Building Multimodal Simulations for Natural Language

James Pustejovsky
Nikhil Krishnaswamy
Brandeis University

EACL 2017
Valencia, Spain
April 3, 2017

Tutorial Outline

- Introduction to Multimodal Semantics (14:30-15:30)
  - Generative Lexicon Types and Habitat Theory:
  - Computational Theory of Affordances:
  - Dynamic Logic for Event Structures:
- Overview of VoxSim (15:00-16:00)
  - Module 1: Architecture and Program Flow:
  - Module 2: Object Modeling:
  - Module 3: Action and Gesture Modeling:
  - Module 4: Event Modeling: Integrating Object and Action Models
- Coffee Break (16:00-16:30)
- Creating Simulations: Modeling Novel Content (16:30-18:00)
  - Activity 1: Voxeme Modeling from 3D Geometry Library:
  - Activity 2: Behavior Attachment to a Voxeme
  - Activity 3: Adding Discriminating Attributes to Voxemes:
  - Activity 4: Creating Novel Behavior:

Starting Assumptions

- **Language visualization and simulation generation**: Creating images from linguistic input; generating dynamic narratives in simulation environments from action-oriented expressions; (Clay and Wilhelms, 1996; Coyne and Sproat, 2001; Seversky and Yin, 2006; Siskind, 2001; Chang et al., 2015)

Starting Assumptions

- **Visual Question-Answering and image content interpretation**: QA and querying over image datasets, based on the vectors associated with the image, but trained on caption-image pairings in the data; (Antol et al., 2015; Chao et al., 2015a; Chao et al., 2015b)

Starting Assumptions

- **Gesture interpretation**: Understanding integrated spoken language with human or avatar-generated gestures; generating gesture in dialogue to supplement linguistic expressions; (Rautaray and Agrawal, 2015; Jacko, 2012; Turk, 2014; Bunt, Beun, and Borghuis, 1998)

Wordseye Coyne and Sproat (2001)

- Automatically converts text into representative 3D scenes.
- Relies on a large database of 3D models and poses to depict entities and actions
- Every 3D model can have associated shape displacements, spatial tags, and functional properties.

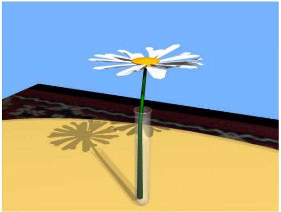


Figure 7: *The daisy is in the test tube.*

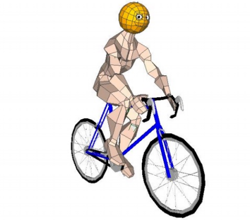
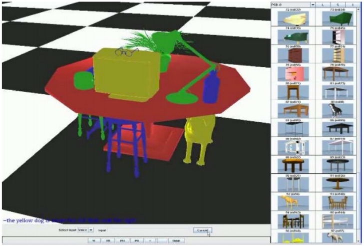


Figure 9: Usage pose for a 10-speed.

Automatic 3D scene generation Seversky and Yin (2006)

- The system contains a database of polygon mesh models representing various types of objects.
- composes scenes consisting of objects from the Princeton Shape Benchmark model database 2



Related Research on Scene and Simulation Construction

- Spatial and temporal interval logics
  - Allen Temporal Relations (Allen, 1983)
  - Region Connection Calculus (RCC8) (Randell et al., 1992)
  - RCC-3D (Albath, et al., 2010)
- Generative Lexicon, DITL (Pustejovsky, 1995; Pustejovsky and Moszkowicz, 2011)
- Static scene generation
  - WordsEye (Coyne and Sproat, 2001)
  - LEONARD (Siskind, 2001)
  - Stanford NLP Group (Chang et al., 2015)
- QSR/Game AI approaches to scenario-based simulation (Forbus et al., 2001; Dill, 2011)
- Spatial constraint mapping to animation (Bindiganavale and Badler, 1998)

Requirements on a Multimodal Semantic Simulation

1. A minimal embedding space (MES) for the simulation must be determined. This is the 3D region within which the state is configured or the event unfolds;
2. Object-based attributes for participants in a situation or event need to be specified; e.g., orientation, relative size, default position or pose, etc.;
3. An epistemic condition on the object and event rendering, imposing an implicit point of view (POV);
4. Agent-dependent embodiment; this determines the relative scaling of an agent and its event participants and their surroundings, as it engages in the environment.

Visual Object Concept Modeling Language (VoxML)

Pustejovsky and Krishnaswamy (2014, 2016)

- Modeling language for constructing 3D visualizations of concepts denoted by natural language expressions
- Used as the platform for creating *multimodal semantic simulations*
- Encodes dynamic semantics of events and objects and object properties
- Platform independent framework for encoding and visualizing linguistic knowledge.

Goals and Methodology

- Envisioning Language through Multimodal Simulations
  - Integrating linguistic and visual modes of representation, expression, and interpretation
- Semantic models are extensible and generative
  - There are base semantic templates for Communicative Acts (CAs)
  - There is an identifiable compositional syntax for building CAs
  - There are contextualization strategies for adapting and modifying CAs
- Semantic models are embodied and multimodal
  - Embodiment must be assumed to ground concepts within a domain
  - Modalities (linguistic, perceptual, and effector) constitute distinct aspects of representation

Communicating through Simulations

- Formal Models Provide a Reasoning Platform for the Computer
  - Minimal finite model enables inference; but ...
- They are not an effective medium for communicating with humans
  - Communication is facilitated through semiotic structures that are shared and understood by both partners.
- Multimodal semantic simulations are embodied representations of situations and events
  - Image schemas and visualizations of actions are core human competencies

Multimodal simulations as visualizing context

- Concepts are realized as types, linguistically, visually, behaviorally, acoustically, emotionally
- Context is encoded through modal collocations
- A multimodal object should encode these types and intra- and inter-modal collocations

Visualizing Context 1/3

Figure: Eagle in flight

Visualizing Context 2/3

Figure: Eagle perching

Visualizing Context 3/3

Figure: Eagle nesting

Visualizing Context 1/3

Figure: Pencil in a cup

Visualizing Context 2/3

Figure: Pencil in a drawer

Visualizing Context 3/3

Figure: Writing with a pencil

The Prototype Effect

- The pencil is in the cup. [+vertical]
- The pencil is in the drawer. [+horizontal]
- The man is using a pencil. [+diagonal]

Object Situation Disambiguation (OSD):
Disambiguate the contextual meaning of an object-denoting word to the appropriate situation.

Mental Models

- Craik (1943)
  Agents carry small-scale models of external reality in their head...
- Johnson-Laird (1983)
  A mental model represents a possibility, capturing what is common to all the different ways in which the possibility may occur (Johnson-Laird and Byrne, 2002). Used to drive inference and reasoning.
- Gentner and Stevens (1983)
  Understanding human knowledge about the world: Domain; Theoretical Approach; Methodology.

Embodiment

- Meaning centrally involves the activation of perceptual, motor, social, and affective knowledge that characterizes the content of utterances
- Understanding a piece of language is hypothesized to entail performing mental perceptual and motor simulations of its content

Approaches to Modeling Events

- Model-Theoretic Semantics:
  Montague (1968), Davidson (1967), Kamp (1969), Partee (1975), Dowty (1979), Verkuyl (1972), Kim (1973), Kratzer (1994), Piñon (1997)
- Decompositional Semantics:
  Lakoff (1965), Fillmore (1968), Jackendoff (1972), Talmy (1975), Langacker (1987), Fillmore (1985), Jackendoff (1983)
- Lexical-semantic approaches:
  Higginbotham (1986), Tenny (1987), Pustejovsky (1991, 1995), Krifka (1998), Levin and Hovav-Rappaport (1995)
- Modern Syntheses:
  Naumann (2001), Steedman (2002), Fernando (2013), van Lambalgen and Hamm (2005), Pustejovsky (2013)

Cognitive Simulations of Events

- Frame Semantics
  Fillmore (1966, 1968, 1977), Jackendoff (1972, 1983), Minsky (1974), Löbner (2013)
- Mental Simulations
  Graesser et al (1994), Barselou (1999), Zwaan and Radvansky (1998), Zwaan and Pecher (2012)
- Embodiment:
  Johnson (1987), Lakoff (1987), Varela et al. (1991), Clark (1997), Lakoff and Johnson (1999), Gibbs (2005)
- Simulation Semantics
  Goldman (1989), Feldman et al (2003), Goldman (2006), Feldman (2010), Bergen (2012), Evans (2013),
- Qualitative Mental Models
  Forbus and Gentner (1997), Klenk et al (2005),

Simulations and Grounded Cognition

- Open-class items tend to activate perceptually based simulations
  - Concrete verbs (within this class) activate motor areas
  - Abstract verbs tend not to activate these areas
- Functional items: no overt simulation. But they provide logical constraints.
- Verbs: abstracted as operational semantic procedures.

