# Multimodal Continuation-style Architectures for Human-Robot Interaction

Nikhil Krishnaswamy and James Pustejovsky
Brandeis University

ACS Cognitive Vision Workshop
August 2, 2019

# Motivation

**Human-robot interaction is inherently multimodal**

- Robotic agents are *embodied*, *situated*, and can affect the physical world;
- must have accurate and fast interpretation of multiple *input modalities*;
- must communicate using all available *communicative modalities* (e.g., natural language, gesture, action demonstration, affect, etc.)

# Motivation

**Multimodal interpretive architectures must capture these in context**

1. relative embodiment of human + robot/agent
2. situatedness w.r.t. environment + each other ("*co-situatedness*") (Pustejovsky et al., 2017)

- agent must model itself in the world of its interlocutors and interpret contextualized input relative to that space (Pustejovsky and Krishnaswamy, 2019);
- a situated, multimodal interface is at minimum a *social interface* (Breazeal, 2003).

# Overview

- Our HRI architecture integrates real-time multimodal input into an agent's contextual model;
  - We treat aligned speech and gesture as an *ensemble* where content may be communicated in either modality;
  - Modified nondeterministic pushdown automaton architecture:
    1. Consumes incremental input using continuation-passing style;
    2. Constructs and asks questions using contextual information;
    3. Keeps track of prior discourse items using multimodal cues.
  - Many reasoning engines can be created using this framework;
  - Built on top of the VoxML modeling language (Pustejovsky and Krishnaswamy, 2016) for object and event semantics;
  - Modular design facilitates integration with other robotic architectures.

# Overview

- Currently deployed on systems using virtual agent;
- *human-avatar interaction* (HAI) = interaction between a human and an *embodied, situated agent*—animated avatar *or* robotic agent.


Link

# Scenario

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

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

- HAI may require different modalities for different information
  - e.g., grounding location directly if description is too complicated.

# Scenario


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

Figure: **L**: Example multimodal interaction with deixis. **R**: VoxML typing of [[POINT]]. $\text{E}_2$ is the target of deixis—intersection of the vector extended in $\text{E}_1$ with location $z$, and reifies that point as variable $w$. $\text{A}_4$, shows the compound binding of $w$ to the indicated region and objects within that region.

# Dialogue Structure

- Human may specify object [A], then location [B] and action
- Deixis grounds to specific location/objects + “there,” selects for location


Figure: Deixis to region with objects.

- Given deixis, object properties select further
  - “cup,” “that cup”, “the blue cup,” “in that blue cup,” “put the knife in the blue cup”...
  - ... all single out the same object in the region
  - Multiple modalities specify objects, locations, or actions

# Dialogue Structure

- Referencing strategies and instructions can be as over- or underspecified as needed;
- Agent may respond with a question to extract missing information;
- Question composition requires tracking:
    1. Information directly acquired from all input modalities
    2. Contextual information acquired from the situation
    3. Information inferred from composition of (1) and (2)

$$\left[\ +\text{cup}\ \right] \oplus \text{“Put it there.”} \oplus \left[\begin{array}{l}\text{-location} \\ \text{-deixis}\end{array}\right]$$

$\rightarrow$ “Where should I put the cup?”

# Grammar

Interaction vocabulary:

- "Moves" by both interlocutors (Krishnaswamy and Pustejovsky, 2018; Pustejovsky, 2018) using CFG format
- Nonterminals = input symbols; terminals = content or intended response communicated by input symbols

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

Table: Interactive grammar snippet. Unexpanded nonterminals represent parsed sentences/phrases or gesture variations.

# Grammar

- Disambiguation symbol $D$ represents question cycle:
  - Acquisition of information agent still needs to complete action initiated or requested by the human.
- Order of instructions may vary;
  - Agent must hold known information "in reserve" pending further instruction or answers.

# Grammar $\nrightarrow$ FSA

- Even in a superficial system, new states for every possible context is intractable;
  - (Reflected in the large number of terminals in grammar.)
  - e.g., if three objects exist in scene, object disambiguation should not require a different state for each;
  - Instead, recurse through the same state with a different contextual symbol until affirmative received, then handle the argument.

