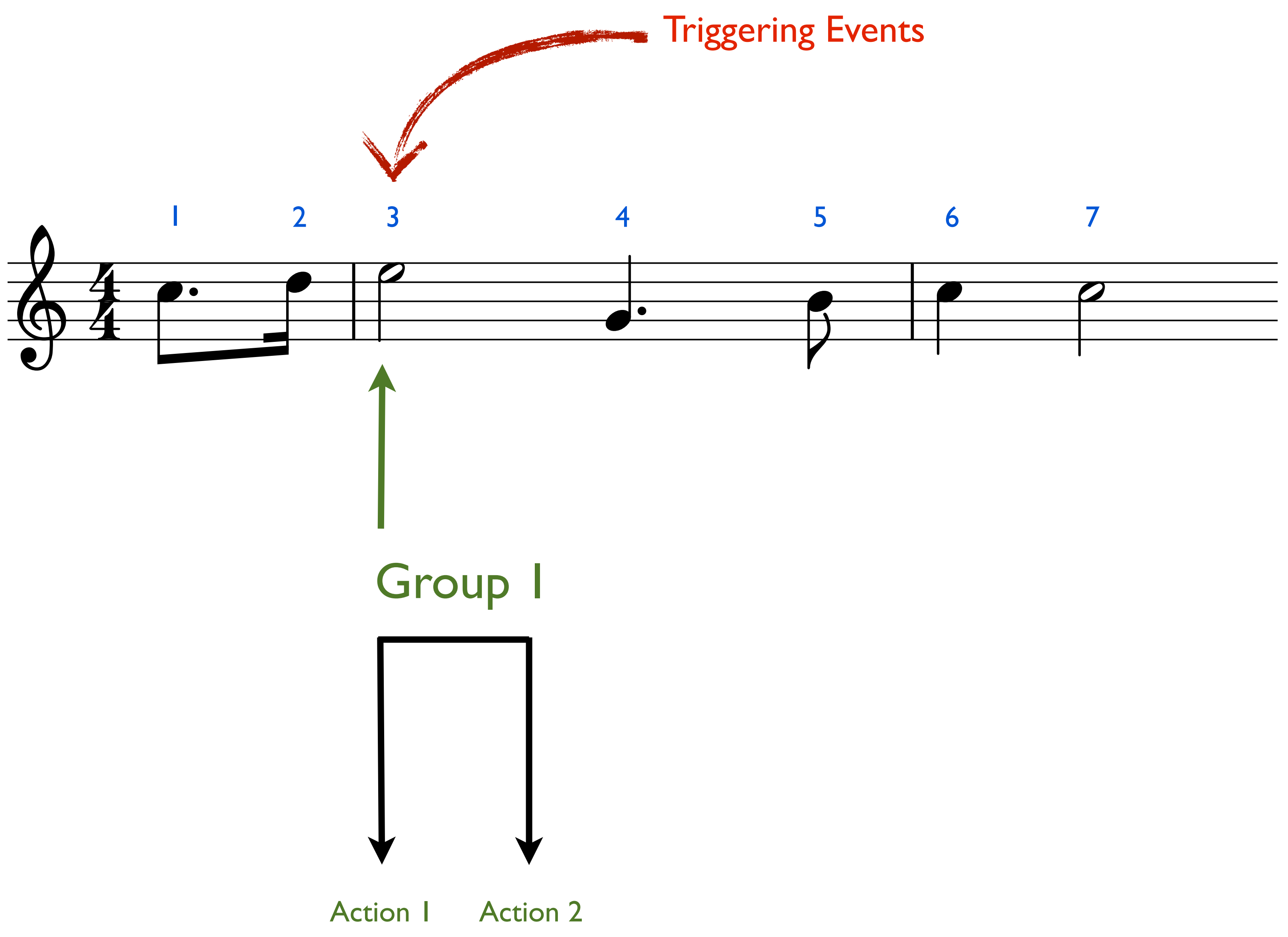
Loose: Synchronization with the tempo stream.



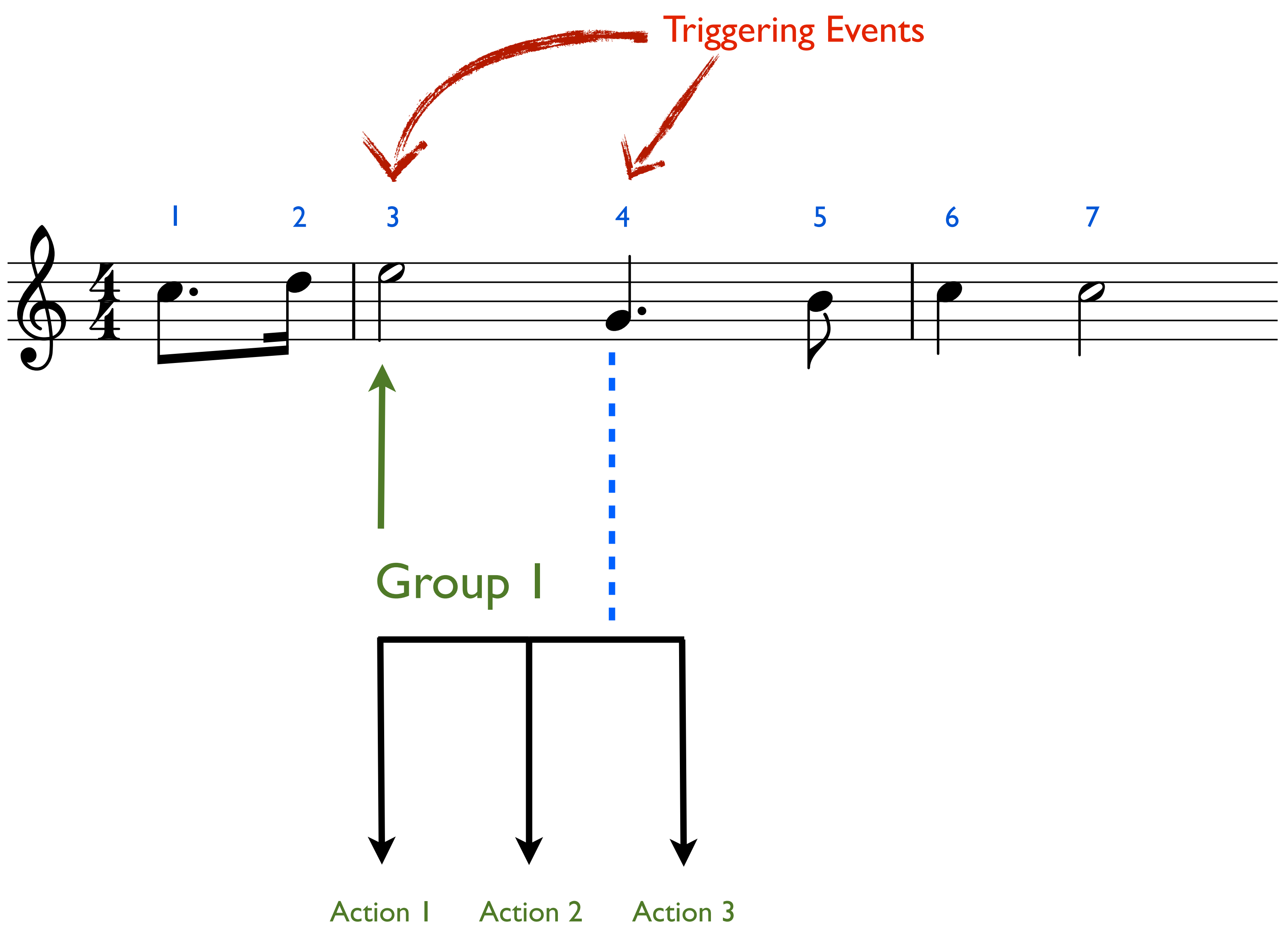
Loose: Synchronization with the tempo stream.