Simulations as Minimal Models

- Theorem proving (essentially type satisfaction of a verb in one class as opposed to another) provides a "negative handle" on the problem of determining consistency and informativeness for an utterance (Blackburn and Bos, 2008; Konrad, 2004)
- Model building provides a "positive handle" on whether two manner of motion processes are distinguished in the model.
- The simulation must specify *how* they are distinguished, demonstrating the informativeness of a distinction in our simulation.

Requirements for Multimodal Models of Semantics

- Internal structure of events
- Temporal anchoring and ordering of events
- Event localization: where the event takes place over time
- Rich object semantics: qualia structure, affordances, habitats
- Capturing the dynamics of the event

Putting Space in Language

- Space as Modality: "add an operator"
  $P_\alpha(meet(john, mary))$
  (Rescher and Garson, 1968, von Wright, 1979, Bennett, 1995, etc.)
- Method of Spatial Arguments: "add an l in a relation"
  $\exists l[meet(john, mary, l) \wedge in(l, \text{Boston})]$
  (Whitehead, 1929, Randell et al, 1992, Cohn et al, 1997, etc.)

"To each their own" (Vendler, 1967)

- Events are temporal entities:
  modified by temporal predicates
- Objects are spatial entities:
  modified by spatial predicates
- Temporal properties of objects are derivative
- Spatial properties of events are derivative

Locating Events (Davidson, 1967)

- An event is a first-order individual, e:

$$P(x_1, \ldots, x_n, e)$$

- We can identify the location of an event by a relation:

$$loc(e, l)$$

- $\exists e \exists x[smoke(j, e) \wedge in(e, x) \wedge bathroom(x)]$

a. John sang in a field.
$\exists e \exists l[sing(j, e) \wedge in(e, l) \wedge field(l)]$
b. Mary ate her lunch under a bridge.
$\exists e \exists l[eat\_lunch(m, e) \wedge under(e, l) \wedge bridge(l)]$
c. The robbery happened behind a building.
$\exists e \exists l[robbery(e) \wedge behind(e, l) \wedge building(l)]$

Locating Events (Kim, 1973, 1975) 1/2

- An event is a structured object exemplifying a property (or *n*-adic relation), at a time, t:

$$[(x_1, \ldots, x_n, t), P^n]$$

- We can identify the location of an object in the event:

$$loc(x, t) = r_x$$

- For purposes of event identity, we can construe an event as:

$$[(x_1, \ldots, x_n, r_{x_1}, \ldots, r_{x_n}, t), P^n]$$

$$= [([x_i], [r_{x_i}], t), P^i]$$

Locating Events (Kim, 1973, 1975) 2/2

- An event is a structured object exemplifying a property (or *n*-adic relation), at a time, t:

$$[(x_1, \ldots, x_n, t), P^n]$$

- We can identify the location of an object in the event:

$$loc(x, t) = r_x$$

- For purposes of event identity, we can construe an event as:

$$[(x_1, \ldots, x_n, r_{x_1}, \ldots, r_{x_n}, t), P^n]$$

$$= [([x_i], [r_{x_i}], t), P^i]$$

- The event location, l_e, is supervenient on the object locations, $r_{x_1}, \ldots, r_{x_n}$.

Linguistic Approaches to Defining Paths

- Talmy (1985): Path as part of the Motion Event Frame
- Jackendoff (1983, 1990,1996): Minimal Path
- Langacker (1987): COS verbs as paths
- Goldberg (1995): way-construction introduces path
- Krifka (1998): Temporal Trace function
- Zwarts (2006): event shape: The trajectory associated with an event in space represented by a path.

Dynamic Model of Motion Events

- Language encodes motion in Path and Manner constructions
- Path: change with distinguished location
- Manner: motion with no distinguished locations
- Manner and paths may compose.

Subatomic Event Structure (Pustejovsky 1991)

a. $\text{EVENT} \rightarrow \text{STATE} \mid \text{PROCESS} \mid \text{TRANSITION}$

b. $\text{STATE:} \rightarrow \; e$

c. $\text{PROCESS:} \rightarrow \; e_1 \ldots e_n$

d. $\text{TRANSITION}_{ach}\text{:} \rightarrow \text{STATE} \;\; \text{STATE}$

e. $\text{TRANSITION}_{acc}\text{:} \rightarrow \text{PROCESS} \;\; \text{STATE}$

Dynamic Extensions to GL

- Qualia Structure: Can be interpreted dynamically
- Dynamic Selection: Encodes the way an argument participates in the event
- Tracking change: Models the evolution of argument attributes

GL Feature Structure

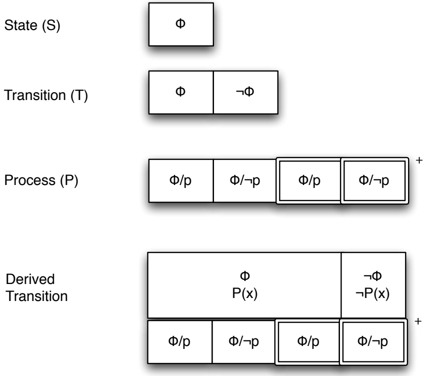
$$
\begin{bmatrix}
\alpha & & \\
\text{ARGSTR} & = & \begin{bmatrix} \text{ARG1} & = & x \\ \ldots & & \end{bmatrix} \\
\text{EVENTSTR} & = & \begin{bmatrix} \text{EVENT1} & = & e1 \\ \text{EVENT2} & = & e2 \end{bmatrix} \\
\text{QUALIA} & = & \begin{bmatrix} \text{CONST} = \textbf{what } x \textbf{ is made of} \\ \text{FORMAL} = \textbf{what } x \textbf{ is} \\ \text{TELIC} = e_2\textbf{: function of } x \\ \text{AGENTIVE} = e_1\textbf{: how } x \textbf{ came into being} \end{bmatrix}
\end{bmatrix}
$$

Inherent Dynamic Aspect of Qualia Structure

- Parameters of a verb, P, extend over sequential frames of interpretation (subevents).
- P is decomposed into different subpredicates within these events:

$$\text{Verb}(\text{Arg}_1\text{Arg}_2) \implies \lambda y \lambda x \boxed{P_1(x,y)}_A \boxed{P_2(y)}_F$$

Frame-based Event Structure



Frame-based Event Structure

Dynamic Interval Temporal Logic
(Pustejovsky and Moszkowicz, 2011)

- Formulas: ϕ propositions. Evaluated in a state, s.
- Programs: α, functions from states to states, $s \times s$. Evaluated over a pair of states, (s, s').
- Temporal Operators: $\bigcirc\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\,\mathcal{U}\psi$.
- Program composition:
  1. They can be ordered, $\alpha;\beta$ (α is followed by β);
  2. They can be iterated, a^* (apply a zero or more times);
  3. They can be disjoined, $\alpha \cup \beta$ (apply either α or β);
  4. They can be turned into formulas
     $[\alpha]\phi$ (after every execution of α, ϕ is true);
     $\langle\alpha\rangle\phi$ (there is an execution of α, such that ϕ is true);
  5. Formulas can become programs, $\phi?$ (test to see if ϕ is true, and proceed if so).

Events are Simple and Macro programs

- The ECI programs include all identifiable basic movements within a given domain, such as: move; grasp; hold; rotate, roll
- Macro programs are complex activities, such as put, stack

put(A, B)
a. Given C being satisfied (A is clear, within reach, etc), then *grasp* A, and **while** *hold* A, *move* A until at position B.
b. $C?; grasp(A); (hold(A)?; move(A))^*; on(A, B)?; ungrasp(A); \neg hold(A)?$

Labeled Transition System (LTS)

The dynamics of actions can be modeled as a Labeled Transition Systems (LTS).

