

Towards a Common Framework for Multimodal Generation

Stefan Kopp, Brigitte Krenn, Stacy Marsella, Andrew N. Marshall, Catherine Pelachaud, Hannes Pirker, Kristinn R. Thórisson, Hannes Vilhjálmsón

A.I. Group, Bielefeld University
Austrian Research Institute for AI, Vienna
IUT Montreuil, University de Paris 8
Information Sciences Institute, Los Angeles
CADIA Lab, Reykjavik University

Introduction

>10 years of generating multimodal communicative behavior with ECAs

- increasing number of multimodal agents
- increasingly sophisticated models for generating even single aspects of multimodal behavior

If you set out to build a multimodal agent, and you don't have 5 years to spend on re-implementing all these models, what do you do?

Let's enable people to better work together and to use each other's work, that has been directed to different aspects of multimodal behavior, with a minimal amount custom work.

A beginning history of working together

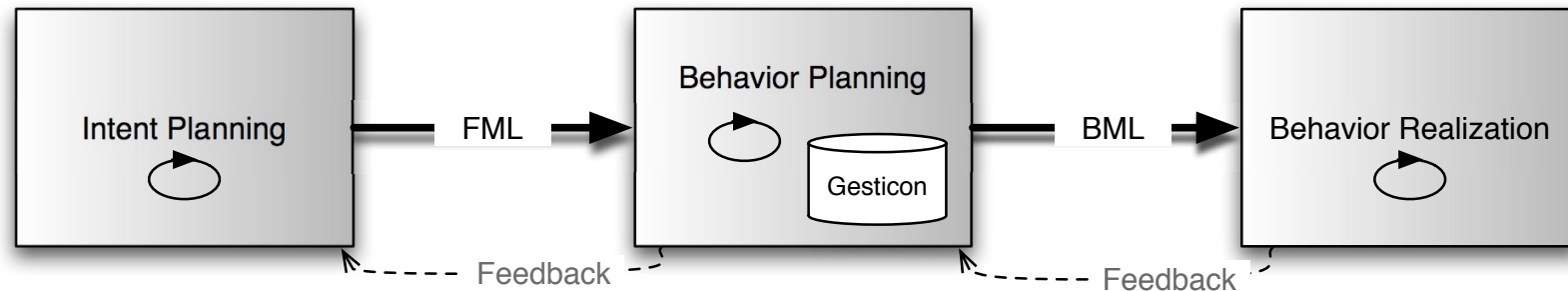
- AAMAS 2002 workshop “*Embodied conversational agents - let's specify and evaluate them!*“, „*Gesticon*“ workshop 2003, „*Representations for Multimodal Generation*“ workshop 2005
- Each of our systems represents a model of a generation process in which certain knowledge structures are identified and transformed: representations of communicative intent or affective state, representations of expressive behaviors, lexicon & rules to map them
- We propose to:
 - frame the problem of multimodal generation in a way that allows us to put it into computational models
 - demarcate planning & processing stages and identify the knowledge structures that mediate between them
 - render these stages and knowledge structures into a framework that lays down modules and interfaces

What's out there?

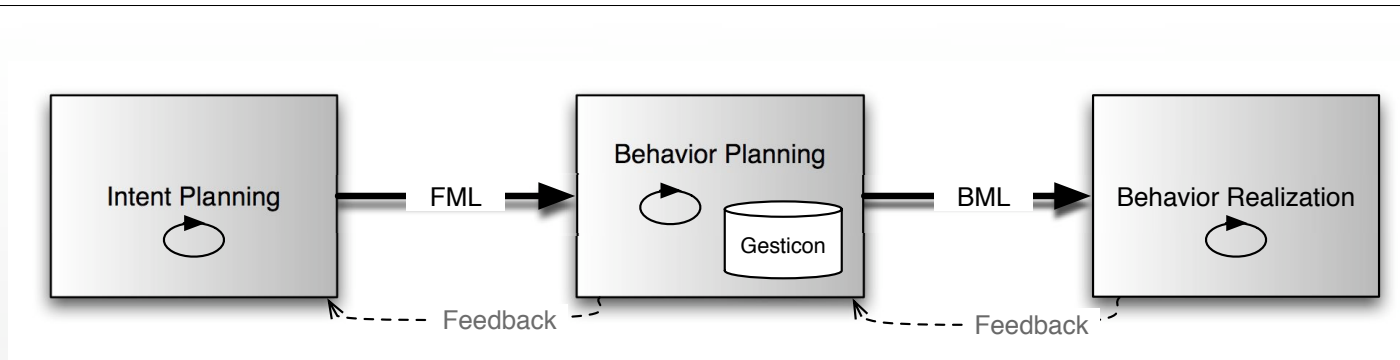
- Commonalities:
 - separation of *content*-related (domain-dependent) and *process*-related (domain-independent) issues
 - separation of *functional* and *behavioral* aspects
 - XML-based representations and processing pipelines that move from communicative intent to behavior selection to scheduling, to execution
- Main differences:
 - Level of abstraction and detail targeted
 - Specific ways of representing communicative intent, functions, behaviors, mapping rules, timings
 - Declarative vs. imperative (script-like) style
 - Agent-centered (cognitive models) vs. world-centered (control over everything)