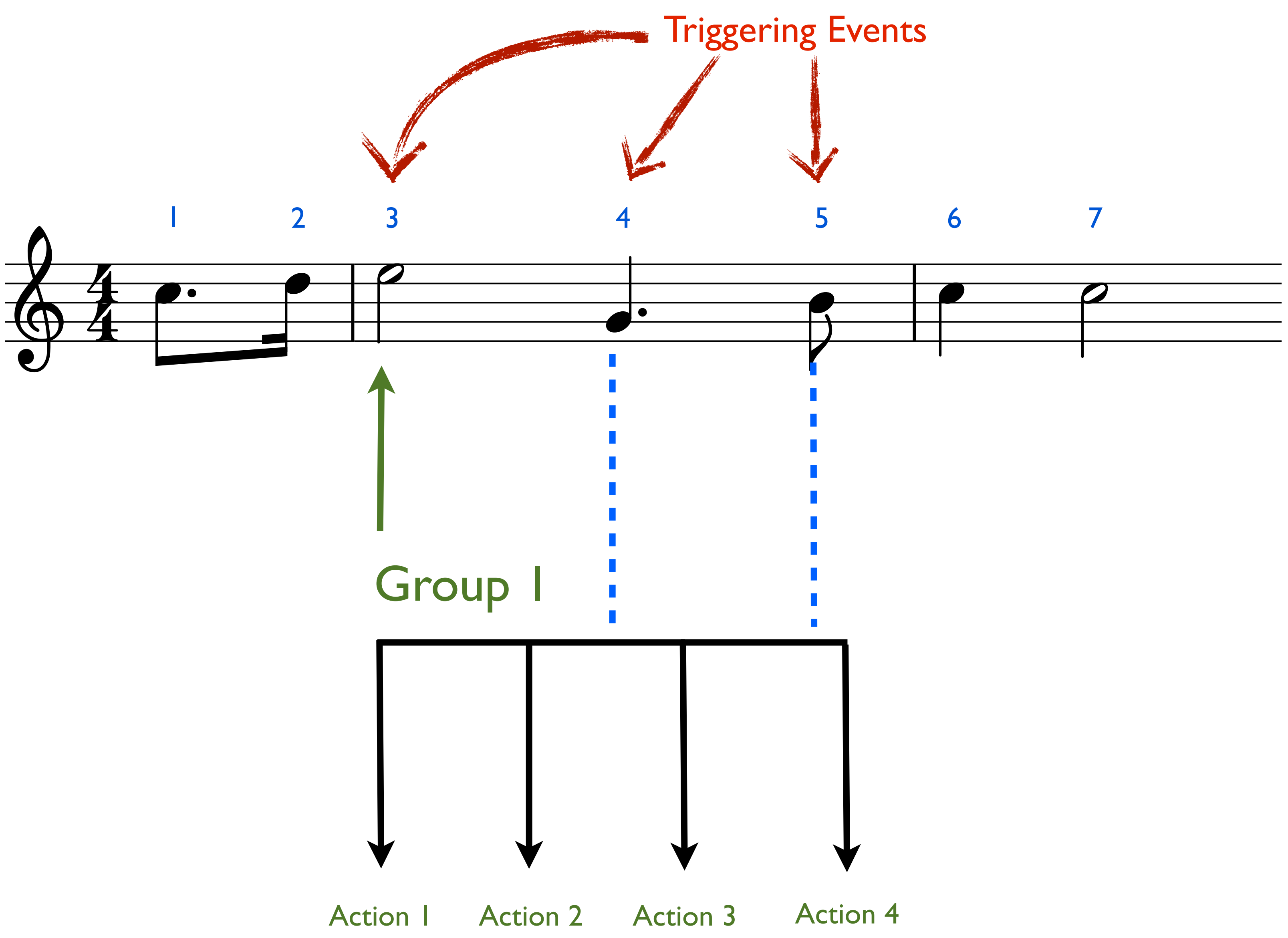
Tight: Synchronization
with tempo and events stream.



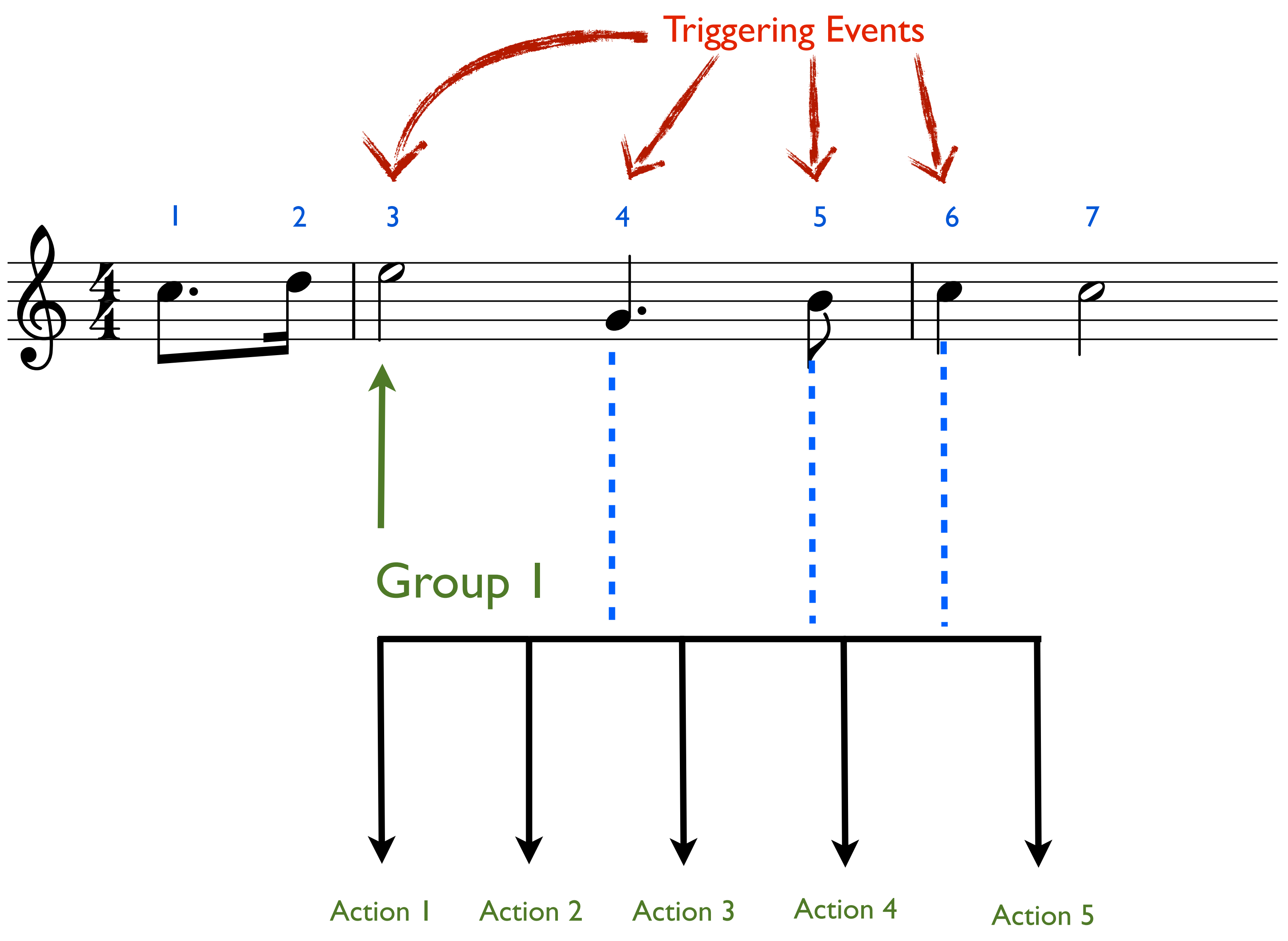
Tight: Synchronization
with tempo and events stream.



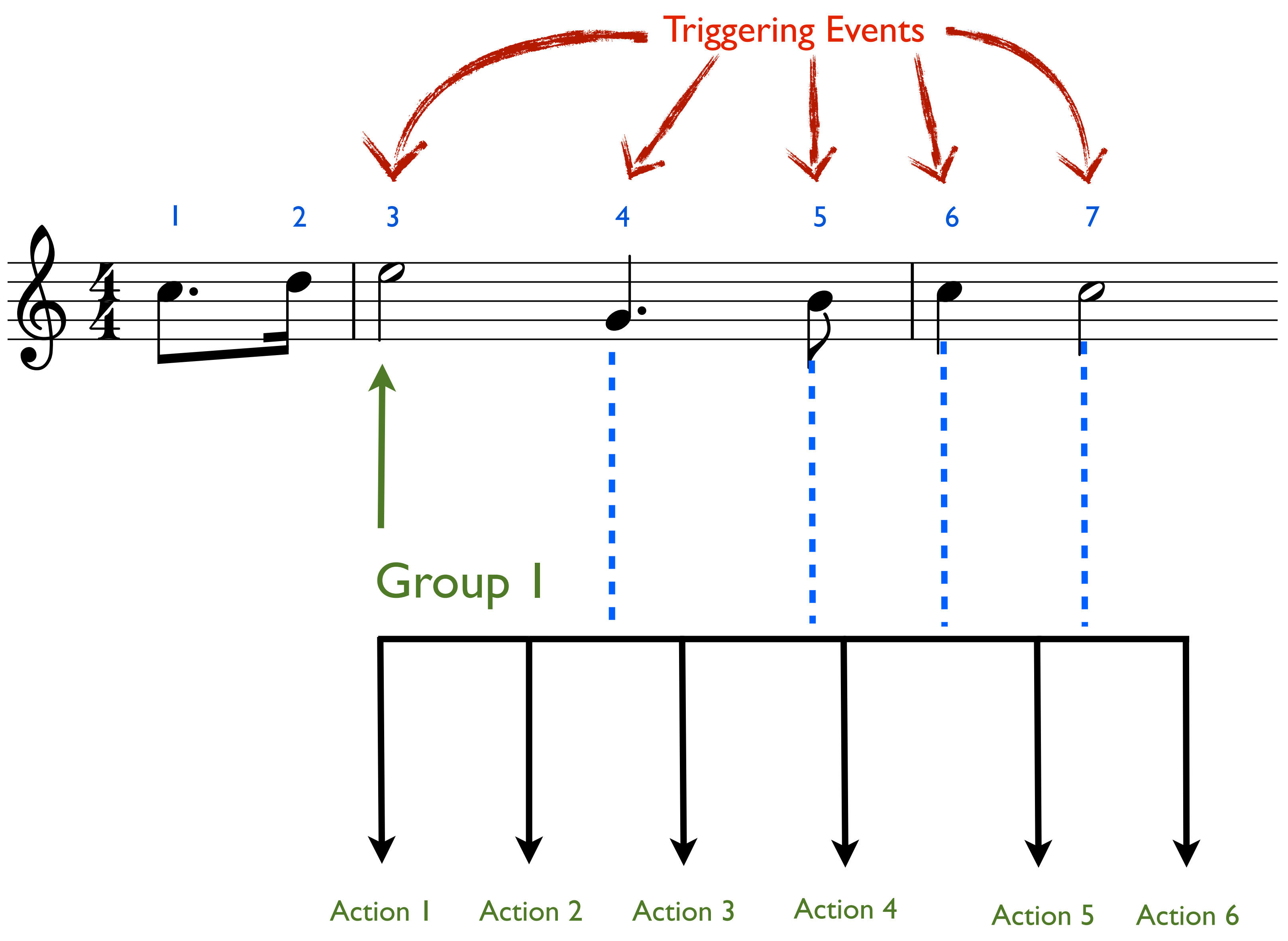
Tight: Synchronization
with tempo and events stream.



