Realizing Multimodal Behavior

Closing the gap between behavior planning and embodied agent presentation

Michael Kipp, Alexis Heloir, Marc Schröder, and Patrick Gebhard

DFKI, Saarbrücken, Germany
firstname.surname@dfki.de

Abstract. Generating coordinated multimodal behavior for an embodied agent (speech, gesture, facial expression. . .) is challenging. It requires a high degree of animation control, in particular when reactive behaviors are required. We suggest to distinguish *realization planning*, where gesture and speech are processed symbolically using the behavior markup language (BML), and *presentation* which is controlled by a lower level animation language (EMBRScript). Reactive behaviors can bypass planning and directly control presentation. In this paper, we show how to define a behavior lexicon, how this lexicon relates to BML and how to resolve timing using formal constraint solvers. We conclude by demonstrating how to integrate reactive emotional behaviors.

1 Introduction

Embodied agents have the potential to make human-computer interaction more intuitive, engaging and accessible. To take full effect, they have to use all modalities of the body (speech, gesture, facial expression, posture etc.) in meaningful coordination. The SAIBA framework aims suggests three principal modules for the behavior production process [1]: *intent planning* for determining the overall content of the message in terms of abstract communicative goals, *behavior planning* for deciding on the choice of words, gestures, facial expressions etc. in terms of modalities but not body part and *realization* for rendering these behaviors with a concrete agent body and voice. BML (behavior markup language) was suggested as a standard language to formulate sequences of abstract behaviors (speech, gesture, gaze ...), independent of concrete animation or speech synthesis platforms. For instance, a gaze behavior toward a person X could be defined, independent of whether this is realized just with the eyes only, a turn of the head, the upper body or by repositioning the whole body. BML also allows to define complex temporal synchronization constraints between behaviors. However, there has been no general solution how to actually solve these in a principled way. We have argued that SAIBA leaves a significant gap between behavior planning and realization [2]. Many choices of actual animation must be either made in the realizer or specified using custom BML extensions, which compromises its conceptual clarity. Therefore, we suggest to separate these processes into *realization*

planning and *presentation* (Fig. 1). The realization planner converts abstract, underspecified and cross-referenced behavior specifications (BML) into linear, executable presentation scripts (so this includes multimodal coordination). The presentation module contains a 3D character animation engine and possibly a separate audio player for the voice. Presentation calls for a new language for the animation part which we call EMBRScript [3] which we also use to define the behavior lexicon.

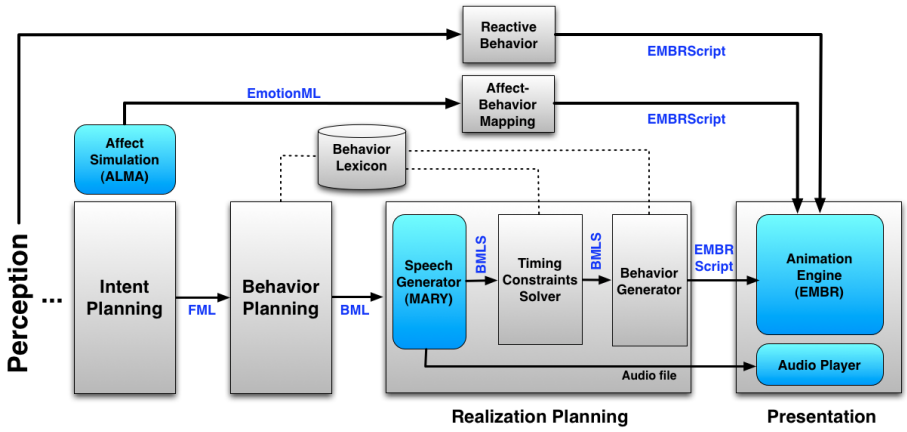

Fig. 1. Overview of our architecture. Reusable standard components depicted as light blue rounded boxes.

One of the earliest frameworks for virtual character control was Improv [4] which provided a scripting language for authoring both the animation and the behavior of real-time animated actors. This system was the first to make a distinction between abstract behavior specifications and executable animation scripts. However, the animation scripts were very low-level, an abstract framework on the level of behaviors was missing. In the BEAT [5] pipeline architecture a translation mechanism from behavior specifications to animation commands exists but the translation process is not explicitly described. SmartBody was the first system to implement BML for agent control [6] but required many custom extensions to BML. Greta [7] is also based on SAIBA and separates realization into behavior planning and realization in the sense that MPEG4 scripts for the player are produced. However, it is not clear how the translation process is done in terms of e.g. temporal constraints resolution. While most existing frameworks implicitly make the distinction between realization planning and presentation, they do not offer declarative languages to control animation and/or define behavior lexemes. The problem of time constraint resolution has not been dealt with in a principled way.

