The Semantics of Doing and Seeing in Multimodal Simulations

James Pustejovsky
Brandeis University

April 10, 2017
University of Colorado Boulder
Linguistics Seminar

Talk Outline

- Simulations and Mental Models
- Computational model for events, focusing on motion verbs
- Locating Events Spatially
- Adding Dynamics to Event Semantics
- VoxML: Visual Modeling Language
- Synthetic Vision
- Conclusions

Talk Outline

- Simulations and Mental Models
- Computational model for events, focusing on motion verbs
- Locating Events Spatially
- Adding Dynamics to Event Semantics
- VoxML: Visual Modeling Language
- Synthetic Vision
- Conclusions

Joint work with my group at Brandeis: Nikhil Krishnaswamy, Tuan Do, Keigh Rim, Gitit Kehat, Marc Verhagen, and colleagues Martha Palmer, Annie Zaenen, and Susan Brown.

Starting Assumptions

- Lexical meaning involves some sort of "componential analysis", either through predicative primitives or a system of types.
- The selectional properties of predicators can be explained in terms of these components;
- An understanding of event semantics and the different role of event participants seems crucial for modeling linguistic utterances.

What is the Mental Representation of an Utterance?

- Interpreted Logical Form
- Network of concepts
- Activation of specific brain areas
- Mental simulation

Encoded as:

- Formal Models
- Mental Models
- Frames

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.

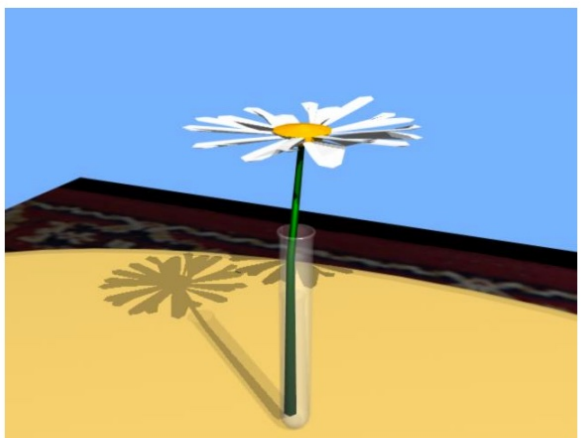


Figure 7: *The daisy is in the test tube.*

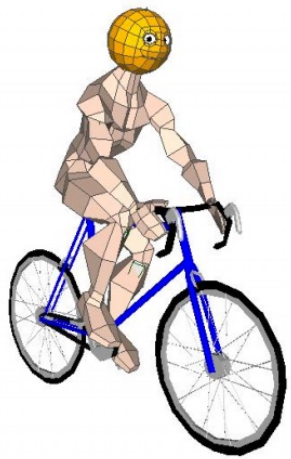
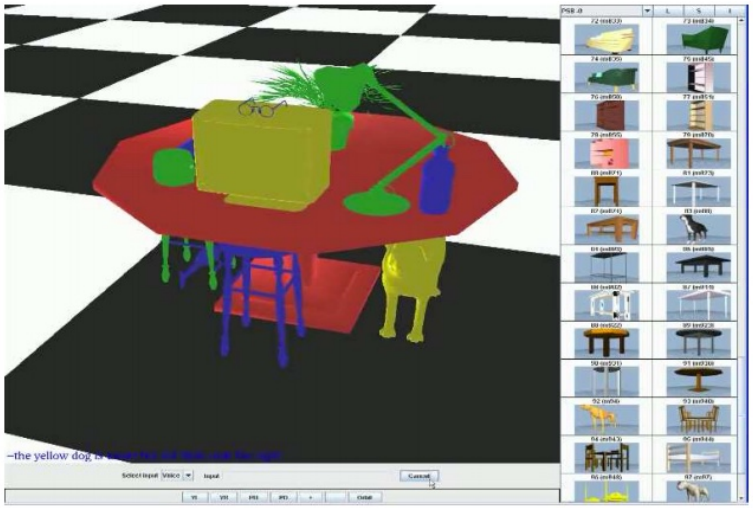


Figure 9: Usage pose for a 10-speed.