The **SAIBA** framework

(Situation, Agent, Intention, Behavior, Animation)



- Macro-scale model of multimodal generation, consisting of processing stages on three levels of processing and abstraction
 - separate function-related features from behavior-related aspects
 - stages treated as black-boxes, no particular micro-architectures assumed
 - time-critical, incremental, highly flexible, bi-directional flow of information & control
- **Clear-cut definition of interfaces** between separate levels
 - maximally independent of specific domain, or player & graphics software used



- **FML -**

- Function Markup Language*

- description of the aspects that are relevant and influential in the planning of verbal and nonverbal behavior, without reference to overt behavior
 - basic semantic and pragmatic units
 - expressive, affective, discursive, epistemic functions/features

- **BML -**

- Behavior Markup Language*

- describes multimodal behaviors as they *are to be realized* in the final stage (i.e. in terms of constraints)
 - occurrence of behaviors
 - relative timings of behaviors
 - form of behaviors
 - must provide means of describing behaviors in a *player-independent* way
 - may be extended to *player-dependent* behavior specifications

BML Design

Independent of domain and implementation

- We cannot rely on particular skeleton joints, speech synthesis systems, available animations, etc.
- Instead, we refer to body parts and common verbs:
head, torso, face, lips, gaze, nod, speech, wait
- But we still also want to make nuanced behaviors available through optional parameters.

Design performances by composition

- Simple to conceptualize and author
- Allows modular behaviors in specific implementations, facilitating collaboration and reuse.

Temporal relations give behaviors context

- The order and synchronization of behaviors shapes the interpretation of the combined performance.
- The temporal parameters need to generalize to all behaviors.

BML's XML Structure

A BML message is a set of behaviors in a <bml> element

- Each behavior is represented by one child XML element.

```
<bml>
  <body posture="HandsAtSide" />

  <gesture type="BEAT" />
  <gesture type="POINT" target="bird002" />

  <head type="NOD" amount="0.2" repeats="2" />
  <head type="ORIENT" direction="LEFT" angle="20" />

  <face type="EYEBROWS" side="BOTH" shape="POINTDOWN" />

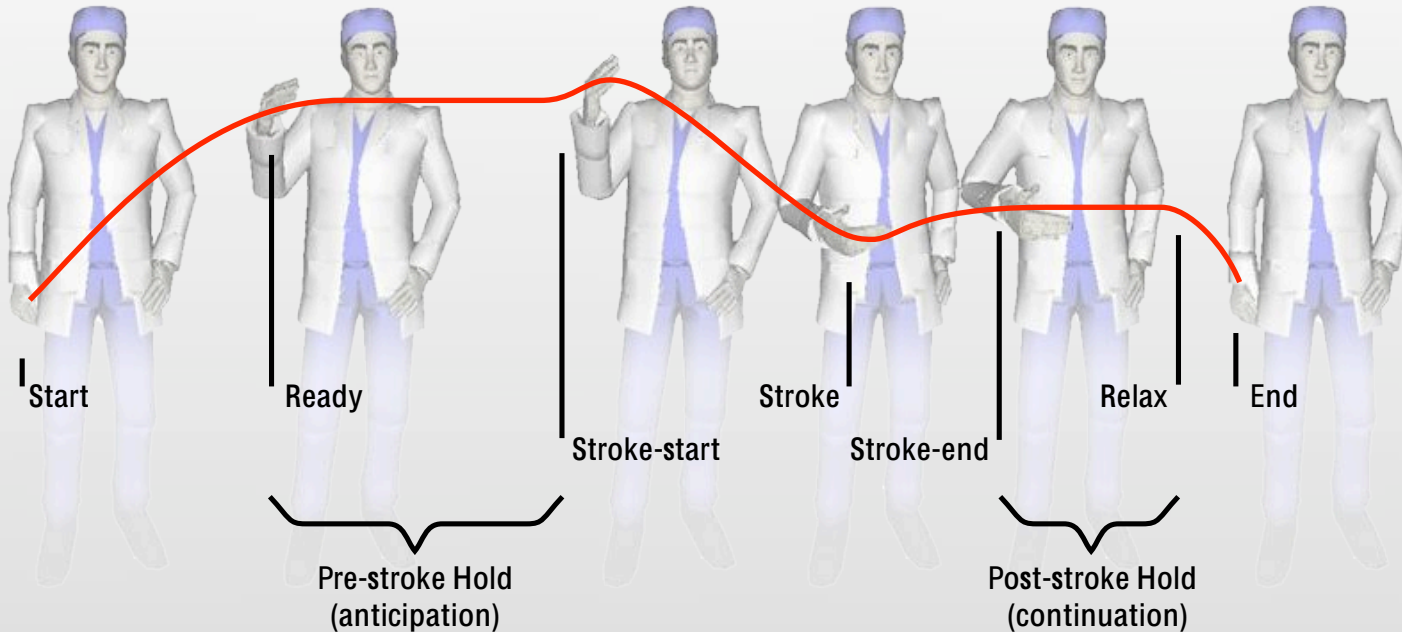
  <gaze target="Patient3" />
  <!-- Angle offset from target... -->
  <gaze target="Patient3" direction="LEFT" angle="20" />

  <speech>Something to synthesize.</speech>
</bml>
```