Tight: Synchronization
with tempo and events stream.

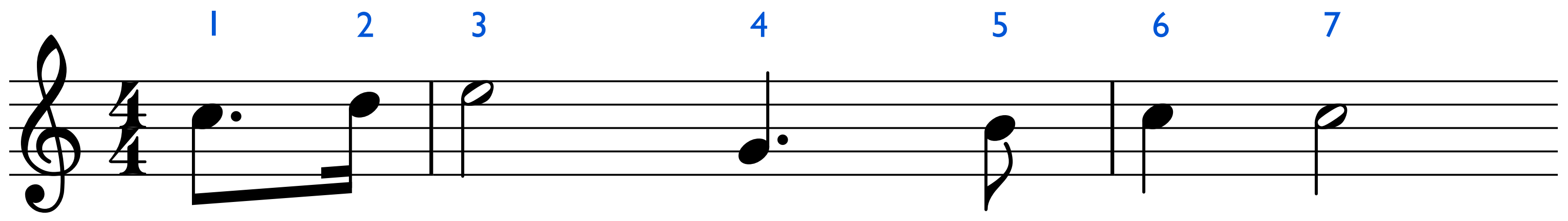


Tight: Synchronization
with tempo and events stream.

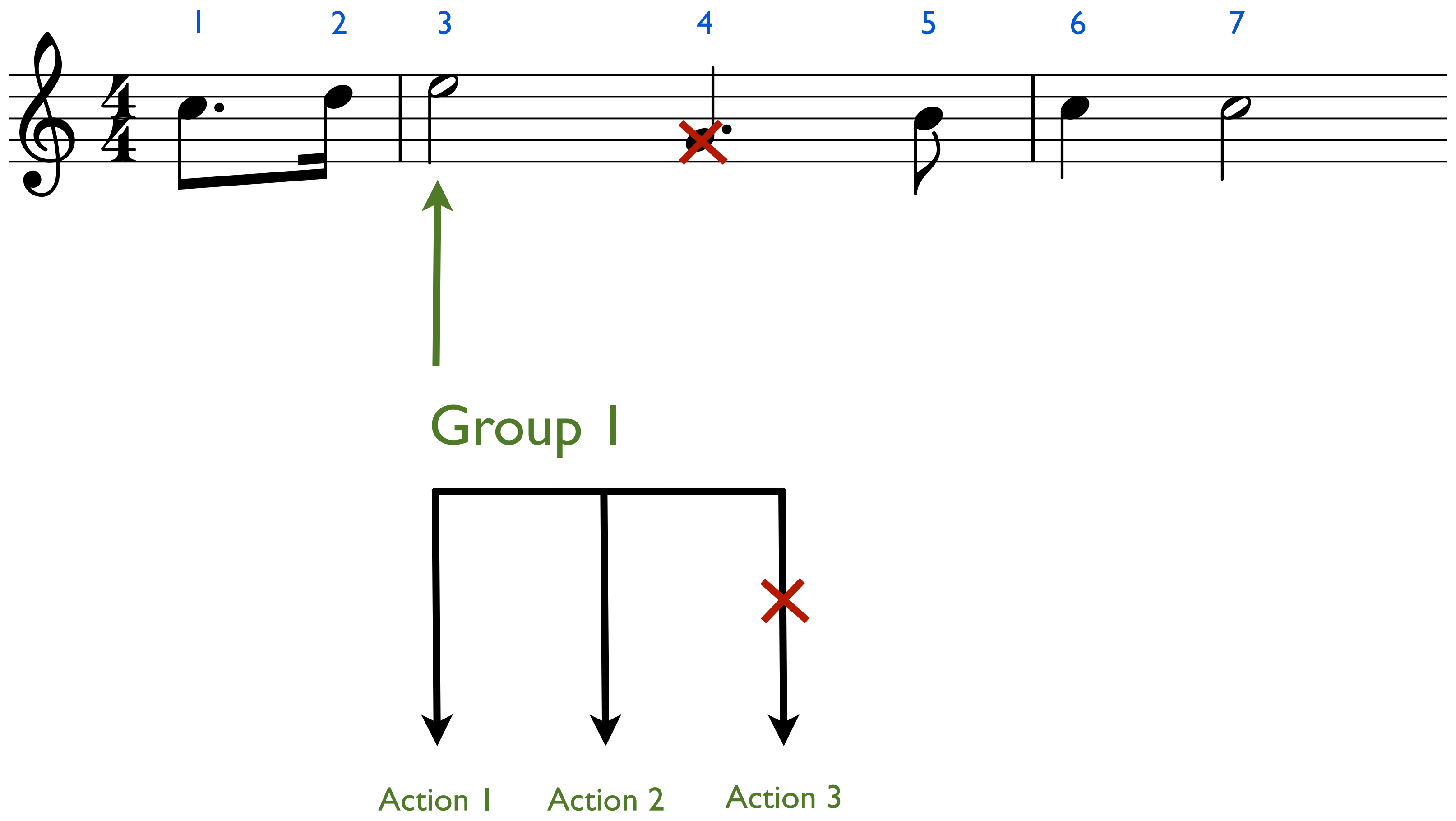


Tight: Synchronization
with tempo and events stream.

