Multimodal Continuation-style Architectures for Human-Robot Interaction

Nikhil Krishnaswamy NKRISHNA@BRANDEIS.EDU
James Pustejovsky JAMESP@BRANDEIS.EDU
Department of Computer Science, Brandeis University, Waltham, MA 02148 USA

Abstract

We present an architecture for integrating real-time, multimodal input into a computational agent's contextual model. Using a human-avatar interaction in a virtual world, we treat aligned gesture and speech as an *ensemble* where content may be communicated by either modality. With a modified nondeterministic pushdown automaton architecture, the computer system: (1) consumes input incrementally using continuation-passing style until it achieves sufficient understanding the user's aim; (2) constructs and asks questions where necessary using established contextual information; and (3) maintains track of prior discourse items using multimodal cues. This type of architecture supports special cases of pushdown and finite state automata as well as integrating outputs from machine learning models. We present examples of this architecture's use in multimodal one-shot learning interactions of novel gestures and live action composition.

1. Introduction

Unlike interaction with other types of interactive agents (e.g., chatbots or personal digital assistants), human-robot interaction inherently requires multi-modality. Robotic agents are *embodied* and *situated* which affords robots the ability to affect the real world, but also requires them to have accurate and robust interpretive capabilities for multiple *input modalities*, which must run in real time. In addition, a robot must be able to communicate with its human interlocutors using all *communicative modalities* humans may use, including natural language, body language, gesture, demonstrated action, emotional cues, etc. As robots take on more human-like appearances, this becomes even more important, as there exists a gulf between expectations that the robot will communicate and understand things in a human-like way and its actual multimodal capability (Luger & Sellen, 2016).

Computers as collaborators (a la social robotics) require architectures that enable communication with humans naturalistically and multimodally, as humans do with each other. These architectures must be able to capture not only natural language (spoken, written, sign, etc.), but also gesture and body language, dynamic discourse semantics (Asher, 1998), affect and emotion (Scheutz et al., 2006), etc., all in context. Context, in the case of a human-robot interaction, comes in large part from the relative embodiment of the human and the robot or agent, and their situatedness with respect both to the scene that they both inhabit and to each other ("*co-situatedness*" (Pustejovsky et al., 2017)). But situatedness in the virtual world or the agent's internal model alone is not enough: the agent must be able to situate and model itself in the physical world of its interlocutors and interpret contextualized input relative to that space (Pustejovsky & Krishnaswamy, 2019). This in turn entails that the agent be *socially situated* and *socially embedded*, following Dautenhahn et al. (2002) and a multimodal interface is at minimum a *social interface*, following Breazeal (2003).

A number of architectures exist that handle at least some of the technical requirements of these kinds of social robots or agents, from natural language processing to motor control (Jaimes & Sebe (2007) and Goodrich et al. (2008) provide surveys of HRI and HCI, including architectures, as of time of publication). We present a computational framework for capturing and reasoning over conversational and situational context, and driving the agent's actions or responses in the world. A number of different particular reasoning architectures can be created using this framework, which is built on top of the VoxML modeling language (Pustejovsky & Krishnaswamy, 2016) as its representation of object and event semantics. It exploits continuation-passing style (van Wijngaarden, 1966; Reynolds, 1993; Van Eijck & Unger, 2010) to retrieve situational contextual information and compose it with object and event properties to conduct reasoning at runtime, and is designed modularly to facilitate integration with other robotic architectures, such as DIARC (Scheutz et al., 2007), POMDP approaches (Zhang et al., 2017), or reinforcement learning (Peters & Schaal, 2008).

Since the architectures described here are currently deployed on systems using an agent in a virtual world, but are being developed for use in interactions with physical robot, we will use the term *human-avatar interaction* (HAI) for interaction between a human and an *embodied, situated agent*, be it an animated avatar in a virtual world or a robotic agent in the physical world.

2. Interactive and Formal Structure

Assume a task-oriented HAI dialogue reproducing most conventions of human-to-human task-oriented dialogue (e.g., cooperation, responsiveness, disambiguation, etc.). Interlocutors might refer to objects and actions in any order, i.e., a single utterance: “put the knife in the blue cup"; or a multi-step dialogue specifying entities and actions involved. Formally, this requires that arguments be applied to previously cached predicates and vice versa at runtime, and in turn requires a representation capable of executing methods where arguments can be raised to the type required.