Synchronization Points

Every BML behavior has several synchronization points.

- Seven standard sync-points are defined as optional XML attributes:



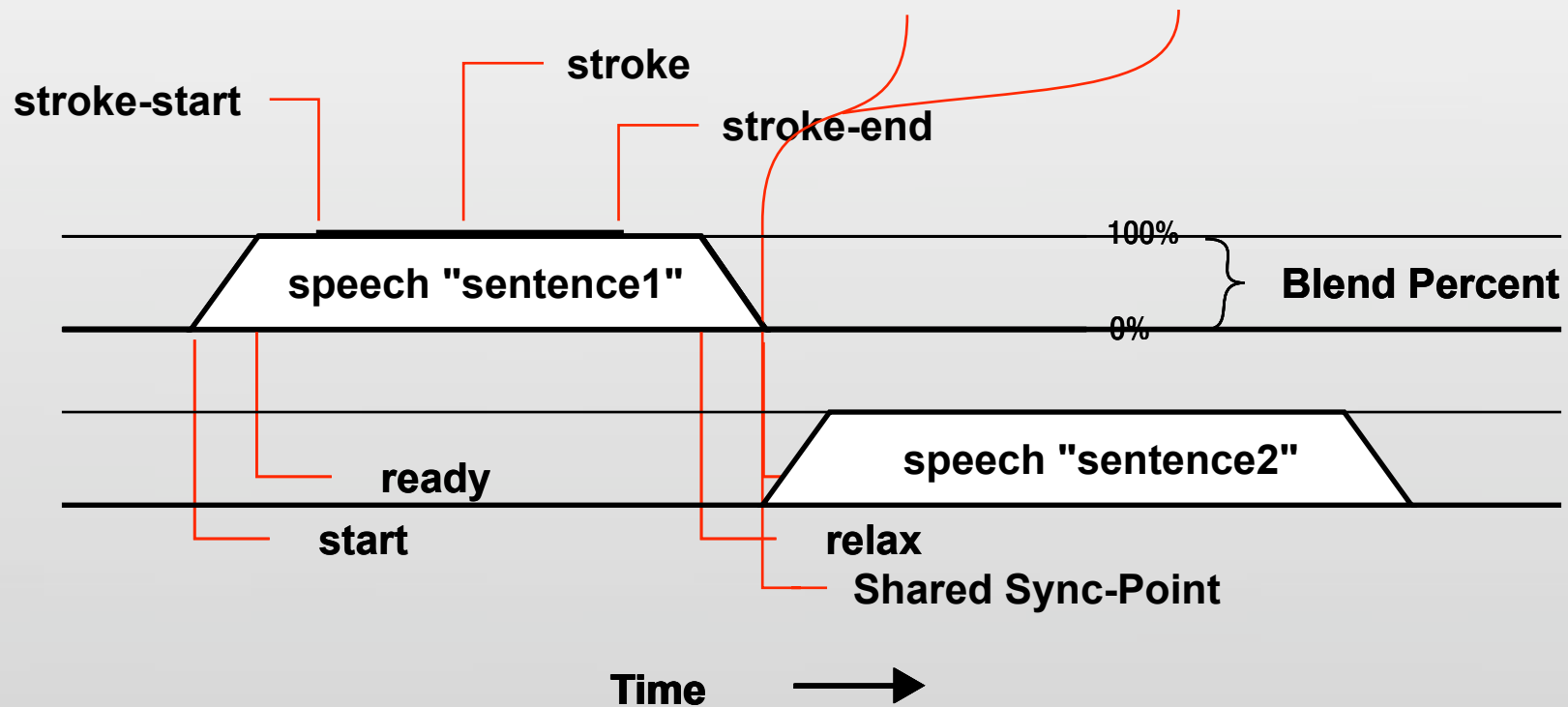
- Some behaviors can define additional synchronization points:

```
<speech>Example <sync id="wb"/> wordbreak.</speech>
```