# Grammar $\nrightarrow$ FSA

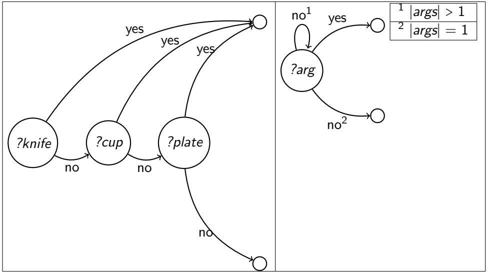



Figure: Contrasting state machine architecture fragments for disambiguation, using individual states for each object (L) and a single state (R) where transitions are also based on conditions on the set of available arguments for disambiguation (1, 2) at the time the agent enters the disambiguation state.

# Evaluating Conditions

- Evaluating transition relation against conditions on the arguments means storing these arguments elsewhere
  - We use a stack, rendering the architecture a pushdown automaton (PDA).
  - CFG of the interaction is Turing-equivalent to a nondeterministic PDA;
  - Disallows operations on non-topmost stack symbols.
- We store existing conversational context (e.g., incl. current focus object) on the PDA stack symbol.

# Evaluating Conditions

- e.g., for disambiguation:
  - “No” response pops stack and proceeds to next option;
  - “Yes” response rewrites or pushes new stack symbol.
- Stack symbol can be constructed to store whatever information needed for interaction
  - Current implementation stores:
    1. indicated objects and regions
    2. objects being grasped by the agent
    3. options for object and action disambiguation

# Innovations

- We implement some modifications to the traditional structure of a PDA;
- Innovations motivated by requirements on using siutatedness to establish context, and composing information in real time.
  - e.g., ▸ Disambiguation
  - "No" transitions differ not in argument value, but in *conditions* on set of possible arguments when entering that state;
- This is important given the continuous nature of the world.

# Innovations

- PDA provides no advantage over FSA if multiple transitions from state $q$ on input symbol $\sigma$ must be generated for every value of a parameter of a continuous symbol $Z$ (e.g., a coordinate).
  - Deixis can move continuously through the 3D world, can be noisy;
  - Why create different transitions for coordinates $(0.0, 2.7, -0.4)$ vs. $(0.1, 2.7, -0.4)$?
  - Check if coordinate falls in range?
  - Check if region is "not undefined"?
- This approach provides usefulness and concision.

# Innovations

- PDA contains standard PUSH, POP, and REWRITE operations;
- We add 2: FLUSH and POPUNTIL.
- FLUSH
  - Agent may need to disregard some/all preceding context;
  - FLUSH clears the stack and stack symbol except for physically persistent information, such as objects held by the agent
- POPUNTIL
  - Takes a state as content argument;
  - Pops the stack until stack symbol equals status in previous occurrence of that state.
  - (this is equivalent to FLUSH if the specified state has never been entered previously)

# Innovations

- Ability to redirect transitions

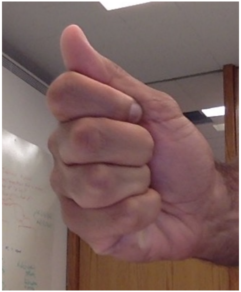
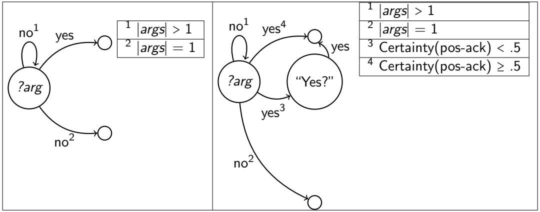


Figure: Ambiguous input

- Computer may need to confirm intent before proceeding

# Innovations




- By passing a function and alternate state to the transition relation update, the state transition can be branched depending on the function output, maintaining established context;
- Transition becomes a test.

# Where Have the Continuations Gone?

- Scenario: prior to entering disambiguation loop, human has already specified an action
  - $\lambda\, \mathrm{x}.grasp(x)@\_\_$
- Action may have been defined many states ago;
  - Once object is known, action has to be retrieved and applied.
- In *continuation-passing style* (CPS), this is the "what to do next" argument (Van Eijck and Unger, 2010);
  - May be represented using CPS function-application over denoted action and indicated object.