An LTS consists of a 3-tuple, $\langle S, Act, \rightarrow \rangle$, where a. S is the set of states;

b. Act is a set of actions;

c. $\rightarrow$ is a total transition relation: $\rightarrow \subseteq S \times Act \times S$. An action, α provides the labeling on an arrow, making it explicit what brings about a state-to-state transition. As a shorthand for $(e_1, \alpha, e_2) \in \rightarrow$, we will also use:

$e_1 \xrightarrow{\alpha} e_3$

If reference to the state content (rather than state name) is required for interpretation purposes, then as shorthand for: $(\{\phi\}_{e_1}, \alpha, \{\neg\phi\}_{e_2}) \in \rightarrow$, we use: $\boxed{\phi}_{e_1} \xrightarrow{\alpha} \boxed{\neg\phi}_{e_2}$

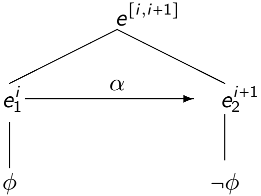
State Transition

Frame-based representation:

$$\boxed{\phi}^{i}_{e_1} \boxed{\neg\phi}^{j}_{e_2}$$

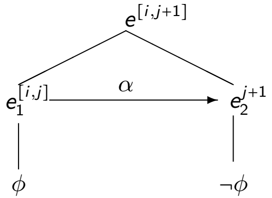
$$\boxed{\phi}^{i}_{e_1} \xrightarrow{\alpha} \boxed{\neg\phi}^{i+1}_{e_2}$$

Dynamic Event Structure



Dynamic Event Structure

Mary awoke from a long sleep.
The state of being asleep has a duration, $[i, j]$, who's valuation is gated by the waking event at the "next state", $j + 1$.



Dynamic Event Structure

$x := y$ (ν-transition)
"x assumes the value given to y in the next state."
$\langle \mathcal{M}, (i, i+1), (u, u[x/u(y)]) \rangle \vDash x := y$
iff $\langle \mathcal{M}, i, u \rangle \vDash s_1 \wedge \langle \mathcal{M}, i+1, u[x/u(y)] \rangle \vDash x = y$

$$\begin{tikzcd}
 & e^{[i,i+1]} \arrow[dl, dash] \arrow[dr, dash] & \\
e_1^i \arrow[rr, "x := y"] \arrow[d, dash] & & e_2^{i+1} \arrow[d, dash] \\
\mathcal{A}(z) = x & & \mathcal{A}(z) = y
\end{tikzcd}$$

Processes

With a ν-transition defined, a *process* can be viewed as simply an iteration of basic variable assignments and re-assignments:

$$e$$

$$e_1 \xrightarrow{\nu} e_2 \ \dots \ \xrightarrow{\nu} \ e_n$$

Spatial Relations in Motion Predicates

- Topological Path Expressions
  arrive, leave, exit, land, take off
- Orientation Path Expressions
  climb, descend
- Topo-metric Path Expressions
  approach, near, distance oneself
- Topo-metric orientation Expressions
  just below, just above

Capturing Motion as Change in Spatial Relations

Dynamic Interval Temporal Logic

- Path verbs designate a distinguished value in the change of location, from one state to another.
  The change in value is tested.
- Manner of motion verbs iterate a change in location from state to state.
  The value is assigned and reassigned.

Motion Leaving a Trail

MOTION LEAVING A TRAIL:

a. Assign a value, y, to the location of the moving object, x.

$$loc(x) := y$$

b. Name this value b (this will be the beginning of the movement);

$$b := y$$

c. Initiate a path p that is a list, starting at b;

$$p := (b)$$

d. Then, reassign the value of y to z, where $y \neq z$

$$y := z, y \neq z$$

e. Add the reassigned value of y to path p;

$$p := (p, z)$$

e. Kleene iterate steps (d) and (e);

Leaving a Trail

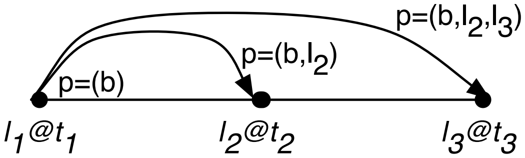


Figure: Directed Motion leaving a Trail

a. The ball rolled 20 feet.
$\exists p \exists x[[roll(x,p) \wedge ball(x) \wedge length(p) = [20, foot]]$

b. John biked for 5 miles.
$\exists p[[bike(j,p) \wedge length(p) = [5, mile]]$

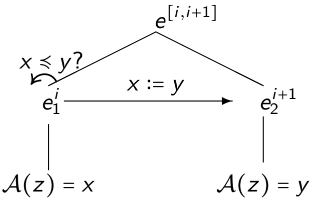
Directed Motion

$$\overset{\overset{x \neq y?}{\curvearrowleft}}{\boxed{loc(z) = x}}_{e_1} \xrightarrow{\nu} \boxed{loc(z) = y}_{e_2}$$

When this test references the ordinal values on a scale, $\mathcal{C}$, this becomes a *directed* ν*-transition* ($\vec{\nu}$), e.g., $x \preccurlyeq y$, $x \succcurlyeq y$.

$$\vec{\nu} \;=_{df}\; \overset{\overset{\mathcal{C}?}{\curvearrowleft}}{e_i} \xrightarrow{\nu} e_{i+1}$$

Directed Motion



Change and Directed Motion

- Manner-of-motion verbs introduce an assignment of a location value:
  $loc(x) := y; y := z$
- Directed motion introduces a dimension that is measured against:
  $d(b, y) < d(b, z)$
- Path verbs introduce a pair of tests:
  $\neg\phi? \ldots \phi?$

Change and the Trail it Leaves

- The execution of a change in the value to an attribute $\mathcal{A}$ for an object x leaves a trail, τ.
- For motion, this trail is the created object of the path p which the mover travels on;
- For creation predicates, this trail is the created object brought about by order-preserving transformations as executed in the directed process above.

Accomplishments Revisited

a. Diana built a staircase.

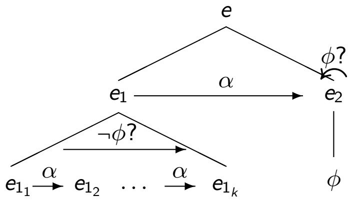
b. Mary walked to the store.

$build(x,z,y)$	$build(x,z,y)^+$	$build(x,z,y), y = v$
$\neg staircase(v)$		$staircase(v)$

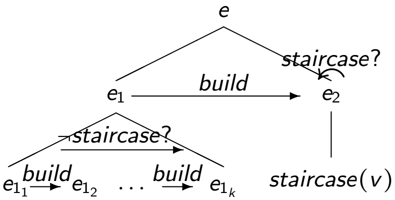
$_{\langle i,j \rangle}$

Table: Accomplishment: parallel tracks of changes

Dynamic Event Structure for Accomplishment



Parallel Scales define an Accomplishment



Event Locus and Spatial Aspect 1/4

- Encoding locations is generally not part of the grammatical system of a language (cf. Ritter and Wiltschko, 2005, Deal, 2008)
- Locating an event in the spatial domain is referential (except for deictic spatial morphology).
- We will distinguish between an event locus and its spatial aspect.

Event Locus and Spatial Aspect 2/4

- l_e: Event Locus: similar to Event Time in Reichenbach.
  it is a referential partition over the Spatial Domain, $\mathcal{D}_{\mathcal{S}}$.
  John walked.
- l_r: Spatial Aspect: a binary partitioning relative to this first partition. Similar to Reference Time.

Event Locus and Spatial Aspect 3/4

Sources of Spatial Aspect in Motion Verbs:

- ANALYTIC ASPECT: verb selects a spatial argument;
  Mary left *the room*.
  John entered *the hall*.
- SYNTHETIC ASPECT: verb is modified through PP adjunction;
  Mary swam *in the pool*.
  John walked *to the corner*.

Event Locus and Spatial Aspect 4/4

- Simple Locus: $l_e = l_r$.
  John **walked**$_{l_e, l_r}$.
- Relative Aspect: $l_e <_d l_r$.
  John **walked**$_{l_e}$ under the tree$_{l_r}$.
- Embedded Aspect: $l_e \subseteq l_r$.
  John **walked**$_{l_e}$ in the building$_{l_r}$.
- Completive Aspect: $\mathbf{EC}(l_e, l_r)$, $\mathbf{end}(l_r, \hat{p})$.
  John **arrived**$_{l_e}$ home$_{l_r}$.
  John **walked**$_{l_e}$ to the park$_{l_r}$
- Ingressive Aspect: $\mathbf{EC}(l_r, l_e)$, $\mathbf{begin}(l_r, \hat{p})$.
  John **walked**$_{l_e}$ from the park$_{l_r}$.

Event Localization

- r_{x_i}: The Kimian spatial extent of an object, x_i;
- $\hat{p}$: The path created by the motion in e;
- R_e: an embedding space (ES) for e, defined as a region containing $\hat{p}$ and r_{x_i} in a specific configuration, $\hat{p} \otimes r_{x_i}$;
- μ, the event locus: the minimum embedding space for e.
- Where μ can be defined as: $\forall e \forall R_e \forall \mu [[ES(R_e, e) \wedge Min(\mu, R_e)] \leftrightarrow [\mu \subseteq R_e \wedge \forall y [y \subseteq R_e \rightarrow \mu \subseteq y]]]$.

