Using the Behavior Markup Language for Human-Robot Interaction

Aaron Holroyd and Charles Rich
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609
(aholroyd | rich)@wpi.edu

ABSTRACT

This paper describes a Behavior Markup Language (BML) realizer that we developed for use in our research on human-robot interaction. Existing BML realizers used with virtual agents are based on fixed-timing algorithms and because of that are not suitable for robotic applications. Our realizer uses an event-driven architecture, based on Petri nets, to guarantee the specified synchronization constraints in the presence of unpredictable variability in robot control systems. Our implementation is robot independent, open source and uses the Robot Operating System (ROS).

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Standardization

Keywords

BML, Realizer, Robots, Petri Net, Event-Driven, ROS

1. INTRODUCTION

In this work, we build upon research in the virtual agent community by using the Behavior Markup Language (BML) [3] to control robots interacting with humans. Many virtual agents, e.g. [5], have been controlled using BML. Using BML for robots will be just as important, since many robots need to interact with people in equally complex ways. This paper describes the benefits of applying BML to human-robot interactin and the problems we solved to make that possible.

Imagine a person ambiguously asking a robot to pick up a soda can when there are two cans on the table. The robot might respond by saying "This can?," while simultaneously looking and pointing toward one of the cans. These looking and pointing behaviors need to be synchronized correctly in relation to the speech for a natural, human-like performance.

```
<bml id="bml-example-1">
  <gaze id="gaze-1" target="can-1"/>
  <gesture id="point-1" type="POINT"
           target="can-1"/>
  <speech id="speech-1">
    <text>This can?</text>  </speech>
  <constraint id="constraint-1">
    <synchronize ref="gaze-1:stroke">
      <sync ref="point-1:start" />
    </synchronize>
    <synchronize ref="point-1:stroke_start">
      <sync ref="speech-1:start" />
    </synchronize>  </constraint>  </bml>
```

Figure 1: Example BML block

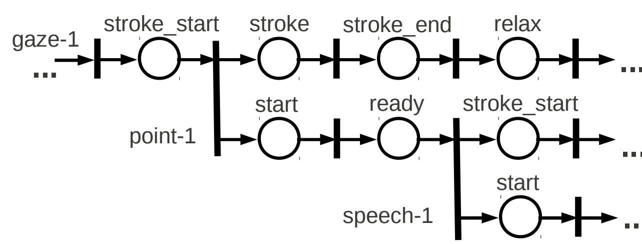

Figure 2: Part of the Petri net for Figure 1

The example BML in Figure 1 shows the behaviors involved in this example and the constraints between them. It contains a single gaze behavior (`gaze-1`) toward `can-1`, along with a pointing behavior (`point-1`). The speech behavior (`speech-1`) contains the text "This can?" to be uttered overlapping with the other behaviors. The constraints ensure that the pointing gesture does not start until the robot is looking at the object, and that the speech starts when the robot is beginning to point toward the object.

2. DIFFICULTIES WITH FIXED-TIMING REALIZERS FOR ROBOTS

BML was created by Kopp *et al.* [3], who identified the need for a common behavior specification framework for virtual agents. All BML realizers have two phases: scheduling and execution. In the scheduling phase, a shortest possible schedule that satisfies the constraints is created. The execution phase then uses this schedule to perform the behaviors in real time. Current animation realizers (see Figure 3), run open-loop using fixed-time schedules, which works because the realizer can generate key-frames. These realizers plan

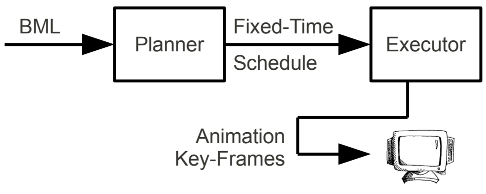

Figure 3: Animation BML realizer architecture

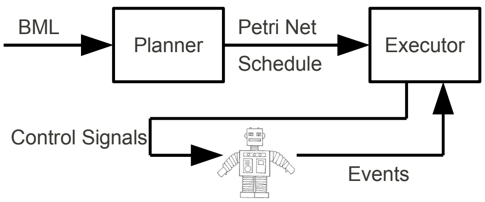

Figure 4: Robot BML realizer architecture

the BML using a fixed-timeline representation and generate key-frames based on the required joint angles. The key-frames are interpolated and shown to the user. Anh and Pelachaud [1] developed a fixed-timing realizer for a Nao robot based on the Greta animation engine. This system uses a database of motions and synchronizes them using fixed-timing constraints. They acknowledge the need for, but do not provide, an event-drive realizer for robots.