Our research makes the following contributions: (1) Separating realization planning from presentation, (2) clarifying the role of behavior lexicons, (3) a principled approach to resolving BML time constraints and (4) demonstrating reactive behavior integration.

2 Framework

The pipeline architecture of our framework is shown in Fig. 1. We focus on realization planning where processing starts with a BML[1] document that describes, e.g., the text to be spoken, gestures, head nods etc. – together with timing constraints regarding the behaviors' sync points. Realization planning consists of generating speech, resolving all behavior timings and retrieving lexeme data for the production of an executable animation script, together with an audio file of the speech. The presentation layer represents character animation and audio playback. We can now issue reactive behaviors like gaze following or continuous adjustments to the current emotion (face, posture) by sending it directly to the presentation module, by-passing intent and behavior planners.

Components Our system is implemented on top of the SEMAINE API, an open-source distributed multi-platform component integration framework for real-time interactive systems [8]. Components communicate via standard representation formats, including BML, BMLS and EmotionML. For realtime 3D character animation we use our free EMBR software[2] (Embodied Agents Realizer) [3, 2] which allows fine-grained animation control via a pose-based specification language called EMBRScript. EMBR allows gesturing, facial expressions, pose shifts, blushing and gaze control and autonomous behaviors like breathing and blinking. For the synthesis of speech we use the *text-to-speech* and *speech realizer* components MARY of the SEMAINE system.

Reactive Affect-Behavior Mapping To demonstrate the modeling of reactive behavior, we implemented a direct coupling of emotional state with behaviors like facial expression, head orientation, breathing and blushing. We use ALMA [9] for affect simulation which continually produces EmotionML [10] to express the current *emotion* and *mood* of the agent. This is mapped to behaviors that are directly sent to the presentation component as EMBRScript documents. For instance, *mood* affects blushing and breathing with its *arousal* component. We defined mappings for emotion events (joy, anger, liking...) to trigger behaviors (facial expressions) and physiological changes (breathing, blushing), based on the literature. The emotion intensity determines the duration and the visible intensity (e.g. blushing level, extent of smile, raised eye brow level) of an emotional cue.

[1] http://wiki.mindmakers.org/projects:bml:draft1.0
[2] http://embots.dfki.de/EMBR

3 Realization Planning

Realization planning input is a behavior specification (BML) to be translated into an executable animation script (EMBRScript). We propose a pipeline where the input BML is successively enriched. We call "solved BML", short BMLS, a subset of BML where all time references are resolved.

Behavior Lexicon Behaviors specified in BML have to be translated to animation plans. Synchronization points must be resolved for each behavior. Both tasks require a central database of behaviors, the *behavior lexicon* L, which effectively restricts the language BML to a subset BML(L).

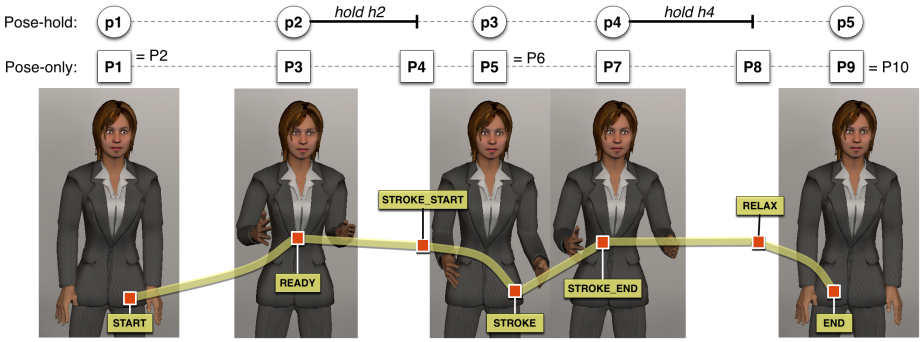

Fig. 2. EMBRScript template for lexeme "beats" with parameter handedness=both. Only the poses P3 ("ready"), P5 ("stroke") and P7 ("stroke_end") need markup, the rest is implicit. The top line shows the pose-hold representation of EMBRScript the line below the pose-only representation.

The behavior lexicon L consists of a collection of *lexemes* $\{l_1, \ldots, l_n\}$. Lexemes are *parametrizable* with regard to BML attributes (e.g. handedness, number of repetitions), i.e. every lexeme represents a *set of variants*. Each lexeme must declare which of the BML *sync points* (start, ready, stroke_start, stroke, stroke_end, relax, end) can be used in conjunction with this lexeme. In our framework, we use EMBRScript to define the variants of a lexeme which are based on key poses, some of which can be equated with BML sync points (see Fig. 2). The key pose representation allows to easily modify the motion by shifting poses in time or adjusting hold durations. For every lexeme we define several EMBRScript templates that correspond to a concrete parametrization, e.g. three templates for handedness variants: left, right or both. We mark up those poses that correspond to sync points. Because of the pose-hold representation, we only need to mark up three sync points at most (namely ready, stroke and stroke_end). We can quickly create lexeme templates using a graphical tool for pose creation, sync point markup and interactive animation viewing. We currently have 51 gesture lexemes in our lexicon, instantiated by 121 EMBRScript templates.