Habitats and Simulations Pustejovsky (2013)

- Habitat: a representation of an object situated within a partial minimal model; Enhancements of the qualia structure.
- With multi-dimensional affordances that determine how habitats are deployed and how they modify or augment the context.
- Compositional combinations of procedural (simulation) and operational (selection, specification, refinement) knowledge.
- A habitat:
  - embeds;
  - orients;
  - positions.

Teleotopology

- The function of space: the actions associated with a region or an object (inherently or opportunistically), i.e., Telic role values.
- The space of function: the regions defined by the Telic actions performed by an agent, or supervenient on the Telic state of an artifact, teleotopology.

Extending Qualia to Modeling Affordances

The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. It implies the complementarity of the animal and the environment. (J. J. Gibson, 1979/1986)

- Gibson (1979), Turvey (1992), Steedman (2002), Sahin et al (2007), Krippendorff (2010);
- Affordance: a correlation between an agent who acts on an object with a systematic or prototypical effect.

Semantics of Function and Purpose

There are two levels of accessibility that can be identified in a Telic role value, as illustrated below.

a. local modality: the conditions under which the activity can be performed on the object;

b. global modality: what is done with the object, and the resulting state.

Telic Qualia Role as part of Affordance Structure

Motivation for Qualia relations comes from the idea that there is a *hidden event* in the lexical representation associated with nouns denoting objects made for a particular purpose:

a. a door is for walking through
b. a window is for seeing through
c. a book is for reading
d. a beer is for drinking
e. a cake is for eating
f. a car is for driving
g. a table is for putting things on
h. a desk is for working on
i. a pen is for writing with

Telic Values and Affordances

$\mathcal{C} \rightarrow [\pi]\mathcal{R}$

The TELIC of *sandwich*:

$$\lambda x \begin{bmatrix} \textbf{sandwich} \\ \text{AS} = \begin{bmatrix} \text{ARG}1 = x : e \end{bmatrix} \\ \text{QS} = \begin{bmatrix} \text{F} = phys(x) \\ \text{T} = \lambda y \lambda e[\mathcal{C} \rightarrow [eat(e, y, x)]\mathcal{R}_{eat}(x)]] \\ \text{A} = \exists z[make(z, x)] \end{bmatrix} \end{bmatrix}$$

Habitat Theory

$$\lambda x \exists y \begin{bmatrix} \textbf{chair} \\ \text{AS} = \begin{bmatrix} \text{ARG1} = x : e \end{bmatrix} \\ \text{QS} = \begin{bmatrix} \text{F} = phys(x) \\ \text{T} = \lambda z, e[sit\_in(e, z, x)] \end{bmatrix} \end{bmatrix}$$

$$\lambda x \begin{bmatrix} \textbf{chair}_{hab} \\ \\ \text{F} = [phys(x), on(x, y_1), in(x, y_2), orient(x, up)] \\ \text{C} = [seat(x_1), back(x_2), legs(x_3), clear(x_1)] \\ \text{T} = \lambda z \lambda e[\mathcal{C} \rightarrow [sit(e, z, x)]\mathcal{R}_{sit}(x)] \\ \text{A} = [made(e', w, x)] \end{bmatrix}$$

Visual Object Concept Modeling Language (VoxML)

Pustejovsky and Krishnaswamy (2014, 2016)

- Modeling language for constructing 3D visualizations of concepts denoted by natural language expressions
- Used as the platform for creating *multimodal semantic simulations*
- Encodes dynamic semantics of events and objects and object properties
- Platform independent framework for encoding and visualizing linguistic knowledge.

Visual Object Concept (Voxeme)

- Object Geometry Structure:
  Formal object characteristics in R3 space
- Habitat: Embodied and embedded object:
  Orientation
  Situated context
  Scaling
- Affordance Structure:
  What can one do to it
  What can one do with it
  What does it enable
- Voxicon: library of voxemes

VoxML Elements

Entities modeled in VoxML can be:

- Objects: Physical objects (Nouns)
- Programs: Events (Verbs)
- Attributes: Properties (Adjectives)
- Functions: Quantifiers, connectives

These entities can then compose into visualizations of natural language concepts and expressions.

VoxML Concepts

- $\mathcal{E}$ — the minimal embedding space (MES)
- $\mathcal{E}_A$ — the axis A of the MES
- $loc(x)$ — location of object x
- $orient(x)$ — orientation of object x
- $vec(A)$ — vector denoted by axis A (+ by default)
- $opp(v)$ — opposite vector of v
- $reify(x, s)$ — relabel object x (a collection ($c_1,\ldots,c_n$)) as s
- $interior(x)$ — the interior surface (and volumetric enclosed space) of object x
- $exterior(x)$ — the exterior surface of object x
- $dimension(x)$ — the number of dimensions defining entity x
- $while(\phi, e)$ — operation e is executed as long as ϕ is true
- $for(x \in y)$ — following operation is executed for each x in y
- $align(A, B)$ – for vectors A, B, defines A as parallel with B

VoxML Template: Object

$$
\begin{bmatrix}
\textbf{OBJECT} \\
\text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{...} \\ \text{TYPE} = \textbf{...} \end{bmatrix} \\
\text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{...} \\ \text{COMPONENTS} = \textbf{...} \\ \text{CONCAVITY} = \textbf{...} \\ \text{ROTATSYM} = \{...\} \\ \text{REFLECTSYM} = \{...\} \\ \text{CONSTR} = \{...\} \end{bmatrix} \\
\text{HABITAT} = \begin{bmatrix} \text{INTR} = ... \\ \text{EXTR} = ... \end{bmatrix} \\
\text{AFFORD\_STR} = \begin{bmatrix} \text{A}_n = H_{[\#]} \to [E(a_{1..n})]R(a_{1..n}) \end{bmatrix} \\
\text{EMBODIMENT} = \begin{bmatrix} \text{SCALE} = \textbf{...} \\ \text{MOVABLE} = \textbf{...} \end{bmatrix}
\end{bmatrix}
$$

VoxML OBJECT is used for modeling nouns: 1/5

LEX	OBJECT's lexical information
TYPE	OBJECT's geometrical typing
HABITAT	OBJECT's habitat for actions
AFFORD_STR	OBJECT's affordance structure
EMBODIMENT	OBJECT's agent-relative embodiment

Objects 2/5

- The TYPE attribute contains information to define the object geometry in terms of primitives. HEAD is a primitive 3D shape that roughly describes the object's form or the form of the object's most semantically salient subpart.

HEAD	`prismatoid, pyramid, wedge, parallelepiped, cupola, frustum, cylindroid, ellipsoid, hemiellipsoid, bipyramid, rectangular_prism, toroid, sheet`

Objects 3/5

- COMPONENTS: subparts of the object
- CONCAVITY: `concave`, `flat`, or `convex`; refers to any concavity that deforms the HEAD shape.
- ROTATSYM (rotational symmetry) defines any of the three orthogonal axes around which the object's geometry may be rotated for an interval of less than 360 degrees and retain identical form as the unrotated geometry.
- REFLECTSYM (Reflectional symmetry): If an object may be bisected by a plane defined by two of the three orthogonal axes and then reflected across that plane to obtain the same geometric form as the original object, it is considered to have reflectional symmetry across that plane.

Objects 4/5

HABITAT defines habitats INTRINSIC to the object, regardless of what action it participates in, such as intrinsic orientations or surfaces, as well as EXTRINSIC habitats which must be satisfied for particular actions to take place.

Objects 5/5

AFFORD_STR describes the set of specific actions, along with the requisite conditions, that the object may take part in. There are low-level affordances, called GIBSONIAN, which involve manipulation or maneuver-based actions (grasping, holding, lifting, touching); there are also TELIC affordances, which link directly to what goal-directed activity can be accomplished, by means of the GIBSONIAN affordances.

EMBODIMENT qualitatively describes the SCALE of the object compared to an in-world agent (typically assumed to be a human) as well as whether the object is typically MOVABLE by that agent.