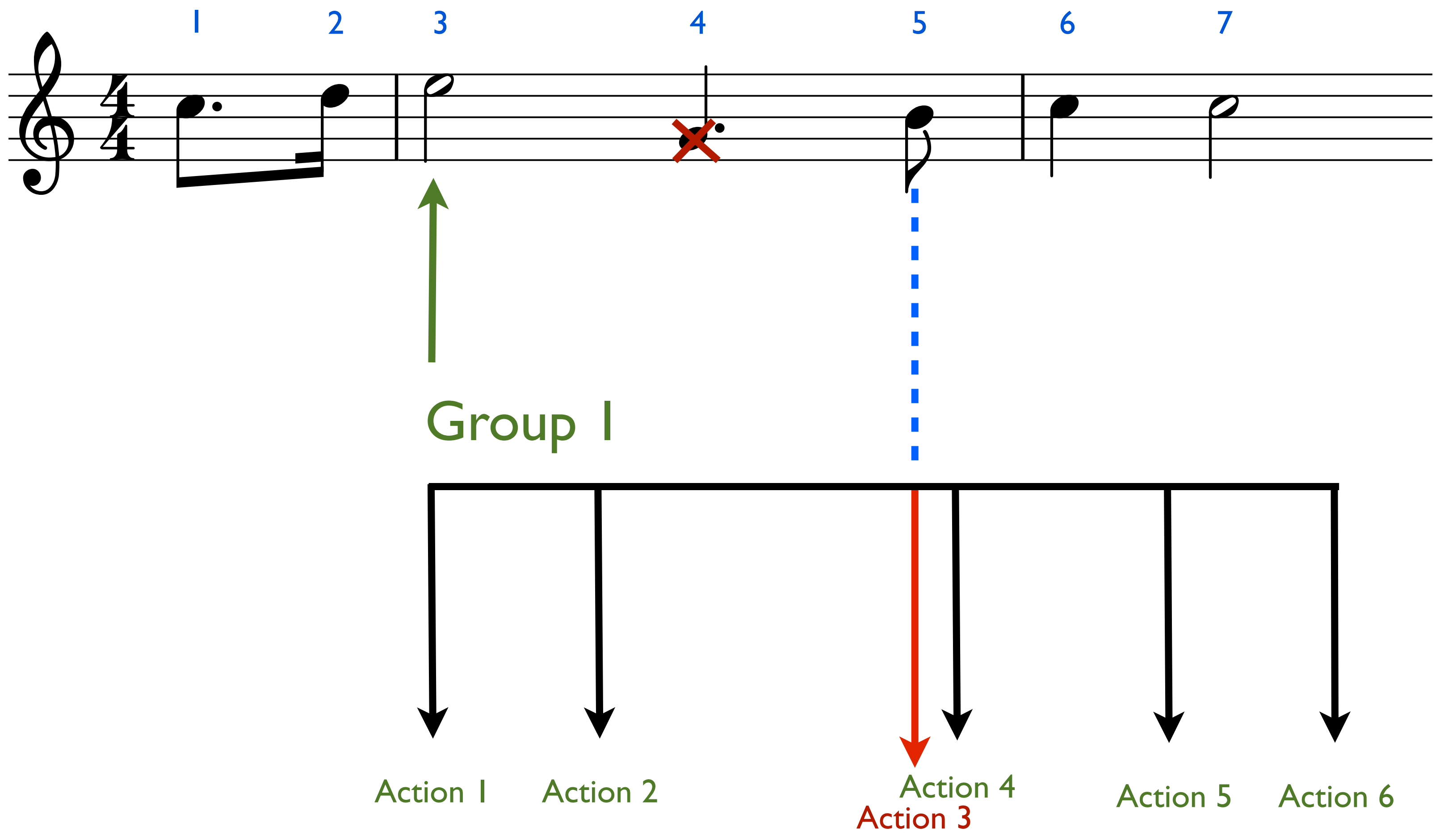
Error Handling Strategies



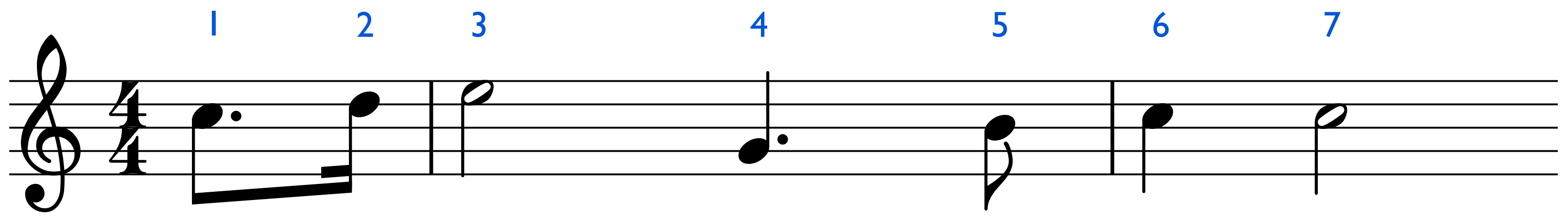
Causal: Actions should be launched immediately when the system recognizes the absence of the triggering event



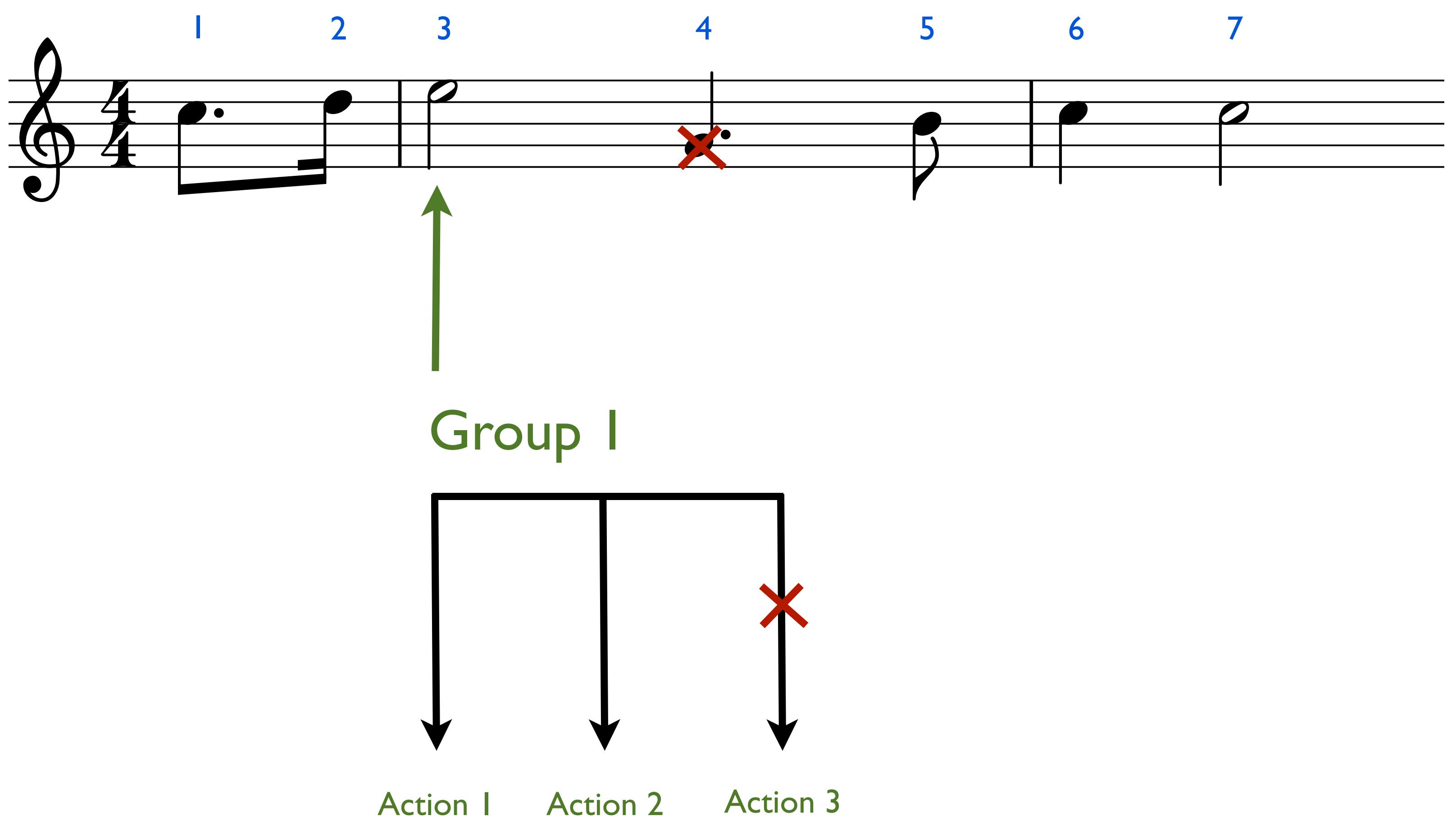
Causal: Actions should be launched immediately when the system recognizes the absence of the triggering event