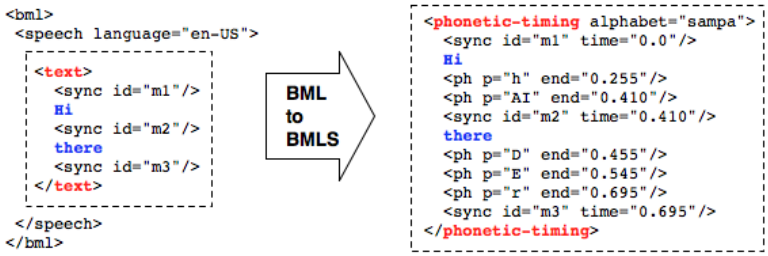

Fig. 3. BML transformed to BMLS$_{speech}$: phoneme timings and sync points resolved.

Speech Generator The text-to-speech (TTS) component reads BML markup and sends audio data to the presentation module (Fig. 1). For temporal synchronization with the face and gestures, the TTS needs to output (a) the phonetic symbols and their timings and (b) the absolute time of BML sync points. We transport this information under the new `phonetic-timing` tag. Note that this tag can also be used in the case that an application uses pre-recorded audio files instead of synthesized speech. BML is transformed to BMLS$_{speech}$ as depicted in Fig. 3, where the `text` tag is replaced by the more informative `phonetic-timing` tag, effectively decoupling the steps of audio generation and synchronized visual realization planning.

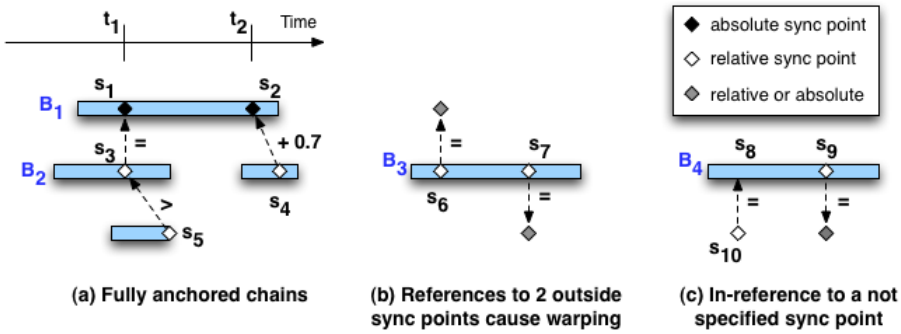

Fig. 4. Cases of sync point linkages.

Timing Constraints Solver The timing constraints solver receives BMLS$_{speech}$ where speech sync points contain absolute time. The task is to resolve relative sync points also for nonverbal behaviors like in <`gesture start="h1:end"`/ >. We translate this problem to a formal constraint problem and explain this by

example. Fig. 4 shows behaviors and how their sync points can relate to each other. Case (a) is probably most frequent: sync points refer to other sync points, and every "chain" of references is grounded in an absolute sync point. For the constraint problem formulation we introduce V^s for every sync point s. Absolute times can be directly assigned: $V^{s1} = t_1$ and $V^{s2} = t_2$. Relative sync point relationships are expressed via their respective constraint variables. For the example in Fig. 4 this translates to $V^{s3} = V^{s1}$ and $V^{s4} = V^{s2} + 0.7$ and $V^{s5} > V^{s3}$.

Fig. 4, case (b), a behavior B_3 has two (or more) sync points s_6 and s_7 which means B_3 has to be stretched or compressed. In case (c), behavior B_4 *is referenced* by another behavior with respect to sync point s_8, although s_8 is *not* specified at all in B_4. This makes sense only if there is another sync point s_9 that "anchors" behavior B_4. However, how do we formulate a constraint for resolving s_{10}? For this, we have to consult our lexicon and introduce s_8 as a virtual sync point, deriving its relation to s_9 from the lexeme template. In the lexicon, each behavior has a pre-specified timing of sync points. We formulate constraints thus that the changes to the original lexeme are kept minimal. So let us assume that of behavior B_4, one sync point s_9 is specified and another one, s_8, is "virtual", i.e. it is referred to but not specified itself. Then, we introduce intra-behavior constraints to express the relationship between s_8 and s_9: $V^{s8} + V^{\Delta} = V^{s9}$ and $V^{\epsilon} = |lexeme_{\Delta}(s_8, s_9) - V^{\Delta}|$. Here, $lexeme_{\Delta}$ is a constant that is equal to the distance of the two sync points in the original lexeme template. V^{Δ} is the actual distance when realized and V^{ϵ} is the deviation from the original distance. In our constraint solver, we ask for minimization of all V^{ϵ} variables. We implemented the described system with the Java constraint solver JaCoP [3].