Plate

$$
\left[\begin{array}{l}
\textbf{plate} \\
\text{LEX} = \left[\begin{array}{l} \text{PRED} = \textbf{plate} \\ \text{TYPE} = \textbf{physobj} \end{array}\right] \\
\text{TYPE} = \left[\begin{array}{l} \text{HEAD} = \textbf{sheet} \\ \text{COMPONENTS} = \textbf{surface, base} \\ \text{CONCAVITY} = \textbf{concave} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLECTSYM} = \{XY, YZ\} \end{array}\right] \\
\text{HABITAT} = \left[\begin{array}{l} \text{INTR} = [1]\left[\begin{array}{l} \text{UP} = \mathit{align}(Y, \mathcal{E}_Y) \\ \text{TOP} = \mathit{top}(+Y) \end{array}\right] \\ \text{EXTR} = \ldots \end{array}\right] \\
\text{AFFORD\_STR} = \left[\begin{array}{l} A_1 = H[1] \rightarrow [\mathit{put}(x,y)]\mathit{hold}(y,x) \\ A_2 = \ldots \\ A_3 = \ldots \end{array}\right] \\
\text{EMBODIMENT} = \left[\begin{array}{l} \text{SCALE} = < \textbf{agent} \\ \text{MOVABLE} = \textbf{true} \end{array}\right]
\end{array}\right]
$$

Plate

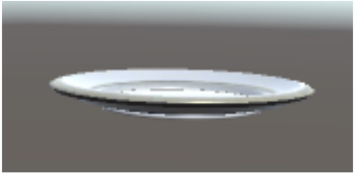


Figure: Plate voxeme instance

VoxML for cup

$$
\begin{bmatrix}
\textbf{cup} & \\
\text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{cup} \\ \text{TYPE} = \textbf{physobj} \end{bmatrix} & \\
\text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{cylindroid[1]} \\ \text{COMPONENTS} = \textbf{surface,interior} \\ \text{CONCAVITY} = \textbf{concave} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLECTSYM} = \{XY, YZ\} \end{bmatrix} & \\
\text{HABITAT} = \begin{bmatrix} \text{INTR} = [2] \begin{bmatrix} \text{UP} = align(Y, \mathcal{E}_Y) \\ \text{TOP} = top(+Y) \end{bmatrix} \\ \text{EXTR} = \ldots \end{bmatrix} & \\
\text{AFFORD\_STR} = \begin{bmatrix} A_1 = H[2] \rightarrow [put(x, on([1]))]support([1], x) \\ A_2 = H[2] \rightarrow [put(x, in([1]))]contain([1], x) \\ A_3 = H[2] \rightarrow [grasp(x, [1])] \end{bmatrix} & \\
\text{EMBODIMENT} = \begin{bmatrix} \text{SCALE} = \textbf{<agent} \\ \text{MOVABLE} = \textbf{true} \end{bmatrix} &
\end{bmatrix}
$$

VoxML for spoon

$$
\left[\begin{array}{l}
\textbf{spoon} \\
\text{LEX} = \left[\begin{array}{l} \text{PRED} = \textbf{spoon} \\ \text{TYPE} = \textbf{physobj, artifact} \end{array}\right] \\
\text{TYPE} = \left[\begin{array}{l} \text{HEAD} = \textbf{sheet[1]} \\ \text{COMPONENTS} = \textbf{handle[2], bowl[3]} \\ \text{CONCAVITY} = \textbf{concave} \\ \text{ROTATSYM} = \mathit{nil} \\ \text{REFLECTSYM} = \{YZ\} \end{array}\right] \\
\text{HABITAT} = \left[\begin{array}{l} \text{INTR} = [4]\left[\begin{array}{l} \text{CONSTR} = \{Z > X, Z \gg Y\} \\ \text{UP} = align(Y, \mathcal{E}_Y) \\ \text{FRONT} = top(+Y) \end{array}\right] \\ \text{EXTR} = [5]\left[\text{UP} = align(Y, \mathcal{E}_{\perp Y}) \right] \end{array}\right] \\
\text{AFFORD\_STR} = \left[\begin{array}{l} A_1 = H_{[4]} \rightarrow [put(x, in([3]))]contain([3], x) \\ A_2 = H_{[4]} \rightarrow [grasp(x, [2])] \\ A_1 = H_{[4]} \rightarrow [put([1], in(x))]contain(x, [1]) \\ A_1 = H_{[5]}, contain(x, [1]) \rightarrow [stir([1], x)] \end{array}\right]
\end{array}\right]
$$

VoxML for book

book

LEX = PRED = **book**; TYPE = **physobj, artifactj**

TYPE =
- HEAD = **rectangular_prism[1]**
- COMPONENTS = **cover[2]+, page[3]+**
- CONCAVITY = **flat**
- ROTATSYM = **nil**
- REFLECTSYM = $\{XY\}$

HABITAT =
- INTR = [4] UP = $align(Y, \mathcal{E}_Y)$; TOP = $front(+Y)$
- EXTR = ...

AFFORD_STR =
- $A_1 = H \rightarrow [grasp(x, [2]), move(x, [2], away(from([3])))]open(x, [1])$
- $A_2 = H \rightarrow [grasp(x, [2]), move(x, [2], toward([3]))]close(x, [1])$

EMBODIMENT =
- SCALE = <**agent**
- MOVABLE = **true**

VoxML Template: Program

$$\begin{bmatrix} \textbf{PROGRAM} & \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{...} \\ \text{TYPE} = \textbf{...} \end{bmatrix} & \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{...} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:a} \end{bmatrix} \\ \text{BODY} = \begin{bmatrix} \text{E}_n = E(a_{1..n}) \end{bmatrix} \end{bmatrix} & \end{bmatrix}$$

Programs are used for modeling verbs

LEX	PROGRAM's lexical information
TYPE	PROGRAM's event typing
EMBEDDING_SPACE	PROGRAM's embodiment as a function of the participants and their changes over time

A PROGRAM's LEX attribute contains the subcomponents PRED, the lexeme predicate denoting the program, and TYPE, the program's type as given in a lexical semantic resource, e.g., its GL type.

VoxML for put

$$\left[\begin{array}{l}\textbf{put}\\ \mathrm{LEX} = \left[\begin{array}{lcl}\mathrm{PRED} & = & \textbf{put}\\ \mathrm{TYPE} & = & \textbf{transition\_event}\end{array}\right]\\ \mathrm{TYPE} = \left[\begin{array}{lcl}\mathrm{HEAD} & = & \textbf{transition}\\ \mathrm{ARGS} & = & \left[\begin{array}{lcl}\mathrm{A}_1 & = & \textbf{x:agent}\\ \mathrm{A}_2 & = & \textbf{y:physobj}\\ \mathrm{A}_3 & = & \textbf{z:location}\end{array}\right]\\ \mathrm{BODY} & = & \left[\begin{array}{lcl}\mathrm{E}_1 & = & \mathit{grasp}(x,y)\\ \mathrm{E}_2 & = & [\mathit{while}(\\ & & \mathit{hold}(x,y),\\ & & \mathit{move}(y)]\\ \mathrm{E}_3 & = & [\mathit{at}(y,z) \rightarrow\\ & & \mathit{ungrasp}(x,y)]\end{array}\right]\end{array}\right]\end{array}\right]$$

VoxML for flip

$$
\left[\begin{array}{l}
\textbf{flip} \\
\text{LEX} = \left[\begin{array}{l} \text{PRED} = \textbf{flip} \\ \text{TYPE} = \textbf{transition\_event} \end{array}\right] \\
\text{TYPE} = \left[\begin{array}{l}
\text{HEAD} = \textbf{transition} \\
\text{ARGS} = \left[\begin{array}{l} \text{A}_1 = \textbf{x:agent} \\ \text{A}_2 = \textbf{y:physobj} \end{array}\right] \\
\text{BODY} = \left[\begin{array}{l}
\text{E}_1 = \mathit{def}(w, \mathit{as}(\mathit{orient}(y)))[\mathit{grasp}(x,y)] \\
\text{E}_2 = [\mathit{while}(\mathit{hold}(x,y), \mathit{rotate}(x,y)] \\
\text{E}_3 = [(\mathit{orient}(y) = \mathit{opp}(w)) \rightarrow \mathit{ungrasp}(x,y)]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

VoxML for in

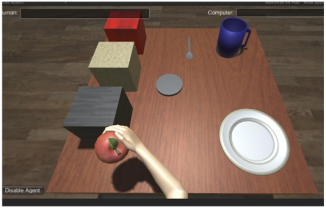
$$
\begin{bmatrix}
\textbf{in} & \\
\text{LEX} = & \begin{bmatrix} \text{PRED} = \textbf{in} \end{bmatrix} \\
\text{TYPE} = & \begin{bmatrix}
\text{CLASS} = \textbf{config} \\
\text{VALUE} = \textbf{ProperPart} \,||\, \textbf{PO} \\
\text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:3D} \\ \text{A}_2 = \textbf{y:3D} \end{bmatrix} \\
\text{CONSTR} = \textbf{...}
\end{bmatrix}
\end{bmatrix}
$$

Modeling Action in VoxML

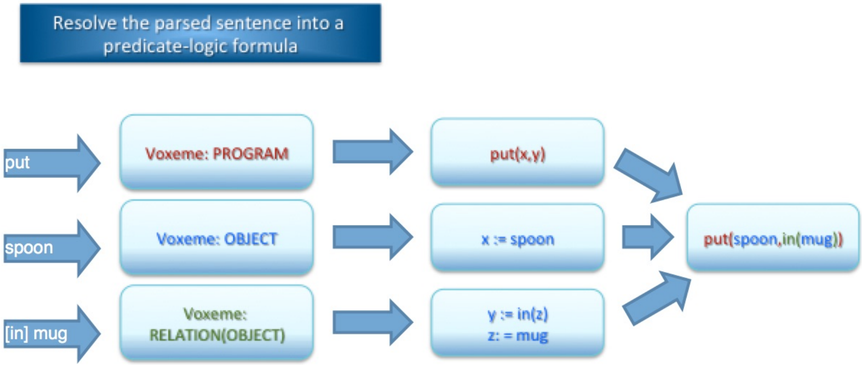
- Object Model: State-by-state characterization of an object as it changes or moves through time.
- Action Model: State-by-state characterization of an actor?s motion through time.
- Event Model: Composition of the object model with the action model.