Using Sync-Points

One behavior's sync-points can be described relative to the sync-points of other behaviors:

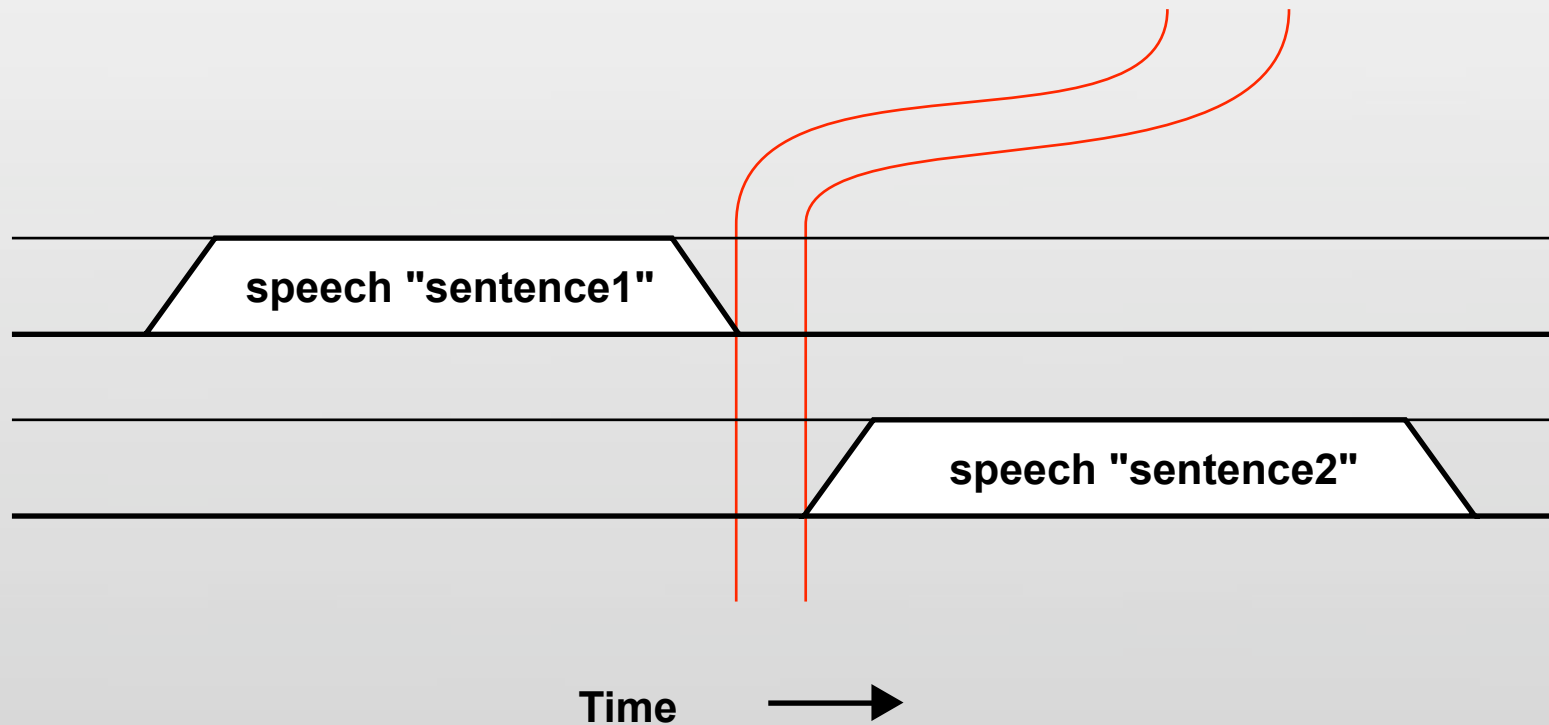
```
<speech id="sentence1" ... />  
<speech id="sentence2" ... start="sentence1:end" />
```