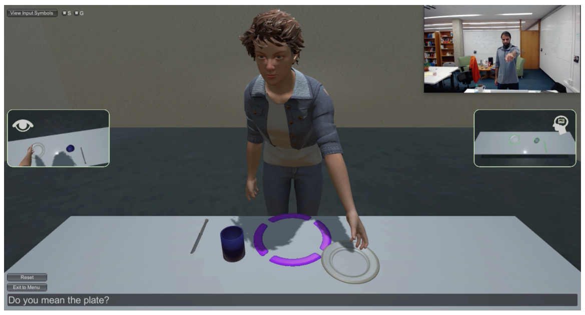
2.1 Interactive Structure

Implementationally, the system in which we deploy our multimodal architectures is build on the VoxSim software (Krishnaswamy & Pustejovsky, 2016), itself build on the VoxML platform and modeling language. VoxSim consumes input from real-time gesture and speech recognition clients, as described in Krishnaswamy et al. (2017) and Narayana et al. (2018), and conducts real-time inference, reasoning, and disambiguation using a visualized and situated simulated environment as the computer's representation of the shared world between agent and human.

Within this environment, consider two dialogues—one conducted entirely with language and one with a combination of language and gesture (Fig. 1). Dialogue 1L is conducted using only language input, and in this example language suffices to give the directions necessary. However, in many cases, HAI may require the ability to indicate information using a different method. For instance, direct grounding to a location may be needed because the location is too complicated to describe in language, or simply for efficiency:

HUMAN: “The plate." [A]	HUMAN: “The plate." [A]
AGENT: [AGENT *reaches for plate*] “Okay, go on."	AGENT: [AGENT *reaches for plate*] “Okay, go on."
HUMAN: “Put it in front of you." [B]	HUMAN: [HUMAN *points*] “Put it there." [B]
AGENT: [AGENT *puts plate in front of itself*] “Okay."	AGENT: [AGENT *puts plate at indicated location*] “Okay."

Figure 1. Dialogues—using only language (L) and language with gesture (R)

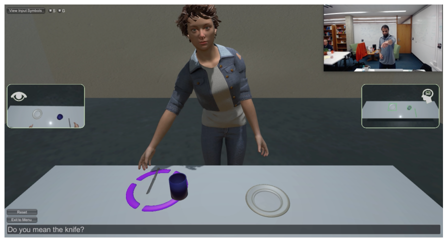


Do you mean the plate?

$$\begin{bmatrix} \textbf{point} & \\ \text{TYPE} = & \begin{bmatrix} \text{HEAD} = \textbf{assignment} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:agent} \\ \text{A}_2 = \textbf{y:finger} \\ \text{A}_3 = \textbf{z:location} \\ \text{A}_4 = \textbf{w:physobj}\bullet \\ \qquad\ \ \textbf{location} \end{bmatrix} \\ \text{BODY} = \begin{bmatrix} \text{E}_1 = extend(x, y) \\ \text{E}_2 = def(vec(\\ \qquad x \rightarrow y \times z), \\ \qquad as(w)) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 2. L: Example multimodal interaction accompanying Dialogue 1R. Upper-right inset shows human pointing to location; purple target in main image shows interpreted location of deixis in virtual world. R: VoxML typing of [[POINT]] (Pustejovsky & Krishnaswamy, 2016). E_2 defines the target of deixis as the intersection of the vector extended in E_1 with a location, and reifies that point as a variable w. A_4, shows the compound binding of w to the indicated region and objects within that region Ballard et al. (1997).

In both dialogues, the human specifies the object to be manipulated in [A] and then specifies a location in [B] along with an action. In Dialogue 1R, the human's use of a demonstrative word, “there" is accompanied by a deictic gesture (Fig. 2), which grounds the demonstrative to a specific location or objects at that location. “There" selects for the location indicated by deixis, not an object.



Do you mean the knife?

Figure 3. Deixis to region with objects.

The human can indicate object properties with accompanying deixis to single out objects in a region subject to particular properties. Thus given a deictic gesture as in Fig. 3, “cup," “that cup", “the blue cup," or even “in that blue cup" or “put the knife in the blue cup" single out the same object in the region, specifying objects, locations, or actions using multiple modalities.