Caused Motion as Rig Attachment

- Temporary parent-child relationship between joint on rig and manipulated object
- Allows agent and object to move together
- Object model + *Action model* = Event model



VoxSim Input



Type-driven Behavior and Constraints

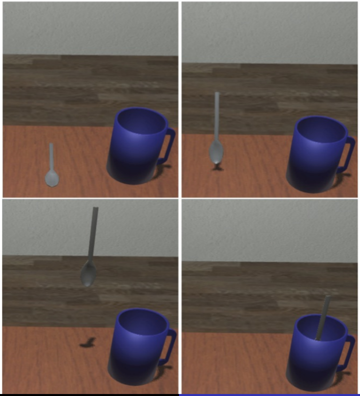
Each predicate is operationalized according to its type structure

$$put(spoon, in(mug))$$

- $in(z)$: takes object, outputs location
- $put(x, y)$: path verb
- $while(\neg at(y), move(x))$

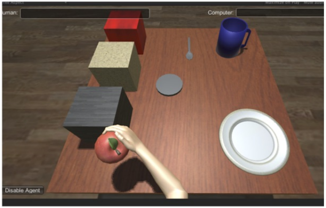
Type-driven Behavior and Constraints

- Can test be satisfied with current object configuration?
- Can test be satisfied by reorienting objects?
- Can test be satisfied at all?



Caused Motion as Rig Attachment

- Temporary parent-child relationship between joint on rig and manipulated object
- Allows agent and object to move together
- Object model + *Action model* = Event model



VoxSim

- https://github.com/VoxML/VoxSim

Architecture

- Built on Unity Game Engine
- NLP may use 3rd-party tools
- Art and VoxML resources loaded locally or from web server
- Input to UI or over network

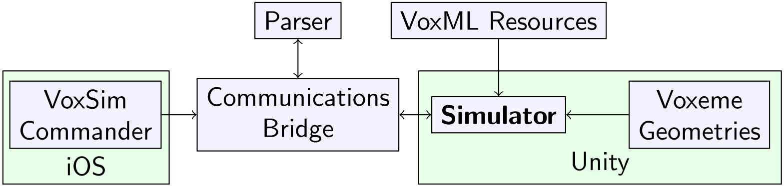
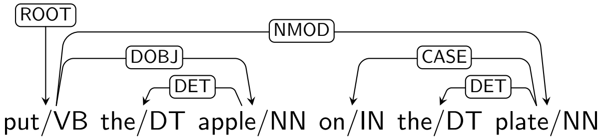


Figure: VoxSim architecture schematic

Processing Pipeline



1. $p := put(a[])$	5. $nmod := on(iobj)$
2. $dobj := the(b)$	6. $iobj := the(c)$
3. $b := (apple)$	7. $c := plate$
4. a.push($dobj$)	8. a.push($nmod$)
$put(the(apple), on(the(plate)))$	

Figure: Dependency parse for *Put the apple on the plate* and transformation to predicate-logic form.

Vocabulary is restricted for this tutorial

Flow of Control

1. Input sentence
2. Generate parse
3. Compute satisfaction conditions from voxeme composition

$$\begin{bmatrix} \textbf{put} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{process} \\ \text{ARGS} = \begin{bmatrix} A_1 = \textbf{agent} \\ A_2 = \textbf{physobj} \\ A_3 = \textbf{location} \end{bmatrix} \\ \text{BODY} = \begin{bmatrix} E_1 = grasp(A_1, A_2) \\ E_2 = [while(hold(A_1, A_2), \\ \quad move(A_2))] \\ E_3 = [at(A_1, A_3) \rightarrow \\ \quad ungrasp(A_1, A_2) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{in} \\ \text{TYPE} = \begin{bmatrix} \text{CLASS} = \textbf{config} \\ \text{VALUE} = \textbf{ProperPart} \parallel \textbf{PO} \\ \text{ARGS} = \begin{bmatrix} A_1 = \textbf{x:3D} \\ A_2 = \textbf{y:3D} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textbf{cup} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{cylindroid[1]} \\ \text{COMPONENTS} = \textbf{surface,interior} \\ \text{CONCAVITY} = \textbf{concave} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLECTSYM} = \{XY, YZ\} \end{bmatrix} \\ \text{HABITAT} = \begin{bmatrix} \text{INTR} = {}_{[2]}\begin{bmatrix} \text{UP} = align(Y, \mathcal{E}_Y) \\ \text{TOP} = top(+Y) \end{bmatrix} \end{bmatrix} \\ \text{AFFORD\_STR} = \begin{bmatrix} A_1 = H[2] \rightarrow \\ \quad [put(x, on([1]))] \\ \quad support([1], x) \\ A_2 = H[2] \rightarrow \\ \quad [put(x, in([1]))] \\ \quad contain([1], x) \\ A_3 = H[2] \rightarrow \\ \quad [grasp(x, [1])] \end{bmatrix} \end{bmatrix}$$

Flow of Control

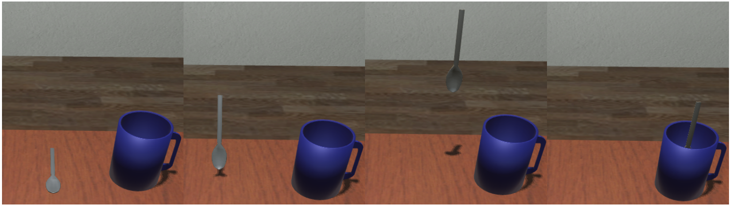


Figure: Object properties impose constraints on motion

4. Move object to target position
5. Update relationships between objects
6. Make or break parent-child rig-attachments
7. Resolve discrepancies between Unity physics bodies and voxemes

Object Model of Motion

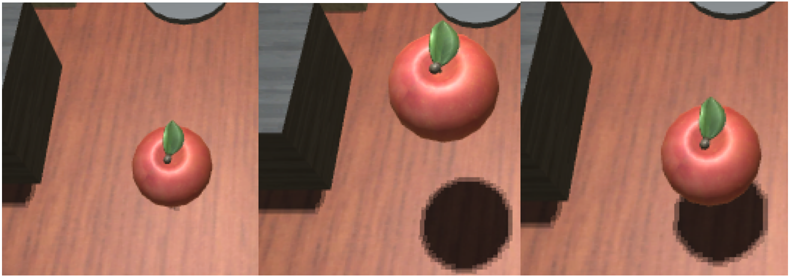


Figure: Object model of lifting and dropping an object

The **object model** of motion enacts the verbal program *only* on the objects involved, independent of any agent.

Object Model of Motion

- Requires reduction of subevent structure, informed by object affordances
  - e.g., $[[\text{PUT}]] = grasp(x, y)$, $[while(hold(x, y), move(x, y)]$, $at(y, z) \rightarrow ungrasp(x, y)$
  - H→[grasp(x, y)]hold(x, y)
  - H, hold(x, y)→[move(x, y)]
- Removing grasper removes "hold" condition and "ungrasp" subevent, $ungrasp(x, y) \rightarrow \neg move(x, y)$
- $at(y, z) \rightarrow \neg move(x, y) \implies \neg at(y, z) \vee \neg move(y)$ by CNF conversion
- One condition must be true, so "while" constraint holds
- $grasp(x, y)$, $[while(hold(x, y), move(x, y)]$, $at(y, z) \rightarrow ungrasp(x, y) \implies [while(\neg at(y, z), move(y))]$

Action Model of Motion

Figure: Action model of lifting and dropping an object

The **action model** factors out the objects, leaving a "pantomime" version of the event.