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

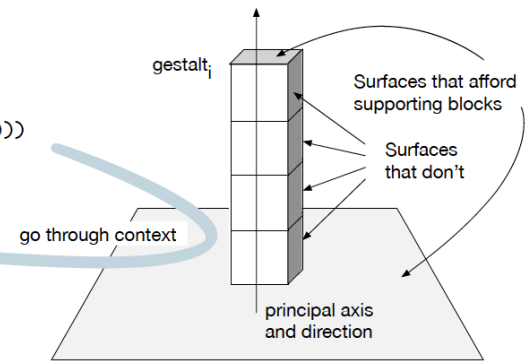


DARPA's Communicating with Computers (CwC)

add one more

"I want you to add one thing of a recent type"
```
want(me,act(you,add (?x:type(?x,recent) & number(?x,1),?y)))
```

"I want you to extend the gestalt formed by the blocks by adding a block"
```
want(me,
    act(you,
        achieve(
            extend(gestalt_i,
                ?x:type(?x,block) & number(?x,1)))))
```



Communication is the process by which an idea in one mind becomes an idea in another.

CwC makes four assumptions:

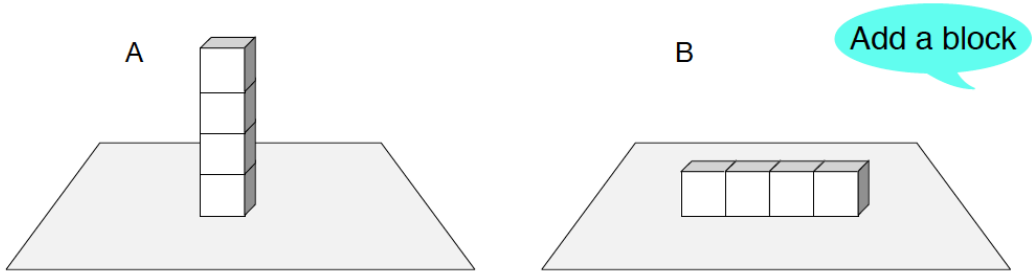
- Complex ideas are composed from elementary ones. There aren't many elementary ideas.
- Many, perhaps most, elementary ideas are about the physical world.
- Understanding language is equivalent to composing complex ideas from elementary ones.
- Language under-constrains the ideas we can compose (ambiguity); context adds constraints.

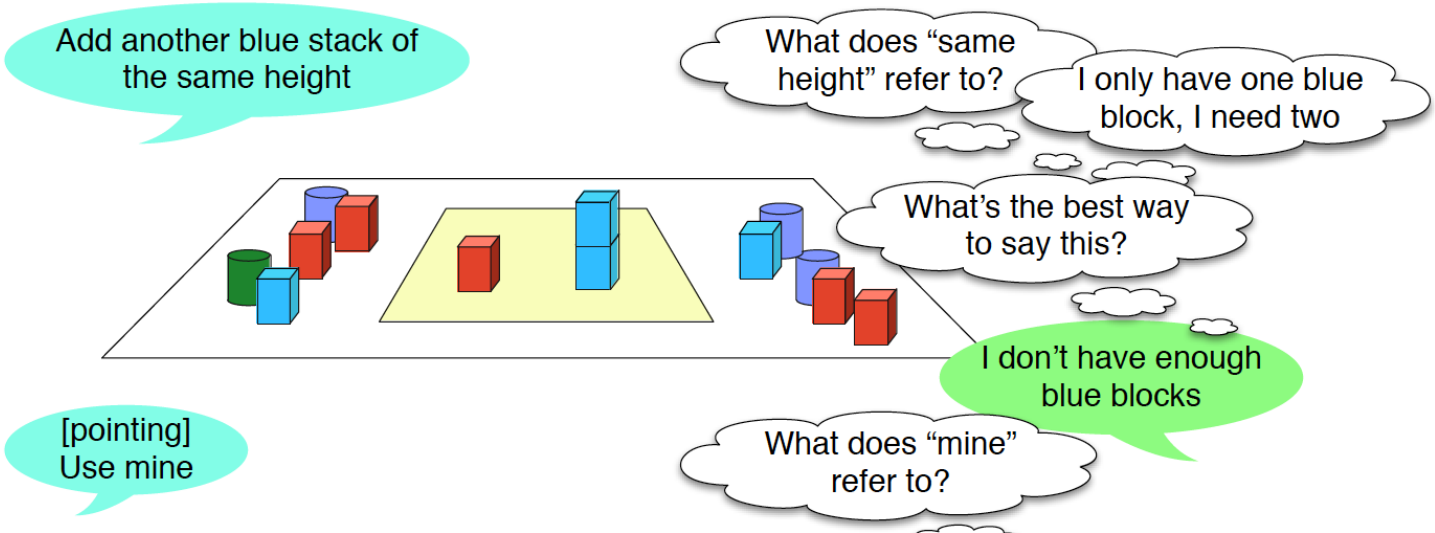
CwC will: 1) explicate the elementary ideas, 2) specify how language and context guide the composition of elementary ideas, 3) evaluate human-machine communication in Blocks World, Biocuration, and Collaborative Composition.