2.2 Formal Structure

Formally, we can define the vocabulary of the interaction, consisting of sequences of “moves" by both interlocutors (as defined in Krishnaswamy & Pustejovsky (2018); Pustejovsky (2018)), using the format of a context-free grammar (CFG) where the nonterminals represent sets of particular input symbols and the terminals represent the particular content or intended response communicated by those input symbols. We can define a portion of the interactive "grammar" as follows, where unexpanded terminals represent parsed sentences or phrases or variations on individual gestures:

Grammar	**Legend**	
$S \rightarrow OA \vert AO$	O: define object	ω: static iconic gesture (object)
$O \rightarrow \delta \vert \delta D \vert \omega \vert \omega D \vert N \vert ND$	A: define action	α: dynamic iconic gesture (action)
$A \rightarrow \alpha \vert \alpha D \vert V \vert VD \vert P \vert PD$	D: disambiguate	δ: deictic gesture
$D \rightarrow \delta \vert \delta D \vert P \vert PD \vert N \vert ND \vert y \vert yD \vert n \vert nD$	V: verb phrase	y: affirmative response
	N: noun phrase	n: negative response
	P: prep. phrase	

Table 1. Interactive grammar snippet

At each step the disambiguation symbol D represents the acquisition of information the agent still needs to know in order to complete some action initiated or requested by the human. Furthermore, the order of instructions may vary, requiring that the agent be able to hold previously acquired information "in reserve" pending further instruction or answers to disambiguatory questions.

Due to the large number of terminal symbols in the "grammar" in even a superficial system, creating new states for every possible contextual configuration is computationally inefficient. For instance, if there are three objects in the scene, an object disambiguation sequence should not have to proceed through three distinct states, waiting for a *yes* or *no* each time, to confirm which of the three objects should be the focus, when instead it can recurse through the same state with a different argument until *yes* is received, and then proceed to the handling of the argument, or proceed to another state when all possible arguments are exhausted.

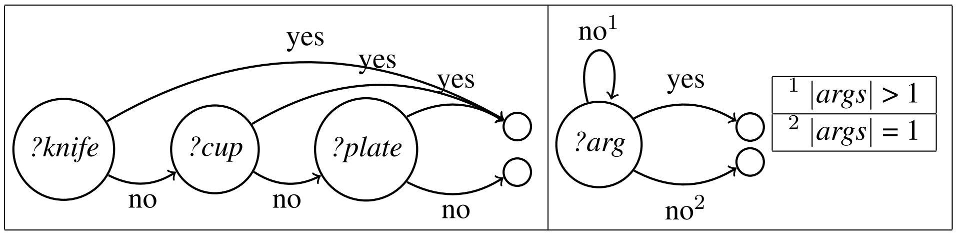


Figure 4. Contrasting state machine architecture fragments for disambiguation, using individual states for each object (L) and a single state (R) where transitions are also based on conditions on the set of available arguments for disambiguation (1, 2) at the time the agent enters the disambiguation state.

Evaluating the transition relation against conditions on the arguments being evaluated over at runtime necessitates storing these arguments as symbols elsewhere. In our implementation we elected to use a pushdown automaton (PDA), as the context-free grammar of the interaction is Turing equivalent to a nondeterministic PDA, and to disallow operations on stack symbols other than the topmost. Therefore, we can store existing conversational context, such as the object that is the current focus of conversation, on the stack symbol of a PDA. In the disambiguation example, a "no" input pops the stack to the next option, while a "yes" answer may rewrite or push a new stack symbol. The stack symbol can be constructed to store whatever information is necessary for the interaction. Our implementation typically stores indicated objects and regions, object that are being grasped by the agent, and options for objects and actions needing disambiguation.

We implement some modifications to the traditional structure of a PDA, which come from the requirements for using situatedness to establish context and composing information in real time. As noted in Fig. 4R, the two “no" transitions shown differ not in the particular value of the argument being evaluated in the disambiguation state, but in the *conditions* on the set of possible arguments when entering that state. This is important given the continuous nature of the world. Using a PDA instead of an FSA is of no help in reducing the number of states in the search space if multiple transitions from the same state on the same input symbol still have to be generated for every particular value of a continuous stack symbol parameter. For example, a coordinate indicated by deixis can move continuously through the 3D world, particularly with noisy vision and interpolated gesture recognition, and it is much more useful and concise to create a condition in the transition relation to check if the indicated coordinate is “not null" or falls within a certain area than to, e.g., create one transition if the indicated coordinate is $(0.0, 2.7, -0.4)$ and another if it is $(0.1, 2.7, -0.4)$.

We add two operations to the traditional PDA set of PUSH, POP, and REWRITE. The agent may need to disregard some or all preceding context, e.g., on the aborting or completion of an action, so FLUSH clears the stack, and the stack symbol, except for information that persists physically, such as objects held by the agent. POPUNTIL functions similarly, but takes a state as a content argument, and pops the stack until the stack symbol equals what it was in the previous occurrence of that state (this is equivalent to FLUSH if the specified state has never been entered previously).