Using Sync-Points with Delays

Can specify a sync-point with offset from another sync-point

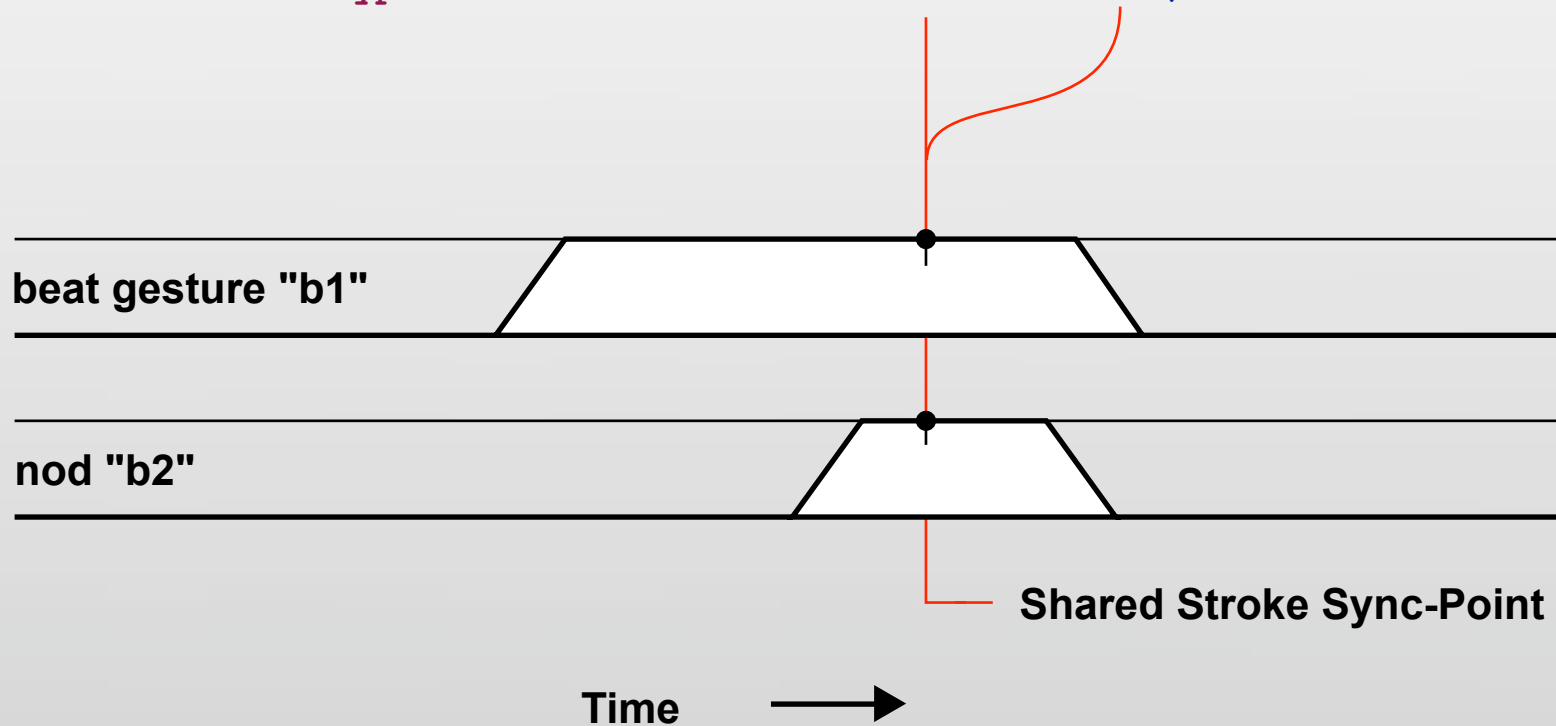
```
<speech id="sentence1" ... />  
<speech id="sentence2" ... start="sentence1:end + 0.5" />
```