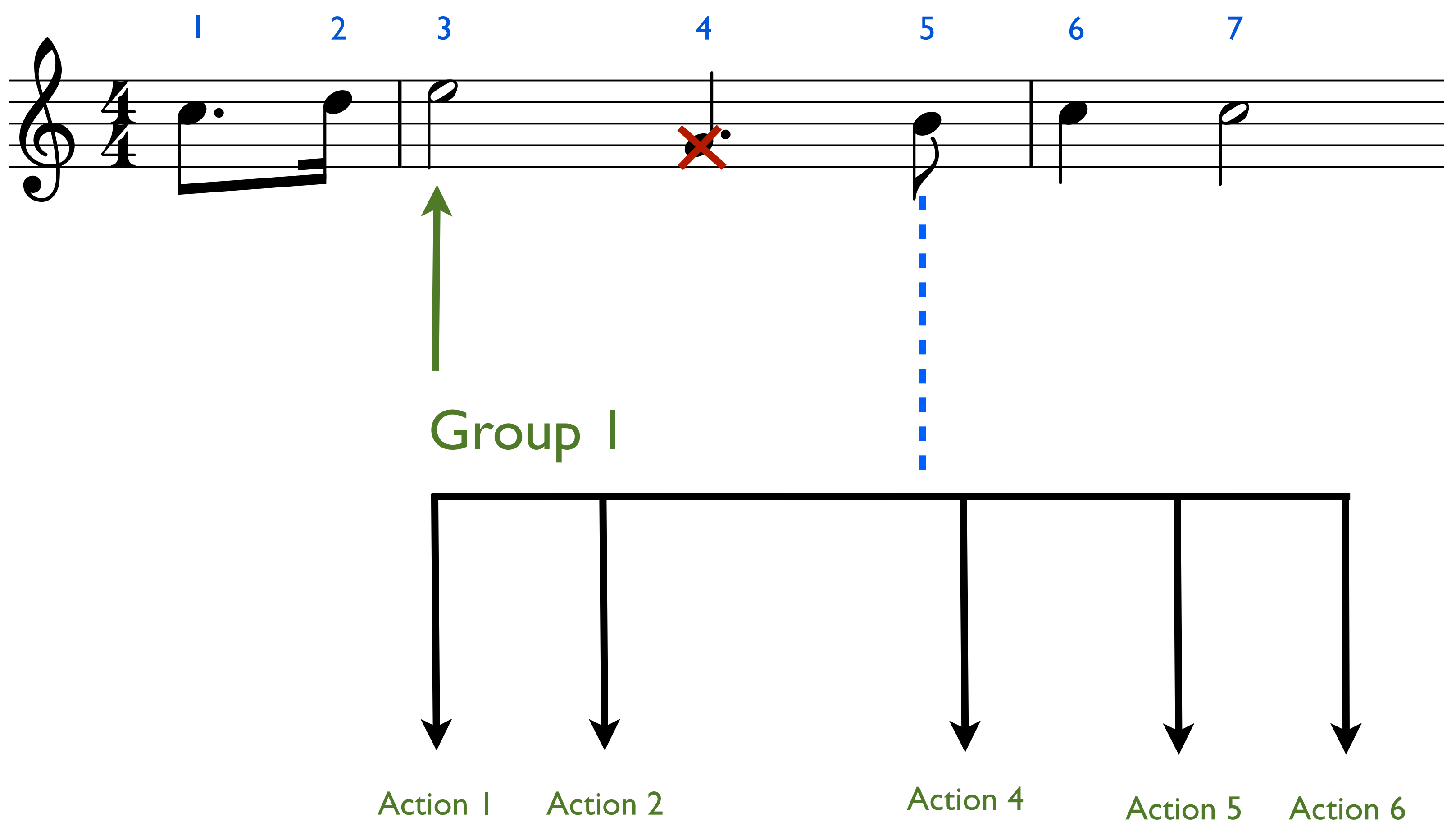
Causal: Actions should be launched immediately when the system recognizes the absence of the triggering event



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



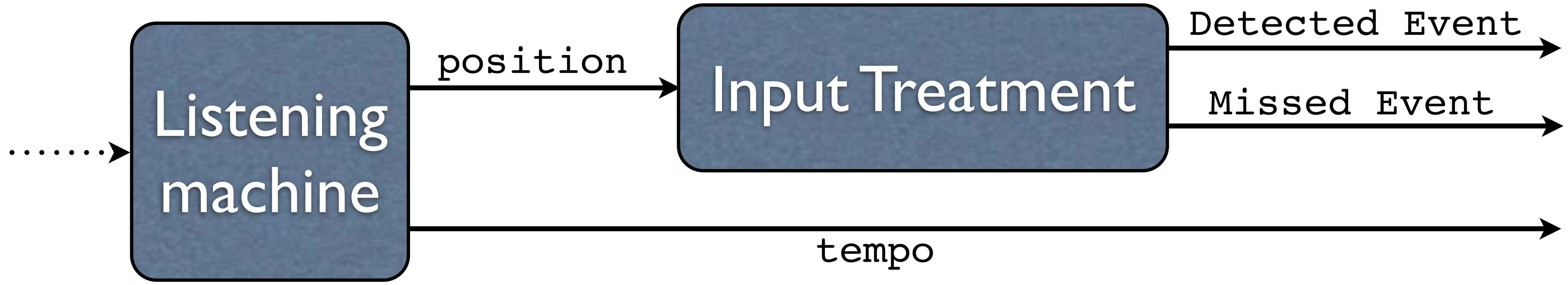
Partial: Actions should be dismissed in the absence of the triggering event