In an animation, since the time between each of the sync points can be *chosen*, fixed-timing algorithms are easy to implement. Also, in the animation setting, the timeline is guaranteed to execute properly. In robotics, however, there are several problems that cause fixed-timing approaches not to work.

The first, and most common, problem is when the motion of a joint motor is slower than expected, perhaps because it is stalled or a battery is partially drained. In this case the other motions need to be delayed and wait. For example, the realizer may need to wait for the arm to reach the correct position before starting to speak. A second example problem occurs when the robot's arm reaches its target position *earlier* than expected, perhaps because the arm was closer to the target than the robot believed.

3. EVENT-DRIVEN SOLUTION

A robotic realizer (see Figure 4) is fundamentally different from an animation realizer in that it must be controlled closed-loop, with control events being sent from the robot to the realizer. The control events are joint position (e.g., when a joint reaches the desired position) or velocity acknowledgements. These events suggest an event-driven system to ensure the synchronization constraints. The output of the scheduling stage in this approach is thus a Petri net [4] representing a minimum-time event-based schedule. We chose Petri nets because they conveniently represent events and their synchronization constraints.

Figure 2 shows part of the Petri net schedule resulting from Figure 1. A Petri net consists of places (represented as circles), transitions (represented as vertical bars), and tokens which are transmitted between places and transitions. In a Petri net BML schedule, each place represents a sync point of a behavior and the transitions represent synchronization constraints. Each transition and place waits for all incoming tokens and sends one token out on each arrow.

Our scheduling algorithm starts by creating a separate seven-place Petri net for each behavior that appears in the BML block. Figure 2 contains three such sub-nets, but none are completely shown due to limits on space.

Scheduling a `synchronize` constraint corresponds to merging the transitions prior to the given sync points. Note that this new merged transition replaces the prior two transitions, and synchronizes the sync points because transitions must wait for all incoming tokens before sending tokens out on all outputs.

Scheduling a `before` or `after` constraint is accomplished by adding a single arrow starting at the first place and ending at the transition prior to the second place. Note that the `after` constraint is the inverse of a `before`; thus without loss of generality the following two constraints are equivalent:

```
<before ref="X">            <after ref="Y">
    <sync ref="Y"/>             <sync ref="X"/>
</before>                   </after>
```

Our BML executor is implemented in ROS Java.[1] Starting from the left-most transition in Figure 2, the executor calls a robot-specific control ROS module for each sync point of each behavior. This control module sends joint commands to the robot and returns events when the given sync point is reached. Each transition and place is a new thread of execution that waits for all incoming tokens before executing. Similarly, the thread ends once all tokens have been sent on all of the output lines.

In conclusion, we have shown the theory and provided an open-source ROS implementation of an event-driven BML realizer for robots. We used this realizer in [2] and will continue to use it in future research.

Acknowledgements: This work is supported in part by the National Science Foundation under awards IIS-0811942 and IIS-1012083.

4. REFERENCES

[1] L. Q. Anh and C. Pelachaud. Expressive gesture model for storytelling humanoid agent. In *Fourth Workshop on Embodied Conversational Agents (WACA 2010)*, Lilles, France, November 2010.

[2] A. Holroyd, C. Rich, C. L. Sidner, and B. Ponsler. Generating connection events for human-robot collaboration. In *20th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2011*, Atlanta, GA, USA, Aug. 2011.

[3] S. Kopp, B. Krenn, S. Marsella, A. Marshall, C. Pelachaud, H. Pirker, K. Thórisson, and H. Vilhjálmsson. Towards a common framework for multimodal generation: The Behavior Markup Language. In *Intelligent Virtual Agents*, 2006.

[4] C. Petri. Fundamentals of a theory of asynchronous information flow. In *Information Processing: Proceedings of the IFIP Congress*. North-Holland, 1962.

[5] M. Thiebaux and S. Marsella. Smartbody: Behavior realization for embodied conversational agents. In *Intelligent Virtual Agents*, 2007.

[1] `http://sourceforge.net/projects/rosbmlrealizer/`