Parallel Synchronized Behaviors

Behaviors can be synchronized on their inner stroke points without breaking behaviors into “pre-stroke” and “post-stroke”.

```
<gesture type="BEAT" id="b1" />  
<head type="NOD" id="b2" stroke="b1:stroke" />
```

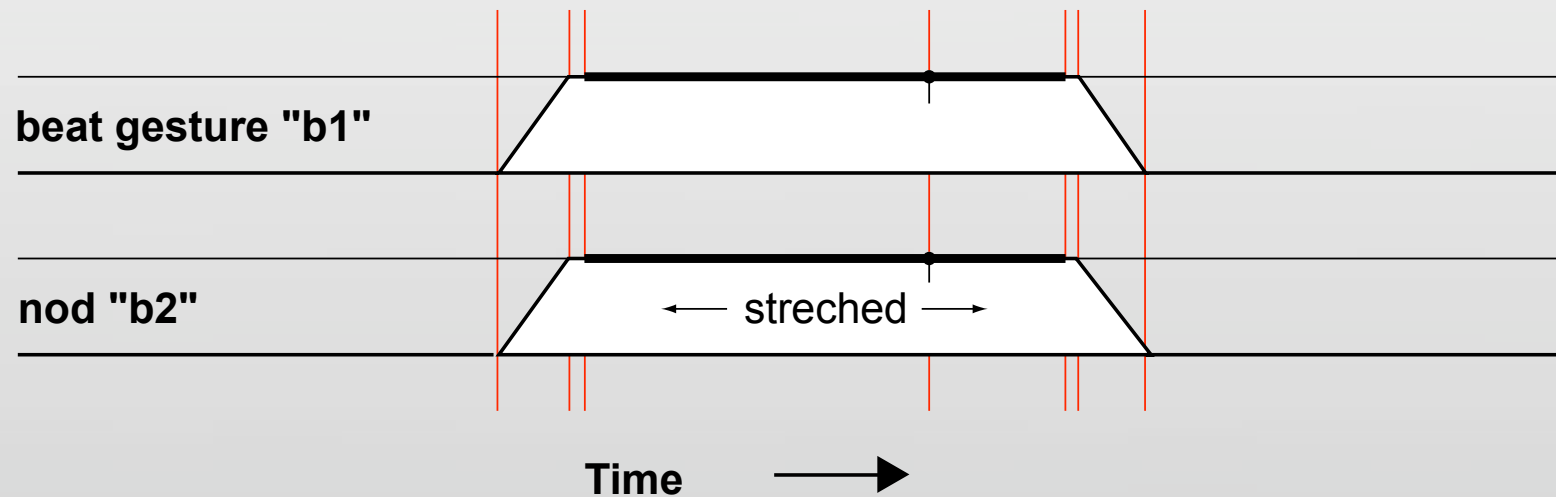


Multiple Sync-Points

Behaviors can be synchronized at multiple sync-points