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

Semantics

Detected and Missed Event

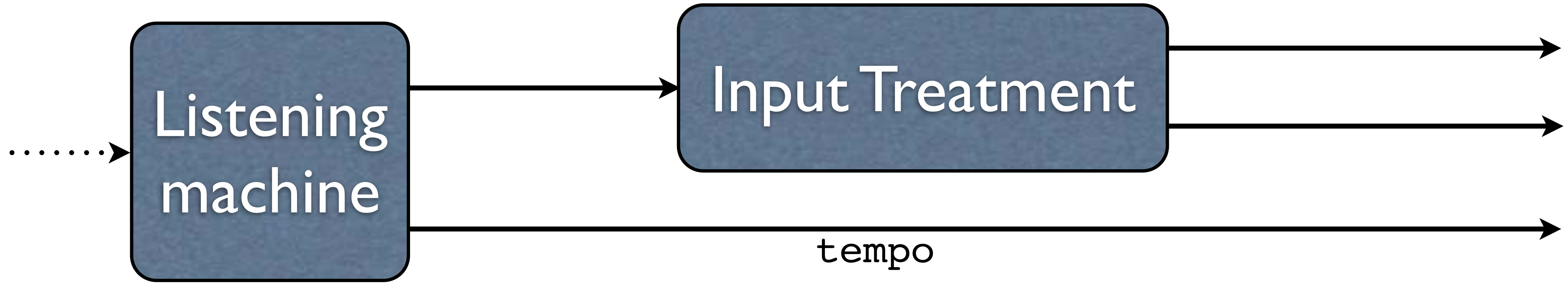


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

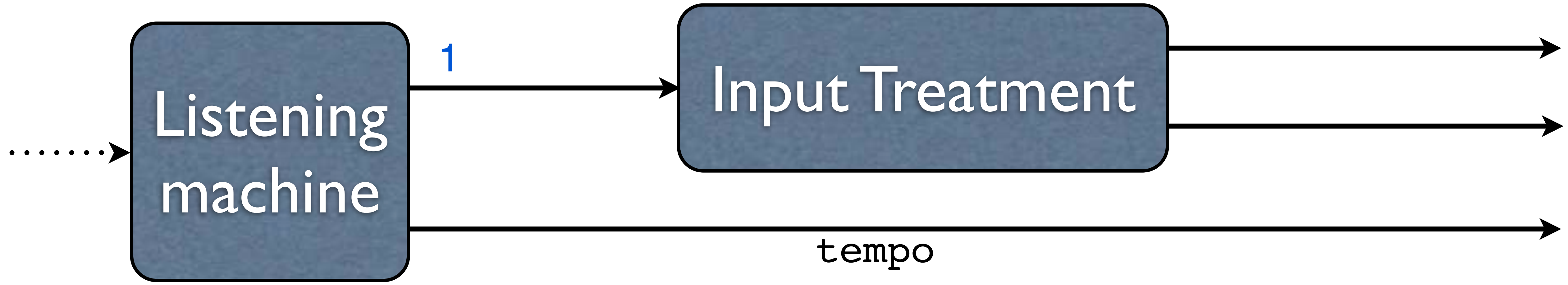


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

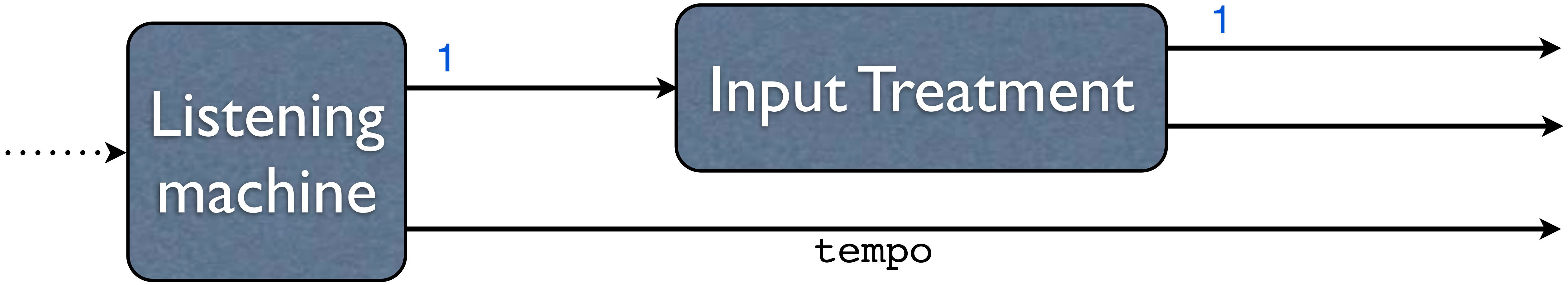


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

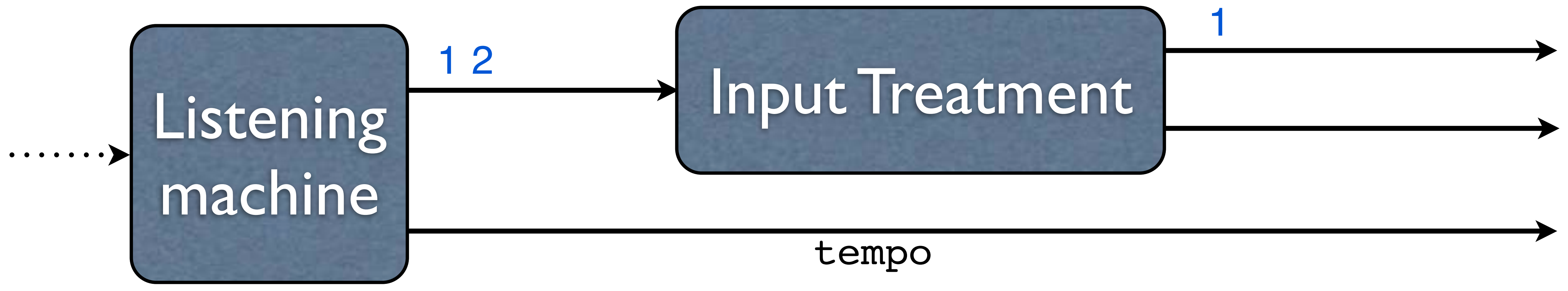


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

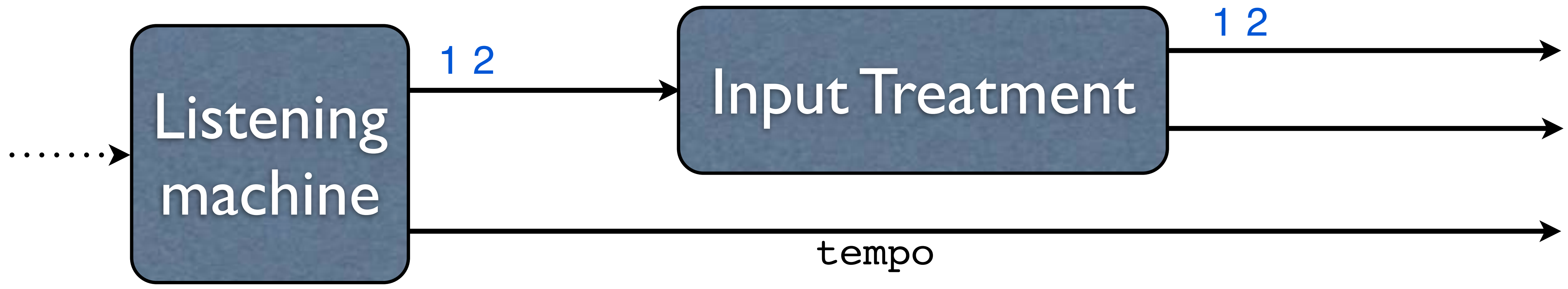


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

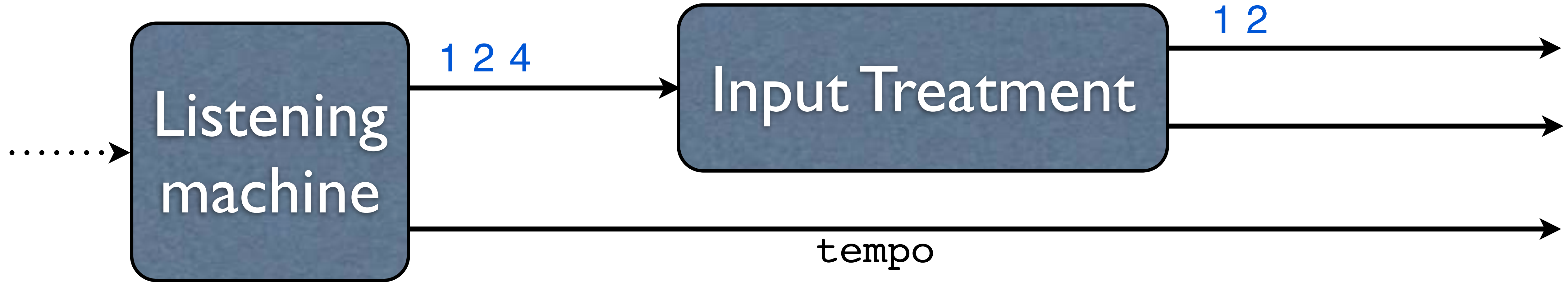


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

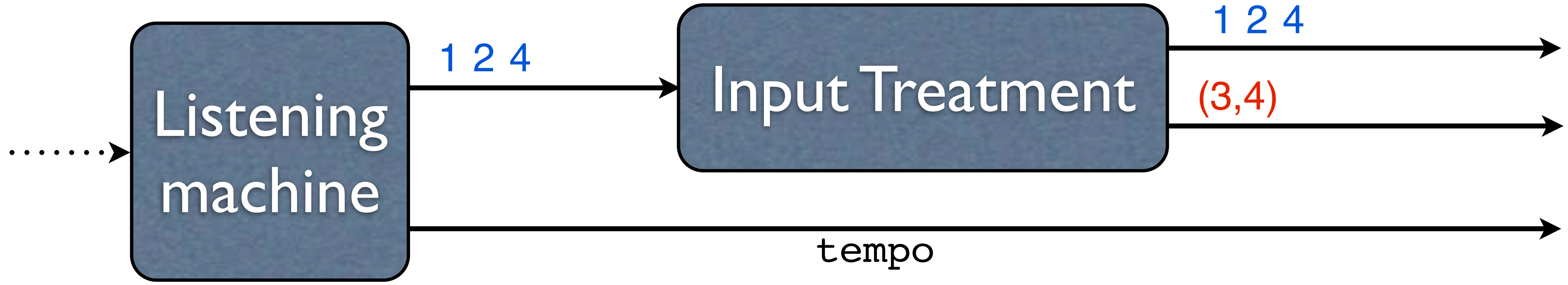


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

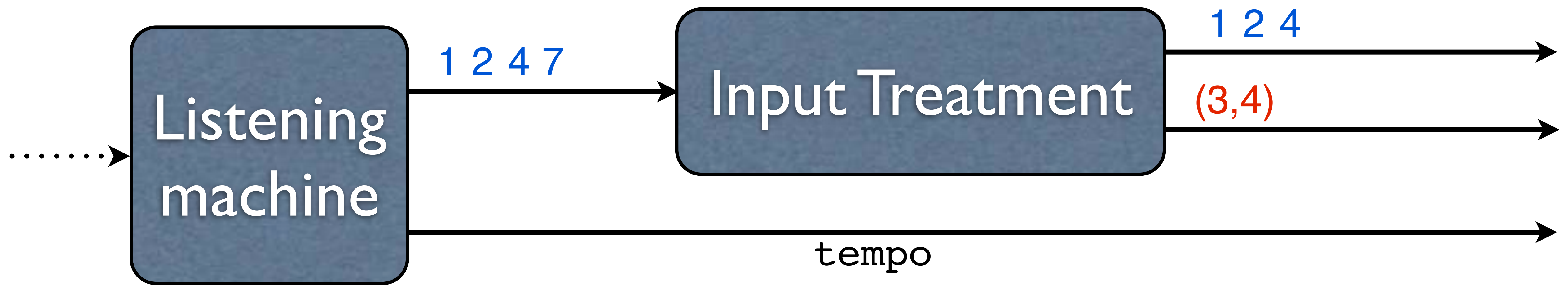


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event

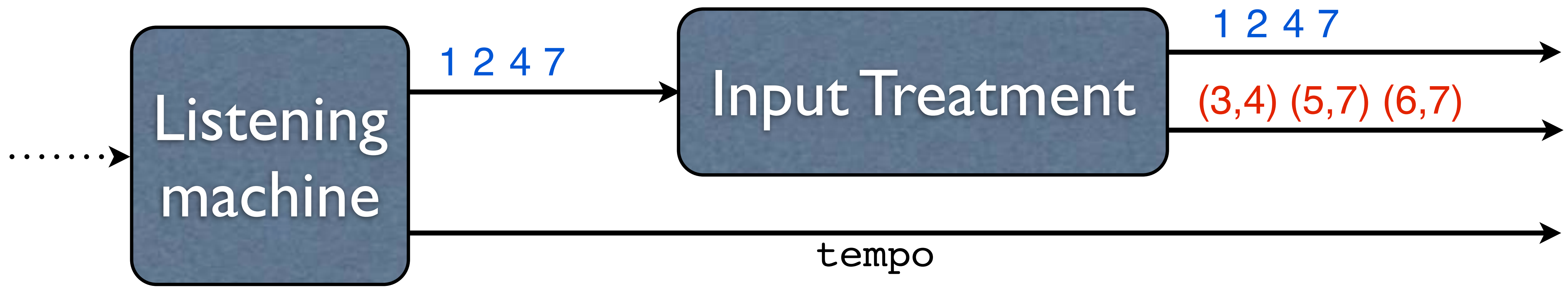


$\mathcal{E}(i)$: date of event i

For each missed event i we associate the next detected event

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

Detected and Missed Event



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

For each missed event i we associate the next detected event

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

Formalization

$$\begin{array}{rcl}
\mathit{score} & ::= & \varepsilon \mid (\mathit{event} : \mathit{seq})\ \mathit{score} \\
\mathit{event} & ::= & \texttt{event}\ i\ t \\
\mathit{seq} & ::= & \varepsilon \mid (\delta\ \mathit{ae})\ \mathit{seq} \\
\mathit{ae} & ::= & \mathit{action} \mid \mathit{group} \\
\mathit{group} & ::= & \texttt{group}\ \mathit{synchro}\ \mathit{error}\ \mathit{seq} \\
\mathit{synchro} & ::= & \texttt{tight} \mid \texttt{loose} \\
\mathit{error} & ::= & \texttt{local} \mid \texttt{global} \mid \texttt{partial} \mid \texttt{causal}
\end{array}$$

t	$\in$	$\mathbb{Q}$	Duration
δ	$\in$	$\mathbb{Q}$	Delay
i	$\in$	$\mathbb{N}$	Label
a	$\in$	$\mathcal{A}$	Action