# Where Have the Continuations Gone?

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)
```

# Where Have the Continuations Gone?

Extended Haskell fragment:

```
intTAct_CPS ::  WorldState -> Gesture -> Comp
      (Loc -> Loc -> Bool) Bool
intTAct_CPS bs Move = cpsConstAct move bs
cpsConstAct ::  (WorldState -> a) -> WorldState ->
      Comp a r
cpsConstAct c bs = \k -> k (c bs)
cpsConst ::  a -> Comp a r
cpsConst c = \k -> k c
cpsConstAct ::  (WorldState -> a) -> WorldState ->
      Comp a r
cpsConstAct c bs = \k -> k (c bs)
```

# Implementation

- VoxSim: built on Unity, written in C# (mostly);
- C# has both imperative and functional features;
- 3 features of C# make this implementation possible: Anonymous delegates, lambda expressions, compiled and invokeable predicates.

Figure: VoxSim (Krishnaswamy and Pustejovsky, 2016)

# VoxSim

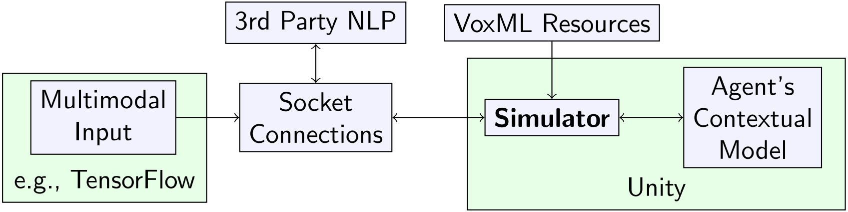



Figure: VoxSim architecture (adapted from Krishnaswamy (2017))

- Unity-based event visualization engine
  - Real-time visual event simulation
  - Human-Avatar Interaction in collaborative task setting

# VoxWorld Architecture

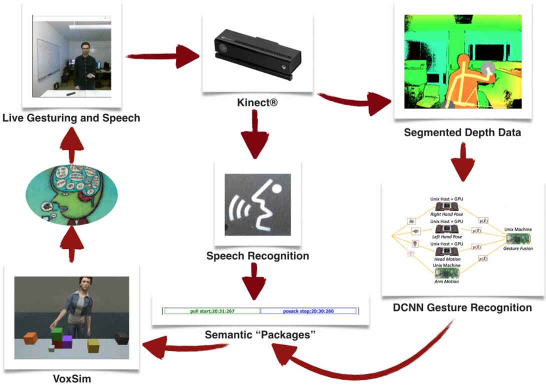
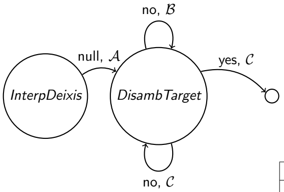



Figure: VoxWorld architecture (Krishnaswamy et al., 2017), in collaboration with Colorado State University and University of Florida

# Implementation

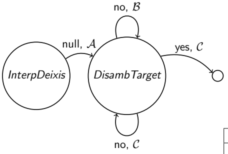



| | |
|---|---|
| $\mathcal{A}$ | $[o_1,\ o_2,\ (x, y, z)],\ \lambda w.\text{put}(b,\text{on}(w))@o_1$ |
| $\mathcal{B}$ | $[o_2,\ (x, y, z)],\ \lambda w.\text{put}(b,\text{on}(w))@o_2$ |
| $\mathcal{C}$ | $[(x, y, z)],\ \lambda w.\text{put}(b,\text{on}(w))@(x, y, z)$ |

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

- Specify a method to execute at transition that:
  1. retrieves the action;
  2. apply it to objects or locations once indicated;
  3. prompts the agent to question its interlocutor about its interpretation of the composed information.

# Implementation




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

Figure: Shown: deictic interpretation and disambiguation, with "put" action on the stack and no specified destination.

- *InterpDexis* → *DisambTarget* executes function that supplies three possible destinations to stack symbol $\mathcal{A}$;
- $\mathcal{B}$ and $\mathcal{C}$ are created by POPping in the "no" transitions;
- Options applied to $w$ until human confirms $(x, y, z)$;
- By exploiting CPS we can raise all options to type required by event.

# Use in Learning

- HAI + CPS facilitates one-shot learning of iconic gestures

Figure: Using iconic "cup" gesture to signal "grasp the cup"

- Having learned the gesture's correlated instruction, the human can instruct the avatar to grasp an object with 1 gesture

Example

# Use in Learning

- CPS allows filling in context for other action sequences.

$$
\left[\begin{array}{ll}
\textbf{put} & \\
\mathrm{LEX} = & \left[\begin{array}{lll} \mathrm{PRED} & = & \textbf{put} \\ \mathrm{TYPE} & = & \textbf{transition\_event} \end{array}\right] \\
\mathrm{TYPE} = & \left[\begin{array}{lll}
\mathrm{HEAD} & = & \textbf{transition} \\
\mathrm{ARGS} & = & \left[\begin{array}{lll} \mathrm{A}_1 & = & \textbf{x:agent} \\ \mathrm{A}_2 & = & \textbf{y:physobj} \\ \mathrm{A}_3 & = & \textbf{z:location} \end{array}\right] \\
\mathrm{BODY} & = & \left[\begin{array}{lll} \mathrm{E}_1 & = & \color{blue}{grasp(x,y)} \\ \mathrm{E}_2 & = & while(hold(x,y) \wedge !at(y,z)) : move\_to(x,y,z) \\ \mathrm{E}_3 & = & if(at(y,z)) : ungrasp(x,y) \end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure: VoxML encoding for $[[\text{PUT}]]$

# Use in Learning

- CPS allows filling in context for other action sequences.
- VoxML for [[PUT]] contains a [[GRASP]] subevent as precondition.