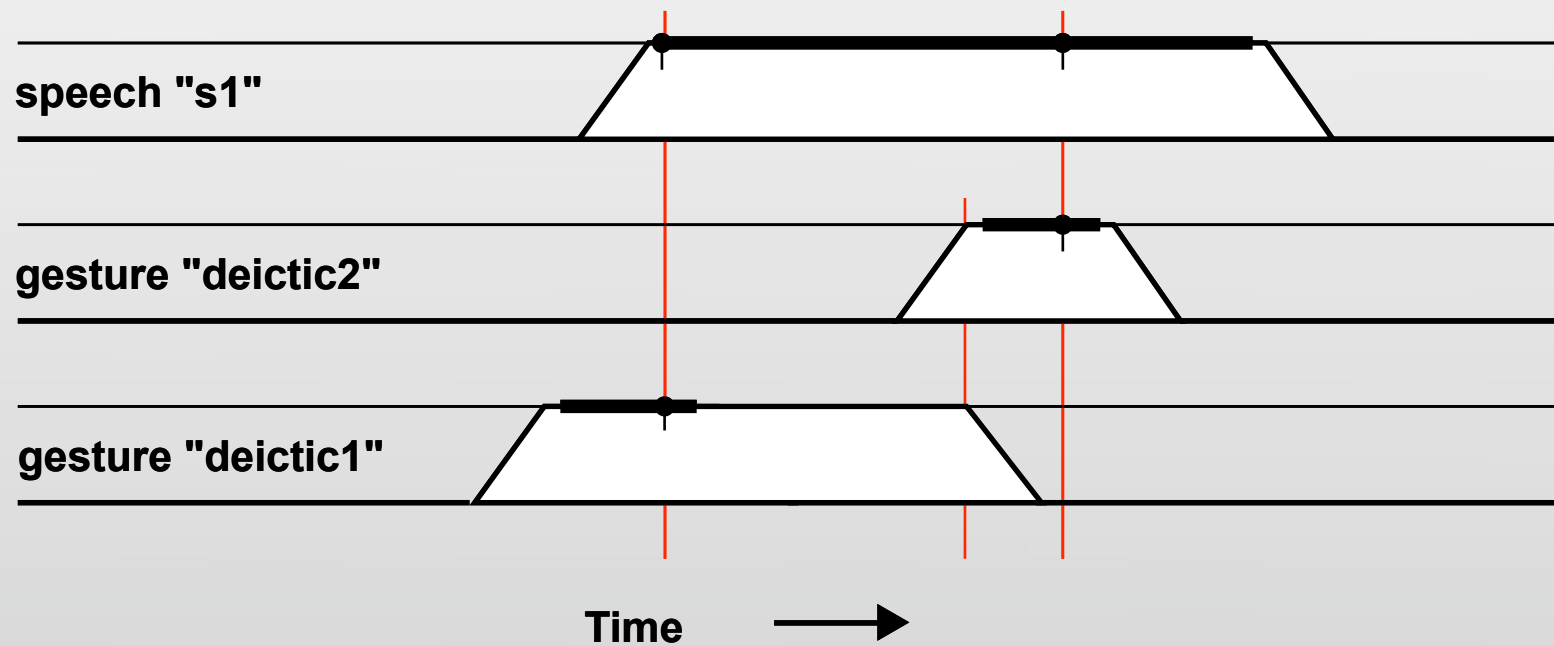
- Example: two behaviors that ready, stroke, and relax at shared sync-points, despite differing “natural” durations

```
<gesture type="beat" id="b1" />  
<head type="nod" id="b2" start="b1:start" ready="b1:ready"  
stroke-start="b1:stroke-start" stroke="b1:stroke" stroke-end="b1:stroke-end"  
relax="b1:relax" end="b1:end" />
```