A performance $perf$ is a set of triplets (i, δ, a)

D is the set of detected instrumental event

Semantics

$$D \overset{exec}{\vdash} score \Rightarrow perf$$

The Three Predicates

$$D \;\overset{exec}{\vdash}\; sc \Rightarrow p$$

Execute a score

$$D, i, \delta \;\overset{detected}{\vdash}\; seq \Rightarrow p$$

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

$$D, i, \delta \;\overset{missed}{\vdash}\; seq \Rightarrow p$$

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

Execution of a score

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

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

Triggering

$$\text{(Detect)}\quad \frac{i \in D \qquad D, i, 0.0 \;\left|\!\overline{\;{}^{detected}\;}\right.\; seq \Rightarrow p}{D \;\left|\!\overline{\;{}^{exec}\;}\right.\; (\texttt{event}\; i\; t : seq) \rightarrow p}$$

$$\text{(Miss)}\quad \frac{i \notin D \qquad D, i, 0.0 \;\left|\!\overline{\;{}^{missed}\;}\right.\; seq \Rightarrow p}{D \;\left|\!\overline{\;{}^{exec}\;}\right.\; (\texttt{event}\; i\; t : seq) \rightarrow p}$$

Execution: Atomic Actions

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

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

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

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

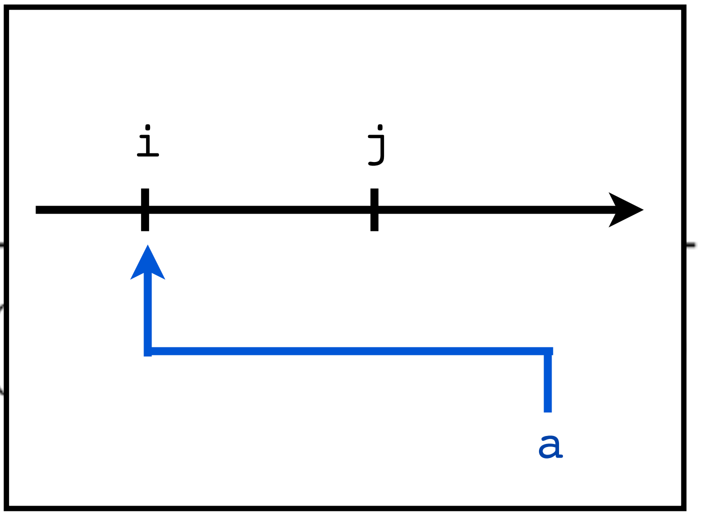
Error detection: i is missed

j is the first detection after i

Execution: Atomic Actions

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

(Missed Action)
$$\frac{\mathcal{M}(i) = j}{D, i, \delta \mid\!\underline{\ missed\ }\ a \rightarrow (j, \max(0.0, \mathcal{E}(}$$



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

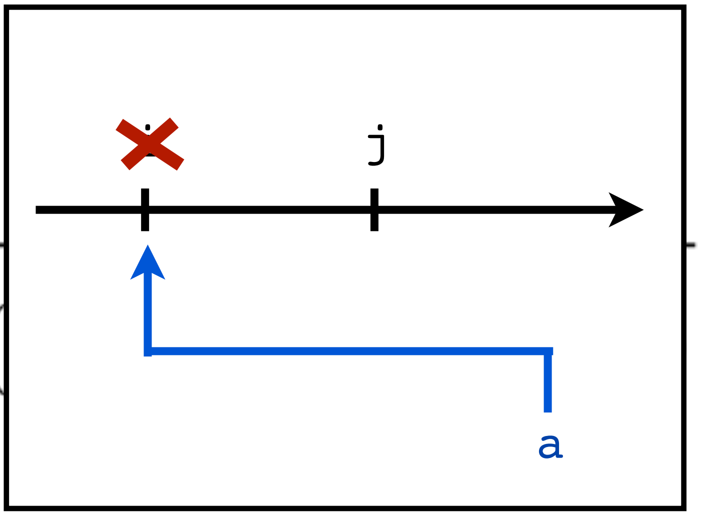
$$\mathcal{M}(i) = \min\{j \in D \mid \mathcal{E}(j) > \mathcal{E}(i)\}$$

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

Execution: Atomic Actions

(Detected Action) $$\frac{}{D, i, \delta \vdash^{detected} a \to (i, \delta, a)}$$

(Missed Action) $$\frac{\mathcal{M}(i) = j}{D, i, \delta \vdash^{missed} a \to (j, \max(0.0, \mathcal{E}(}$$



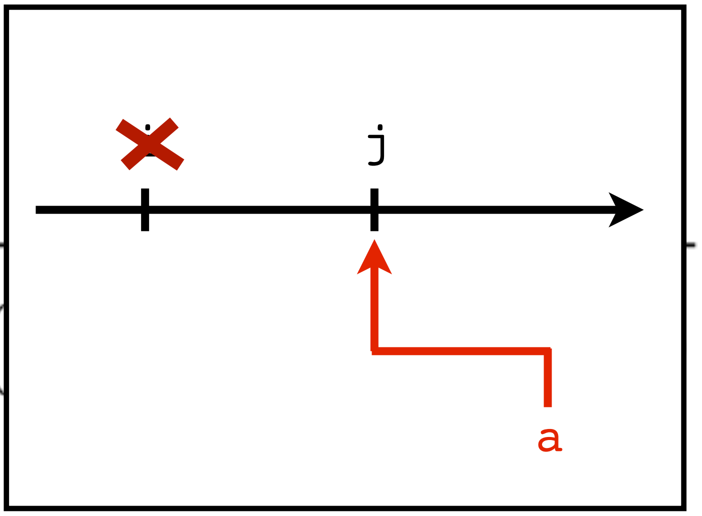
$\mathcal{E}(i)$: Position of event i

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

Execution: Atomic Actions

(Detected Action) $$\frac{}{D, i, \delta \vdash^{\mathit{detected}} a \rightarrow (i, \delta, a)}$$

(Missed Action) $$\frac{\mathcal{M}(i) = j}{D, i, \delta \vdash^{\mathit{missed}} a \rightarrow (j, \max(0.0, \mathcal{E}(}$$