Action Model of Motion

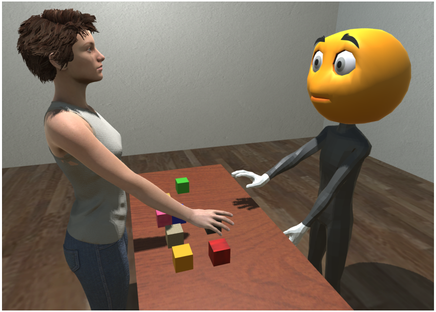


Figure: Sample gesture and object manipulation interaction

Action models can also be used to execute gestures as programs that operate over objects without affecting them.

Event Model of Motion

Figure: Event model of lifting and dropping an object

Objects can be attached to joints of an animated agent to make it move with the agent.

Event Model of Motion

- Programs enacted over the agent also affect the object
- Generalizes: trajectory from source to destination
- Specifies: how to grasp, how to calculate destination given object properties, current habitats, etc
- "Event model" = "object model" + "action model"
- Primitive behaviors (grasp, translocate, turn, etc.) can be composed into complex behaviors
  - roll, slide, flip, lean, etc.

Composition Example: LEAN (Object Model Only)

Theoretical formulation:

- Instruction: "Lean [[THEME]] on [[DEST]]"
- Goal: [[THEME]] is supported by [[DEST]] at an angle θ
  - For this example, assume $\theta = 45°$

1. Turn [[THEME]] such that major axis is θ off from +Y axis
2. Move [[THEME]] so it touches a side of [[DEST]]

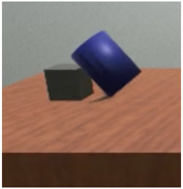


Figure: Desired goal state of "lean *x* on *y*"

Composition Example: LEAN (Object Model Only)

Operationalization:

- Instruction: "Lean [[THEME]] on [[DEST]]"
- Goal: [[THEME]] is supported by [[DEST]] at an angle θ
  - For this example, assume $\theta = 45°$
- Starting position of [[THEME]] is arbitrary
  - Not necessarily lying flat
  - Not necessarily axis-aligned
- 3D transformations take shortest path
  - Single rotation may result in unstable configuration

1. Turn [[THEME]] such that **minor axis** is $90°-\theta$ off from +Y axis
2. Turn [[THEME]] **about minor axis** such that major axis is θ off from +Y axis
3. Move [[THEME]] so it touches a side of [[DEST]]

Composition Example: LEAN (Object Model Only)

- Three types of primitive motions
  1. TURN-1: turn(x:**obj**,V$_1$:**axis**,$\mathcal{E}_{V_2}$:**axis**) — turn object x so that object axis V_1 is aligned with world axis V_2
  2. TURN-2: turn(x:**obj**,V$_1$:**axis**,$\mathcal{E}_{V_2}$:**axis**,$\mathcal{E}_{V_3}$:**axis**) — turn object x so that object axis V_1 is aligned with world axis V_2, constraining motion to around world axis V_3
  3. PUT: put(x:**obj**,y:**loc**) — put object x at location y

$$
\begin{bmatrix}
\textbf{lean} \\
\text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{lean} \\ \text{TYPE} = \textbf{transition\_event} \end{bmatrix} \\
\text{TYPE} = \begin{bmatrix}
\text{HEAD} = \textbf{transition} \\
\text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:agent} \\ \text{A}_2 = \textbf{y:physobj} \\ \text{A}_3 = \textbf{z:location} \end{bmatrix} \\
\text{BODY} = \begin{bmatrix}
\text{E}_1 = grasp(x,y) \\
\text{E}_2 = [while(hold(x,y), turn(x,y, \\ \quad align(minor(y)), \\ \quad \mathcal{E}_Y \times (90-\theta, about(\mathcal{E}_{\perp Y})))))] \\
\text{E}_3 = [while(hold(x,y), turn(x,y, \\ \quad align(major(y)), \\ \quad \mathcal{E}_Y \times (\theta, about(\mathcal{E}_{\perp Y}))), \\ \quad about(minor(y)))))] \\
\text{E}_4 = [while(hold(x,y), put(x,y))] \\
\text{E}_5 = [at(y,z) \rightarrow ungrasp(x,y)]
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Composition Example: LEAN (Object Model Only)

Result: "Lean the cup on the block"

VoxSim Summary

- Provides method for generating 3D visualizations using NL interface
- Provides platform to conduct experiments on observables of motion events
- Provides intuitive way to trace trace spatial cues and entailments through narrative
- Used to generate data on theoretical intuitions
- Enables broader study of event and motion semantics

VoxSim Summary

Voxeme Modeling from 3D Geometry Library

- Executables available at http://www.voxicon.net/eacl-2017/download-voxsim/

Voxeme Modeling from 3D Geometry Library

- Object voxemes consist of geometry + VoxML markup
- An object without VoxML markup contains no semantic information
- In a scene, we may have multiple instances of the same object
    - Different instances may have different properties
    - We need VoxML to reflect this

Behavior Attachment to a Voxeme

- Afforded behaviors require habitat conditions to be satisfied

$H_{[2]} \rightarrow [put(x, on[1])]support([1], x)$ can be paraphrased as "In habitat 2, x can be put on component 1, resulting in component 1 supporting x"
$H_{[3]} \rightarrow [grasp(x, [1])]$ can be paraphrased as "In habitat 3, component 1 can be grasped by x"
$H_{[4]}, grasp(x, [1]) \rightarrow [lift(x, [1])]$ can be paraphrased as "In habitat 4, if x is grasping component 1, component 1 can be lifted by x"

Adding Discriminating Attributes to Voxemes

- Discriminating attributes may be nominal, such as color
  - e.g., red, blue, green, black, etc.
- or sortal, such as relative location
  - e.g., leftmost, center, rightmost

Creating Novel Behavior

- "Switch the bottle and the block"
  - Interpretation: swap the locations of the bottle and the block in scene

$$\left[\begin{array}{l}\textbf{switch}\\ \text{LEX} = \left[\begin{array}{lcl}\text{PRED} & = & \textbf{switch}\\ \text{TYPE} & = & \textbf{transition\_event}\end{array}\right]\\ \text{TYPE} = \left[\begin{array}{lcl}\text{HEAD} & = & \textbf{transition}\\ \text{ARGS} & = & \left[\begin{array}{lcl}\text{A}_1 & = & \textbf{y:physobj}\\ \text{A}_2 & = & \textbf{z:physobj}\end{array}\right]\\ \text{BODY} & = & \left[\begin{array}{lcl}\text{E}_1 & = & \mathit{def}(\mathit{loc}(y), \mathit{as}('l1'))\\ \text{E}_2 & = & \mathit{def}(\mathit{loc}(z), \mathit{as}('l2'))\\ \text{E}_2 & = & \mathit{put}(y, \mathit{near}(z))\\ \text{E}_3 & = & \mathit{put}(z, l1))\\ \text{E}_4 & = & \mathit{put}(y, l2))\end{array}\right]\end{array}\right]\end{array}\right]$$

Creating Novel Behavior

- "Switch the bottle and the block"
    - Interpretation: swap the locations of the bottle and the block in scene

$$
\left[\begin{array}{l}
\textbf{switch} \\
\mathrm{LEX} = \left[\begin{array}{lcl} \mathrm{PRED} & = & \textbf{switch} \\ \mathrm{TYPE} & = & \textbf{transition\_event} \end{array}\right] \\
\mathrm{TYPE} = \left[\begin{array}{lcl}
\mathrm{HEAD} & = & \textbf{transition} \\
\mathrm{ARGS} & = & \left[\begin{array}{lcl} \mathrm{A}_1 & = & \textbf{y:physobj} \\ \mathrm{A}_2 & = & \textbf{z:physobj} \end{array}\right] \\
\mathrm{BODY} & = & \left[\begin{array}{lcl}
\mathrm{E}_1 & = & \mathit{def}(\mathit{loc}(y), \mathit{as}('l1')) \\
\mathrm{E}_2 & = & \mathit{def}(\mathit{loc}(z), \mathit{as}('l2')) \\
\mathrm{E}_2 & = & \mathit{slide}(y, \mathit{near}(z)) \\
\mathrm{E}_3 & = & \mathit{slide}(x, l1)) \\
\mathrm{E}_4 & = & \mathit{slide}(y, l2))
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Creating Novel Behavior