Note that we can now define the *realizability* of an BML(L) input document in terms of three conditions: (1) every sync point of behavior b must be marked-up in the lexeme of b in lexicon L, (2) every chain of relative sync points must be grounded, i.e. end in an absolute sync point[4] and (3) every lexeme reference in BML must have a counterpart in L.

Behavior Generator The behavior generator receives BMLS and produces final EMBRScript. For viseme generation, the phonetic mark-up is translated to a sequence of morph targets (currently 16 visemes). For gesture generation, each BMLS behavior tag B that refers to a lexeme in the behavior lexicon is looked up based on the parameters (e.g. handedness) to retrieve the correct lexeme variant in the form of an EMBRScript template e. This template is then modified according the the resolved sync points in B.

4 Conclusion

We presented a framework for realizing multimodal behaviors for an embodied agent. Our framework is based on the modified SAIBA framework where realiza-

[3] http://jacop.osolpro.com

[4] Here, we need an extended definition of a "chain" where, if sync point s refers to behavior b, the only condition is that b has at least one other sync point.

tion and presentation are separated, and utilizes existing and coming standards for data exchange (BML and EmotionML). We showed how to resolve timing constraints in BML and how to expand gesture lexemes to executable scripts. We demonstrated the potential for reactive behaviors by implementing a direct affect-behavior mapping. Our work clarifies that BML is tightly linked to the behavior lexicon. The *realizability* of a BML document depends internally on the linking of relative constraints and externally on the lexicon entries (lexemes) and the semantic meta-data of each lexeme. We advocate to use standardized XML languages for data exchange, even within modules, in the form of "solved BML" (BMLS), to facilitate the exchange of components in the community.

In future research we plan to integrate a dialogue manager and rich sensor input, e.g. tracking gaze, facial expressions and biosignals. These signals can directly be mapped to presentation for reactive behaviors like gaze following.

Acknowledgments. This research has been carried out within the framework of the Excellence Cluster Multimodal Computing and Interaction (MMCI), sponsored by the German Research Foundation (DFG).

References

1. Vilhjalmsson, H., Cantelmo, N., Cassell, J., Chafai, N.E., Kipp, M., Kopp, S., Mancini, M., Marsella, S., Marshall, A.N., Pelachaud, C., Ruttkay, Z., Thórisson, K.R., van Welbergen, H., van der Werf, R.J.: The behavior markup language: Recent developments and challenges. In: Proc. of Intelligent Virtual Agents. (2007)
2. Heloir, A., Kipp, M.: EMBR - a realtime animation engine for interactive embodied agents. In: Proc. of the Intl. Conf. on Intelligent Virtual Agents. (2009)
3. Heloir, A., Kipp, M.: Realtime animation of interactive agents: Specification and realization. Applied Artificial Intelligence (2010)
4. Perlin, K., Goldberg, A.: Improv: A System for Scripting Interactive Actors in Virtual Worlds. Proc. of SIGGRAPH '96 **29**(3) (1996)
5. Cassell, J., Vilhjálmsson, H., Bickmore, T.: BEAT: the Behavior Expression Animation Toolkit. In: Proceedings of SIGGRAPH 2001. (2001) 477–486
6. Thiebaux, M., Marshall, A., Marsella, S., Kallman, M.: Smartbody: Behavior realization for embodied conversational agents. In: Proc. of the Intl. Conf. on Autonomous Agents and Multiagent Systems. (2008)
7. Niewiadomski, R., Bevacqua, E., Mancini, M., Pelachaud, C.: Greta: an interactive expressive ECA system. In: AAMAS '09: Proc. of The 8th International Conference on Autonomous Agents and Multiagent Systems, Richland, SC (2009) 1399–1400
8. Schröder, M.: The SEMAINE API: towards a standards-based framework for building emotion-oriented systems. Advances in Human-Computer Interaction (2010)
9. Gebhard, P.: ALMA - a layered model of affect. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, ACM Press (June 2005) 29–36
10. Schröder, M., Baggia, P., Burkhardt, F., Pelachaud, C., Peter, C., Zovato, E.: Emotion markup language (EmotionML) 1.0. W3C first public working draft, World Wide Web Consortium (October 2009)