**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.
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.
### Scenario

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent: [ \text{agent puts plate at indicated location} ] “Okay.”</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

Figure: L: Example multimodal interaction with deixis. R: VoxML typing of [[[POINT]]]. E₂ is the target of deixis—intersection of the vector extended in E₁ with location z, and reifies that point as variable w. A₄ shows the compound binding of w to the indicated region and objects within that region.
Introduction
Interactive Structure
Formal Structure
Situatedness, Composition, and Reasoning
Discussion and Conclusions
References

Scenario
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
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)

\[
\begin{array}{c}
+\text{cup} & \oplus & \text{“Put it there.”} & \oplus & -\text{location} \\
\end{array}
\]

\[
\rightarrow \text{“Where should I put the cup?”}
\]
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**

\[
S \rightarrow OA | AO \\
O \rightarrow \delta|\delta D|\omega|\omega D|N|ND \\
A \rightarrow \alpha|\alpha D|V|VD|P|PD \\
D \rightarrow \delta|\delta D|P|PD|N|ND|y|yD|n|nD
\]

**Legend**
- $O$: define object
- $A$: define action
- $D$: disambiguate
- $\delta$: deictic gesture
- $\omega$: static iconic gesture (object)
- $\alpha$: dynamic iconic gesture (action)
- $y$: affirmative response
- $n$: negative response
- $N$: noun phrase
- $P$: 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.
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.
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
We implement some modifications to the traditional structure of a PDA;

Innovations motivated by requirements on using situatedness 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
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.
Scenario: prior to entering disambiguation loop, human has already specified an action

\[ \lambda x.\text{grasp}(x)@\_\_ \]

Action may have been defined many states ago;
- Once object is known, action has to be retrieved and applied.

In \textit{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.
As shown in a Haskell fragment:

```haskell
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:

```haskell
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

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

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

- **InterpDeixis** → **DisambTarget** executes function that supplies three possible destinations to stack symbol *A*;
- *B* and *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
Use in Learning

- CPS allows filling in context for other action sequences.

\[
\text{put} \\
\text{LEX} = \begin{bmatrix}
\text{PRED} = & \text{put} \\
\text{TYPE} = & \text{transition\_event}
\end{bmatrix} \\
\text{HEAD} = \text{transition} \\
\text{ARGS} = \begin{bmatrix}
A_1 = & x:\text{agent} \\
A_2 = & y:\text{physobj} \\
A_3 = & z:\text{location}
\end{bmatrix} \\
\text{TYPE} = \begin{bmatrix}
E_1 = & \text{grasp}(x, y) \\
E_2 = & \text{while}(\text{hold}(x, y) \land \neg \text{at}(y, z)) : \text{move\_to}(x, y, z) \\
E_3 = & \text{if}(\text{at}(y, z)) : \text{ungrasp}(x, y)
\end{bmatrix}
\]

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:
  - $\lambda b.\text{put}(b,z)$
  - Human supplies learned gesture for $\text{grasp}(\text{knife})$;
  - Directly lift $e \rightarrow t$ from $\text{grasp}(\text{knife})$ to $\lambda b.\text{put}(b,z)$;
  - Apply the argument $\text{knife}$ to $b$: $\lambda b.\text{put}(b,z)@\text{knife} \Rightarrow \text{put}(\text{knife},z)$.

Example
Putting It All Together

Got it! Do that with your left hand and I'll grasp the cup like this.
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.
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)
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.


References II