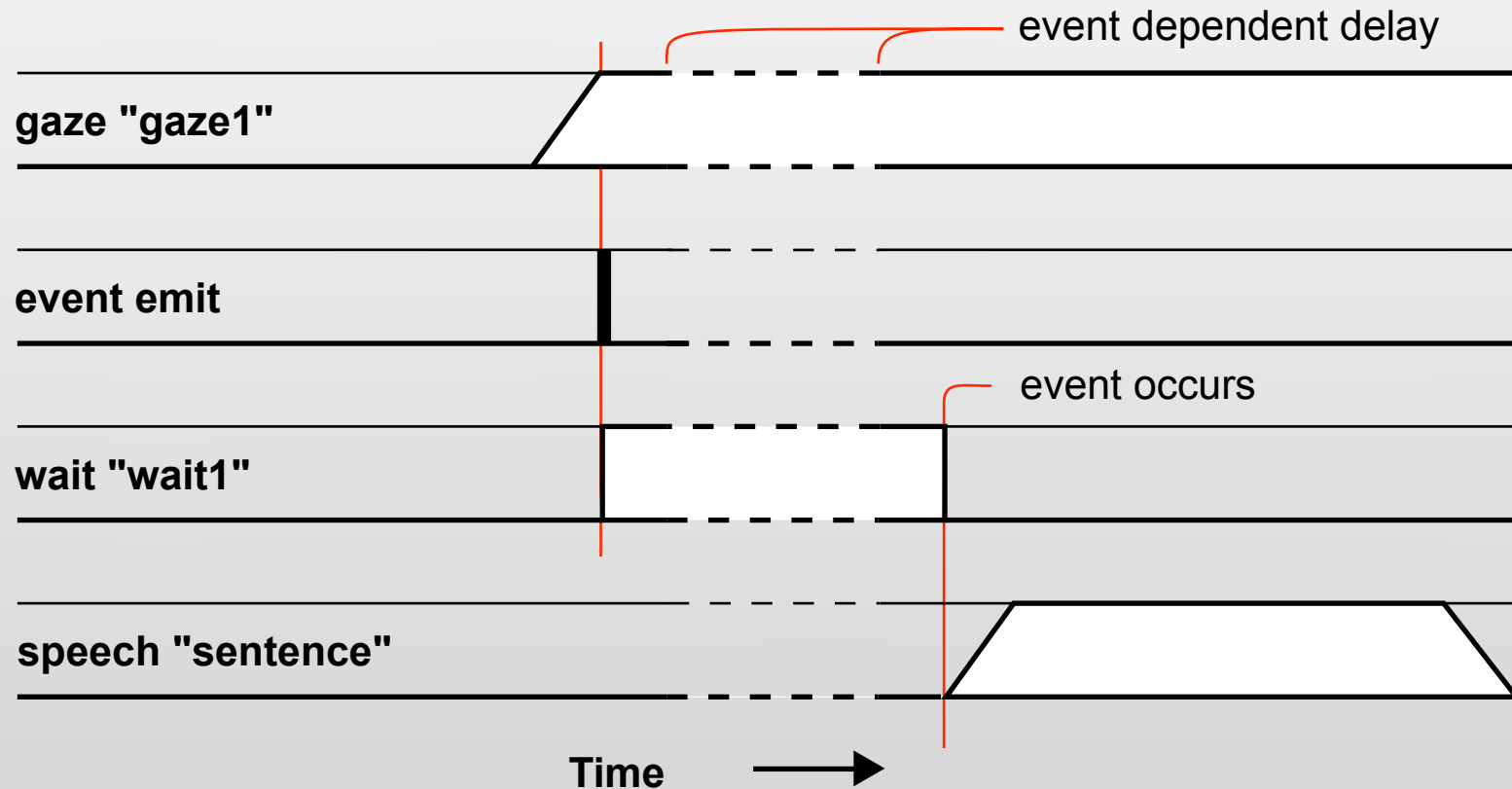
Timing Holds

```
<speech id="s1"><sync id="this"/>This or <sync id="that"/>that.</speech>  
<gesture id="d2" type="DEICTIC" ... stroke="s1:that" />  
<gesture id="d1" type="DEICTIC" ... stroke="s1:this" relax="d2:ready" />
```



Waiting and Events

```
<gaze id="gaze1" target="audience" />  
<event emit="speaker-gaze-at-audience" start="gaze1:ready" />  
<wait id="wait1" start="gaze1:ready" event="audience-return-gaze" />  
<speech id="sentence" ... start="wait1:end" />
```



How to use (contribute to) BML?

- Adopt and use "core BML" (frame structure, behavior elements, timing elements) and augment it with elements and attributes you need for your project or system
- Use XML *namespaces* to mark your proprietary additions
- Post at the WiKi (mindmakers.org) your „ BML dialect“ that extends the core parts by your project- or player-dependent parts
- Let the community discuss these suggestions and, eventually, decide which parts will become "core BML“

Extending BML

- New behaviors should be described elements using XML namespaces.
 - I.e., in SmartBody: `<me:controller name="interpolator4"/>`
 - Standards-only implementations should ignore the behavior in the act, unless the behavior includes the attribute `required="true"`, in which case the entire BML request should fail.
- Similarly, modifications of standard behaviors should use XML namespaced elements.
 - Allows a standards-only implementation to perform the BML by ignoring the non-standard attributes.
- For the standard behaviors with a `type="..."` attribute, new types can be introduced using namespace-like type prefixes.
 - I.e. `<face type="SBM:FACS" sbm:au="12" />`
 - Also, ignored unless the required attribute is true.

Future Work

- Flesh out the definitions of the standard behaviors of BML and the standard communicative functions of FML.
 - We're hoping people use and extend BML, and submit the additions they find useful as part of an ongoing discussion process.
- Design a standard language for upstream feedback about conflicts, interruptions, and constraints.
 - I.e., a behavior can't complete because the necessary body part is in use, or a world object causes a collision.
- Handle integration of behaviors from different <bml> requests.
 - Parameters for interruption, queuing, or merging of real-time acts.
- Support agent-world interactions, i.e. world effects from behaviors.
 - I.e., a "turn-knob" behavior may result in unlatching a door.