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

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

Error detection: i is missed

j is the first detection after i

Execution: Atomic Actions

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

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

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

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

Error detection: i is missed

j is the first detection after i

Implementation

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

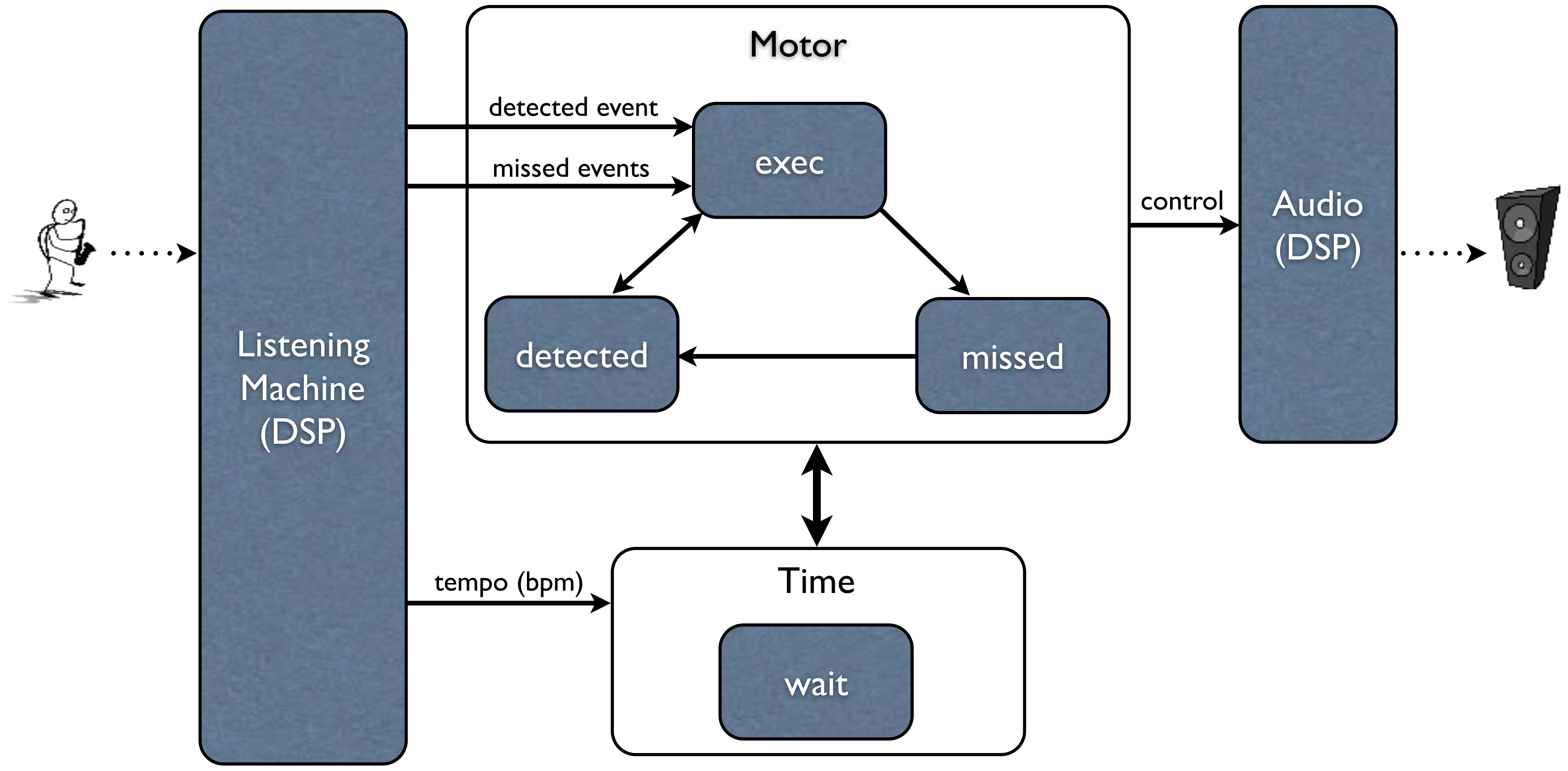
Definition: `signal <id>`
Emission: `emit <id>`
Waiting: `await <id>`

Broadcast communication between processes

Why ReactiveML?

- **A synchronous language**
  expressiveness for time and events
- **Functional, typed language, on top of OCaml**
  recursion and higher order processes
- **Efficient implementation**
  no busy waiting
- **Dynamical features**
  new interactions, live coding

Architecture



Execution of a score

$$\text{(Exec Score)}\ \frac{D \overset{exec}{\vdash} (\texttt{event}\ i\ t : seq) \rightarrow p_1 \qquad D \overset{exec}{\vdash} sc \Rightarrow p_2}{D \overset{exec}{\vdash} (\texttt{event}\ i\ t : seq)\ sc \Rightarrow p_1 \cup p_2}$$

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

Triggering

$$(\text{Detect})\quad \frac{i \in D \qquad D, i, 0.0 \xrightarrow[]{\mathit{detected}} \boldsymbol{seq} \Rightarrow p}{D \xrightarrow[]{\mathit{exec}} (\texttt{event}\ i\ t : \boldsymbol{seq}) \to p}$$

$$(\text{Miss})\quad \frac{i \notin D \qquad D, i, 0.0 \xrightarrow[]{\mathit{missed}} \boldsymbol{seq} \Rightarrow p}{D \xrightarrow[]{\mathit{exec}} (\texttt{event}\ i\ t : \boldsymbol{seq})\ sc \to p}$$

Triggering

$$\text{(Detect)}\ \frac{i \in D \qquad D, i, 0.0 \vdash^{detected} seq \Rightarrow p}{D \vdash^{exec} (\texttt{event}\ i\ t : seq) \rightarrow p}$$

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

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

- **Contributions**
    - a new semantics for Antescofo
    - a sequencer efficient enough to compare well with the actual one
    - prototyping new features:
      new attributes, reactive behaviors, live coding, ...
- **Next?**
    - interaction with other system: gesture follower, voice recognition, ...
    - link with synthesis tool or other media

To continue...

www.reactiveml.org/emsoft13

References

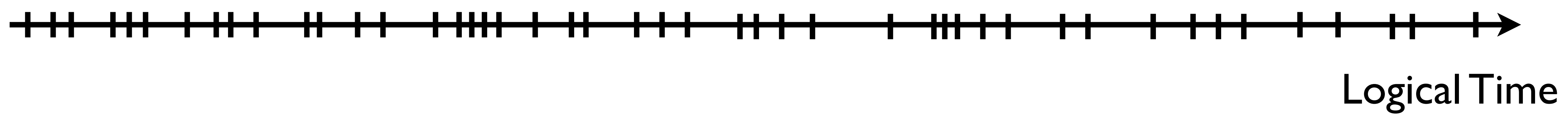
[Mandel-Pouzet 2005] L. Mandel and M. Pouzet. *ReactiveML: a reactive extension to ML.* In Proceedings of the International Conference on Principles and Practice of Declarative Programming, 2005.

[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.

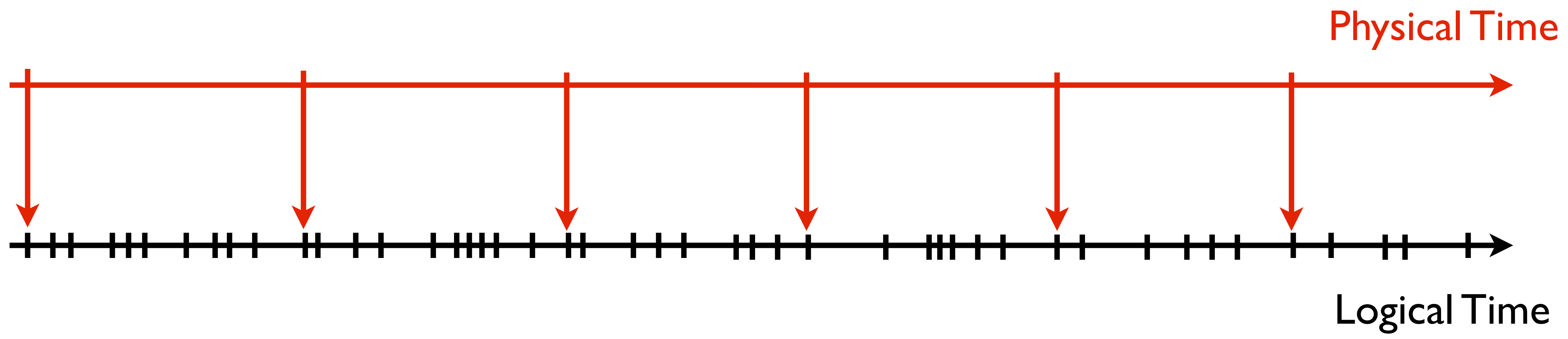
[Cont 2008] A. Cont. *Antescofo: Anticipatory synchronization and control of interactive parameters in computer music.* In International Computer Music Conference, 2008.

[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.

From Logical Time to Physical Time



From Logical Time to Physical Time



From Logical Time to Physical Time

```
let process tick period clock =
  let next = ref (Unix.gettimeofday () +. period) in
  loop
    let current = Unix.gettimeofday () in
    if (current >= !next)
    then (emit clock (); next := !next +. period);
    pause;
  end
```

```
val tick : float -> (unit, 'a) event -> unit process
```

Wait !

```
let process wait dur period clock =
  let d = int_of_float (dur /. period) in
  do
    for i=1 to d do pause done
  when clock done
```

```
val wait : float -> float -> ('a, 'b) event -> unit process
```

Detection

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

$$\text{(Detected Loose Group)}\quad \frac{D, i, \delta \vdash^{\mathit{detected}} \mathit{seq} \Rightarrow p}{D, i, \delta \vdash^{\mathit{detected}} \texttt{group loose}\ \mathit{err}\ \mathit{seq} \rightarrow p}$$

$$\text{(Detected Tight Group)}\quad \frac{D \vdash^{\mathit{exec}} \mathrm{Slice}(i, \delta, (\texttt{group tight}\ \mathit{err}\ \mathit{seq})) \rightarrow p}{D, i, \delta \vdash^{\mathit{detected}} \texttt{group tight}\ \mathit{err}\ \mathit{seq} \rightarrow p}$$

```
and process detected i delta ae =
  match ae with
  | Action(a) ->
      (* rule (Detected Action) *)
      run (wait date delta);
      emit perf (i,delta,a)
  | Group(g) ->
      begin match g.group_synchro with
      | Loose ->
           (* rule (Detected Loose Group) *)
           let bg = g.group_seq in
           run (exec_seq (detected i) delta bg)
      | Tight ->
          (* rule (Detected Tight Group) *)
          let gs = slice i delta g in
          run (exec gs)
      end
```