3. Situatedness, Composition, and Reasoning

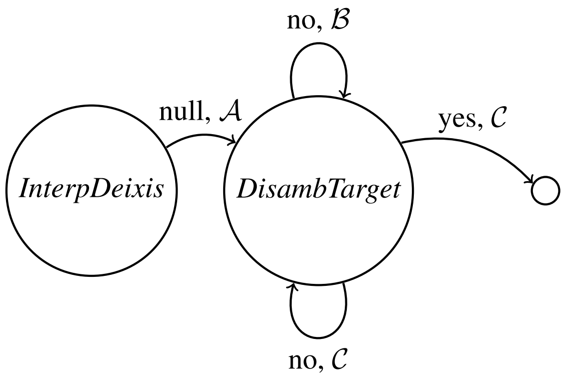
Now imagine that in prior to entering the disambiguation loop in Fig. 4, the human has already specified an action to be undertaken with whatever object is to be singled out via the available disambiguation strategies. Once the object has been successfully indicated, the action, which may have been defined many states ago, must be retrieved and applied to the object. In continuation-passing style (CPS), this information is specified as the “what to do next" argument (Van Eijck & Unger, 2010) as in a Montague grammar (Barker, 2004), and can be represented using Van Eijck & Unger’s CPS function-application over the action denoted and object emerging from the disambiguation loop, as shown in a Haskell fragment:

```
cpsApply ::  Comp (a -> b) r -> Comp a r -> Comp b r
cpsApply m n = \ k -> n (\ b -> m (\ a -> k (a b)))

intAct_CPS ::  WorldState -> Action -> Comp (Object -> Bool) Bool
intAct_CPS bs (Action act obj) = cpsApply (intTAct_CPS bs act)
   (intObj_CPS obj)
```

For example, we can specify a method to execute at the moment of state transition that will retrieve the action specified, apply it to objects or locations indicated by deixis, and prompt the agent to ask appropriate questions using possible interpretations of the action+object/location composition and present them to its interlocutor.

Fig. 5 shows the steps through a deictic interpretation and disambiguation step with a previously-specified "put" action on the stack with no destination yet specified. The transition from *InterpDexis* to *DisambTarget* executes a function that supplies three possible destinations to stack symbol $\mathcal{A}$: objects o_1 and o_2 and location (x, y, z). The subsequent stack symbols $\mathcal{B}$ and $\mathcal{C}$ are created by POPping in the "no" transitions, returning to the same state. At each step, the next destination option is applied to the variable w in the action predicate until the human confirms that (x, y, z) is the desired destination. By exploiting continuation-passing style we can raise the type of the objects or location to the type required by the action "put."



$\mathcal{A}$	[o_1, o_2, (x, y, z)], λw.put(b,on(w))@o_1
$\mathcal{B}$	[o_2, (x, y, z)], λw.put(b,on(w))@o_2
$\mathcal{C}$	[(x, y, z)], λw.put(b,on(w))@(x, y, z)

Figure 5. PDA disambiguation fragment with continuation-passing style and function application on stack symbol.

Our HAI system is also capable to one-shot learning for iconic gestures to indicate grasping particular objects, such as learning that miming holding a cup (such as part of the American Sign Language sign for "cup") is an instruction to grasp the cup in that pose. Having learned this instruction, the human can instruct the avatar to grasp the cup with a single gesture instead of indicating the cup first and then the grasping action. However, this can also be used to fill in gaps in the existing context as part of action sequences other than "grasp." The VoxML encoding for $put(x, y)$ contains a $grasp(x)$ precondition before the actual object movement. Thus, if the agent enters a state where the stack symbol contains an action with an outstanding variable—λb.put(b,v)—and the human supplies the iconic gesture for $grasp(cup)$, the avatar can directly lift the type $e \rightarrow t$ from $grasp(cup)$ to λb.put(b,v) and apply the argument cup to b: λb.put(b,v)@$cup \Rightarrow$ put(cup,v).

4. Discussion and Conclusions

The nondeterministic PDA architecture presented facilitates multimodal reasoning and interaction in real time. Implementationally, we exploit the continuation-passing style available in the C# language to use it with the Unity game engine on which VoxSim is built.[1] There may be cases where simpler or more restrictive behaviors are needed, while still requiring access to the contextual information provided by the agent's situatedness relative to the human and the world. In these cases, the nondeterministic PDA serve as a general case of a deterministic PDA (where probabilities on all transition arcs equal 1), a nondeterministic finite automaton (where the stack symbol is always NULL), or a standard deterministic FSA (where all probabilities are 1 and the stack symbol is NULL).