- If agent enters state where context contains an action with an outstanding variable:
  - $\lambda b.\mathrm{put}(b,z)$
  - Human supplies learned gesture for *grasp*(*knife*);
  - Directly lift $e \rightarrow t$ from *grasp*(*knife*) to $\lambda b.\mathrm{put}(b,z)$;
  - Apply the argument *knife* to $b$: $\lambda b.\mathrm{put}(b,z)@knife \Rightarrow \mathrm{put}(knife,z)$.

▸ Example

# Putting It All Together


Go!

# Discussion and Conclusions

- Nondeterministic PDA Architecture presented facilitates multimodal reasoning and interaction in real time;
- There may be cases where simpler behaviors are needed, still requiring access to context provided by agent's situatedness and multimodal input.
- NPDA can serve as a special case of DPDA:
  - All conditions have 1 associated transition;
  - If governed by probabilities, probabilities on all arcs equal 1.
- NPDA can serve as NFA:
  - Stack symbol is always NULL.
- NPDA can serve as DFA:
  - NULL stack symbol, single transition arcs.

# Discussion and Conclusions

- Ability to execute a function or method during transitions can be exploited to run custom code, e.g.:
  - Path planners (implemented)
  - Parsers (implemented)
  - Network learners (in progress)

# Discussion and Conclusions

- Continuation-passing style through a discourse to incrementally aggregate contextual information functions with all these architectures;
- Methods of any return type can be executed in state transitions if return type can be raised to the type required by the calling function;
- This makes it effective at composing from multiple modalities in real time.

# Thank You!

https://github.com/VoxML/VoxSim
Currently undergoing extensive refactor!
We hope to make a release publicly available soon!

# Thank You!

# Thank You!


- This research was funded by a contract with the US Defense Advanced Research Projects Agency (DARPA), Contract CwC-W911NF-15-C-0238.

**Brandeis Lab for Linguistics and Computation**

- Kyeongmin Rim, Kelley Lynch, Tuan Do
  - Research assistants: Mark Hutchens, Jin Zhao

**Collaborators at Colorado State University**

- Bruce Draper, Ross Beveridge, Pradyumna Narayana, Rahul Bangar, Dhruva Patil, Jason Yu, David White, Joe Strout

**Collaborators at University of Florida**

- Jaime Ruiz, Isaac Wang, Brett Benda

# References I

- Breazeal, Cynthia (2003). "Toward sociable robots". In: *Robotics and autonomous systems* 42.3-4, pp. 167–175.
- Krishnaswamy, Nikhil (2017). "Monte-Carlo Simulation Generation Through Operationalization of Spatial Primitives". PhD thesis. Brandeis University.
- Krishnaswamy, Nikhil and James Pustejovsky (2016). "VoxSim: A Visual Platform for Modeling Motion Language". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. ACL.
- – (2018). "An Evaluation Framework for Multimodal Interaction". In: *Proceedings of LREC*.

# References II

Krishnaswamy, Nikhil et al. (2017). "Communicating and Acting: Understanding Gesture in Simulation Semantics.". In: *12th International Workshop on Computational Semantics*.

Pustejovsky, James (2018). "From actions to events". In: *Interaction Studies* 19.1-2, pp. 289–317.

Pustejovsky, James and Nikhil Krishnaswamy (2016). "VoxML: A Visualization Modeling Language". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Portoroz, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

– (2019). "Situational Grounding within Multimodal Simulations". In: *arXiv preprint arXiv:1902.01886*.

# References III

Pustejovsky, James et al. (2017). "Creating Common Ground through Multimodal Simulations". In: *Proceedings of the IWCS workshop on Foundations of Situated and Multimodal Communication.*

Van Eijck, Jan and Christina Unger (2010). *Computational semantics with functional programming*. Cambridge University Press.