- Novel predicates can be composed from other novel predicates
- Novel predicates can be assigned arbitrary labels
- "Shuffle the bottle and the block"
  - Interpretation: iterated "switch"

$$\begin{bmatrix} \textbf{shuffle} & \\ \text{LEX} = & \begin{bmatrix} \text{PRED} = \textbf{shuffle} \\ \text{TYPE} = \textbf{process} \end{bmatrix} \\ \text{TYPE} = & \begin{bmatrix} \text{HEAD} = \textbf{process} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{y:physobj} \\ \text{A}_2 = \textbf{z:physobj} \end{bmatrix} \\ \text{BODY} = \begin{bmatrix} \text{E}_1 = \mathit{repeat}(n, '\mathit{switch}(y,z)') \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Future Work

- Modeling and Interpreting Gesture
- Semi-supervised Learning of Events
- Growing the Voxicon

Multimodal Semantics of Gesture

Given a Simulation Model, $\mathcal{M}_S$, the denotation of a multi-modal expression, E_{mm} is a function of the denotations of the component modal expressions, E_{m_i}, for all modalities, $i \in M$. For a linguistic expression, S, and a gesture, G:

$$[\![MME]\!]_{\mathbf{M},g} = f([\![S]\!], [\![G]\!])$$

a. G = (prep); (prestroke_hold); stroke; retract
b. $[\![stroke]\!] = [\![Ref(point)]\!]$

Cf. Wahlster (1998)
Lascarides and Stone (2009)

Gestures as Interpreted Programs

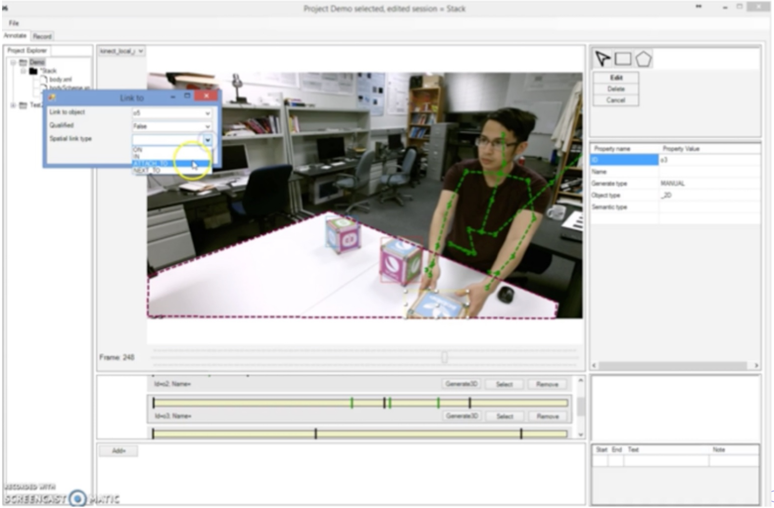
$$
\left[\begin{array}{l}
\textbf{GESTURE} \\
\text{LEX} = \left[\begin{array}{l} \text{PRED} = \ldots \\ \text{TYPE} = \ldots \\ \text{SEM} = \ldots \end{array}\right] \\
\text{TYPE} = \left[\begin{array}{l} \text{HEAD} = \ldots \\ \text{ARGS} = \left[\text{A}_1 = \textbf{x:a} \right] \\ \text{BODY} = \left[\text{E}_n = E(a_{1..n}) \right] \end{array}\right]
\end{array}\right]
$$

$$
\left[\begin{array}{l}
\textbf{palm\_converge} \\
\text{LEX} = \left[\begin{array}{l} \text{PRED} = \textbf{palm} \\ \text{TYPE} = \textbf{gesture} \\ \text{SEM} = \textbf{push\_together} \end{array}\right] \\
\text{TYPE} = \left[\begin{array}{l} \text{HEAD} = \textbf{transition} \\ \text{ARGS} = \left[\begin{array}{l} \text{A}_1 = \textbf{x:agent} \\ \text{A}_2 = \textbf{y:hand+} \\ \text{A}_3 = \textbf{z:finger+} \end{array}\right] \\ \text{BODY} = \left[\begin{array}{l} \text{E}_1 = [while([open\_flat(y) \wedge forward(z)], \\ \quad move(x, y, toward(center(z)))] \\ \text{E}_2 = [IN(y, interior(\mathcal{E})) \rightarrow \\ \quad move(x, y, toward(center(z)))] \end{array}\right] \end{array}\right]
\end{array}\right]
$$

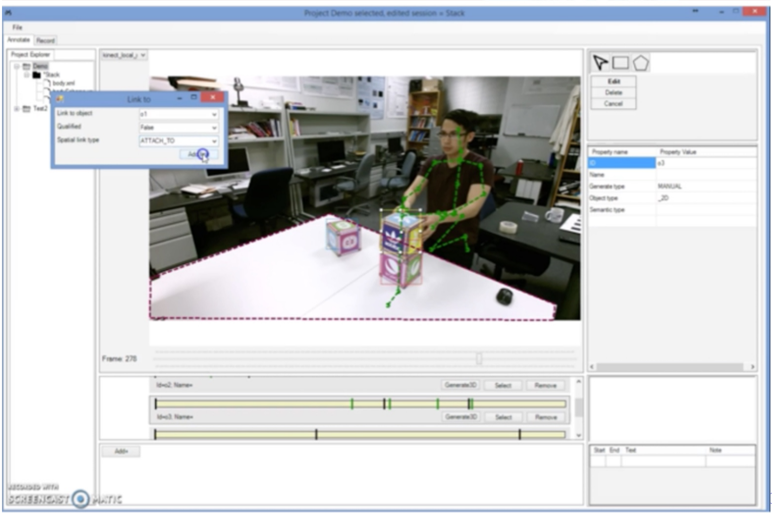
Event Aquisition

- Video annotation tool for marking up subevents
  - provides links to linguistic expression for event or activity
  - well suited for building a corpus of event-annotated multimodal simulations for use in the study of spatial and motion semantics
- Built on VoxML
  - Annotated video segments are immediately encoded in VoxML.
  - Provides initial scaffolding for rigs and objects in our simulations.

Event Aquisition



Event Aquisition



Event Aquisition

Learning to recognize motion events from 3D data

- Motion capture setup: 3 object setup
  - Rig: 3-d coordinates (in meter) of upper body of a performer.
  - Blocks: 2 blocks are detected and mapped into 3-d coordinates
- 15 descriptions, each is captured 30 times:
  - A pushes B across C
  - A pulls B to C
  - B slides to C
  - B rolls from C

Projected Voxicon: 4,000 voxemes

- 3,000 object voxemes (nouns)
- 500 program voxemes (verbs) with links to VerbNet
- 300 attribute voxemes (adjectives)
- 200 primitive relations (prepositions and stative verbs)
- Functional expressions (quantifiers, partitives, collections)
- Geometries: 512 objects complete
- Complete voxemes: 200 with affordances
- Attached behaviors: 40 distinct actions
- Animated human models: 3

THANK YOU!


Human:
Diana:
OK, I did it.
Thank you!
Reset
Disable Agent
Exit to Menu
Language Off
Help

References I

- Antol, Stanislaw et al. (2015). "Vqa: Visual question answering". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433.
- Bunt, Harry, Robbert-Jan Beun, and Tijn Borghuis (1998). *Multimodal human-computer communication: systems, techniques, and experiments*. Vol. 1374. Springer Science & Business Media.
- Chang, Angel et al. (2015). "Text to 3D Scene Generation with Rich Lexical Grounding". In: *arXiv preprint arXiv:1505.06289*.
- Chao, Yu-Wei et al. (2015a). "HICO: A benchmark for recognizing human-object interactions in images". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1017–1025.

References II

Chao, Yu-Wei et al. (2015b). "Mining semantic affordances of visual object categories". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4259–4267.

Clay, Sharon Rose and Jane Wilhelms (1996). "Put: Language-based interactive manipulation of objects". In: *IEEE Computer Graphics and Applications* 16.2, pp. 31–39.

Coyne, Bob and Richard Sproat (2001). "WordsEye: an automatic text-to-scene conversion system". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, pp. 487–496.

Jacko, Julie A (2012). *Human computer interaction handbook: Fundamentals, evolving technologies, and emerging applications.* CRC press.

References III

- Rautaray, Siddharth S and Anupam Agrawal (2015). "Vision based hand gesture recognition for human computer interaction: a survey". In: *Artificial Intelligence Review* 43.1, pp. 1–54.
- Seversky, Lee M and Lijun Yin (2006). "Real-time automatic 3D scene generation from natural language voice and text descriptions". In: *Proceedings of the 14th ACM international conference on Multimedia*. ACM, pp. 61–64.
- Siskind, Jeffrey Mark (2001). "Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic". In: *J. Artif. Intell. Res.(JAIR)* 15, pp. 31–90.
- Turk, Matthew (2014). "Multimodal interaction: A review". In: *Pattern Recognition Letters* 36, pp. 189–195.