Continuation-passing style as a method of incrementally aggregating contextual information through a discourse functions with all these methods. Methods of any return type can be executed in state transitions as long as the return type can be raised to the type required by the calling function, and this makes it effective at composing from multiple modalities in real time.

[1] VoxSim repository at https://github.com/VoxML/VoxSim

Acknowledgements

This work is supported by a contract with the US Defense Advanced Research Projects Agency (DARPA), Contract CwC-W911NF-15-C-0238. Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We would like to thank Kelley Lynch for the Haskell implementation code snippet, and Kyeongmin Rim for work on the VoxSim platform.

References

Asher, N. (1998). Common ground, corrections and coordination. *Journal of Semantics*.

Ballard, D. H., Hayhoe, M. M., Pook, P. K., & Rao, R. P. (1997). Deictic codes for the embodiment of cognition. *Behavioral and Brain Sciences*, *20*, 723–742.

Barker, C. (2004). Continuations in natural language. *CW*, *4*, 1–11.

Breazeal, C. (2003). Toward sociable robots. *Robotics and autonomous systems*, *42*, 167–175.

Dautenhahn, K., Ogden, B., & Quick, T. (2002). From embodied to socially embedded agents–implications for interaction-aware robots. *Cognitive Systems Research*, *3*, 397–428.

Goodrich, M. A., Schultz, A. C., et al. (2008). Human–robot interaction: a survey. *Foundations and Trends® in Human–Computer Interaction*, *1*, 203–275.

Jaimes, A., & Sebe, N. (2007). Multimodal human–computer interaction: A survey. *Computer vision and image understanding*, *108*, 116–134.

Krishnaswamy, N., & Pustejovsky, J. (2016). Multimodal semantic simulations of linguistically underspecified motion events. *Spatial Cognition X: International Conference on Spatial Cognition*. Springer.

Krishnaswamy, N., & Pustejovsky, J. (2018). An evaluation framework for multimodal interaction. *Proceedings of LREC*.

Krishnaswamy, N., et al. (2017). Communicating and acting: Understanding gesture in simulation semantics. *12th International Workshop on Computational Semantics*.

Luger, E., & Sellen, A. (2016). Like having a really bad pa: the gulf between user expectation and experience of conversational agents. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 5286–5297). ACM.

Narayana, P., et al. (2018). Cooperating with avatars through gesture, language and action. *Intelligent Systems Conference (IntelliSys)*.

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, *21*, 682–697.

Pustejovsky, J. (2018). From actions to events. *Interaction Studies*, *19*, 289–317.

Pustejovsky, J., & Krishnaswamy, N. (2016). VoxML: A visualization modeling language. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Paris, France: European Language Resources Association (ELRA).

Pustejovsky, J., & Krishnaswamy, N. (2019). Situational grounding within multimodal simulations. *arXiv preprint arXiv:1902.01886*.

Pustejovsky, J., Krishnaswamy, N., Draper, B., Narayana, P., & Bangar, R. (2017). Creating common ground through multimodal simulations. *Proceedings of the IWCS workshop on Foundations of Situated and Multimodal Communication*.

Reynolds, J. C. (1993). The discoveries of continuations. *Lisp and symbolic computation*, *6*, 233–247.

Scheutz, M., Schermerhorn, P., Kramer, J., & Anderson, D. (2007). First steps toward natural human-like hri. *Autonomous Robots*, *22*, 411–423.

Scheutz, M., Schermerhorn, P., Kramer, J., & Middendorff, C. (2006). The utility of affect expression in natural language interactions in joint human-robot tasks. *ACM SIGCHI/SIGART Human-Robot Interaction: Proceeding of the 1 st ACM SIGCHI/SIGART conference on Human-robot interaction* (pp. 226–233).

Van Eijck, J., & Unger, C. (2010). *Computational semantics with functional programming*. Cambridge University Press.

van Wijngaarden, A. (1966). *Recursive definition of syntax and semantics*. North Holland Publishing Company.

Zhang, S., Sinapov, J., Wei, S., & Stone, P. (2017). Robot behavioral exploration and multimodal perception using pomdps. *AAAI Spring Symposium on Interactive Multisensory Object Perception for Embodied Agents*.