DARPA's CwC Goals

- To develop technology whereby machines can communicate with humans.
- To show that machines can be colleagues, not merely power tools or servants. Symmetric communication, in which the machine has ideas of its own to express.
- To embrace ambiguity and to develop methods by which context helps to resolve it.
- To get at the elementary composable ideas that all humans have, and to show that they underlie communication about blocks, biology, music and other domains.
- To test the idea that language is "directions" for building complex ideas in context.
- To give computational accounts of nonliteral, e.g., metaphorical, language.
- To incorporate gesture, facial expressions and other modalities into communication.
- To generalize the idea that communication is the process by which an idea in one mind becomes an idea in another, to music, to drawing, and other modalities.

Cooperative Goals through Discourse





Requirements on a Multimodal 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.

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.

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)

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.

Formal Requirements for Multimodal Models of Semantics

- Temporal Grounding: anchoring and ordering of events
- Event Localization: where the event takes place over time
- Internal structure of events: identifying preconditions, subevents, and resulting states
- Rich object semantics: qualia structure, affordances, habitats
- Capturing the dynamics of events: identifying how objects change during an event and what causes this change

Multimodal Modeling 1/2

- Objects have behaviors
  - Object-based knowledge includes habitats and affordances
  - Affordances are modeled dynamically as distinct event types
- Behaviors have agents
  - Behaviors and affordances are tied to agent embodiments
  - Behaviors are identified with participants of events
- Agents operate in contexts
  - Habitat: Situatedness of behavior dictates plans and actions
  - Context specifies and determines the values of communicative acts compositionally

Multimodal Modeling 2/2

- Generative Lexicon (GL): object and frame-based semantics that distributes compositionality
  - argument structure establishes the basic typing of an expression, along with directions for how it behaves in syntax.
  - event structure provides a scaffolding for anchoring events and action sequences
  - qualia structure organizes the applicable constraints and affordances associated with a concept, and how it integrates into an event.
- Dynamic Semantic Operations:
  - coercion: shift an expression's meaning to fit a selectional context
  - co-composition: multi-function application bringing about novel senses

What Properties do Events Have?

- Aspectual Type:
  state, process, achievement, accomplishment
- Semantic Type:
  action, motion, contact, change_of_state ...
- Participants :
  Agent, Patient, Theme, Goal, Source, Location, ...
- Temporal Anchoring or Ordering:
  before, equal, after, overlap, ...
- Modality and Evidentiality:
  future, necessary, possible, heard-of, seen, ...
- Principle of Individuation

When do Events Happen?

- Time as Modality: "add an operator"
  $P(happy(john))$
  (Prior, 1957, Kamp, 1968, Rescher and Urquhart, 1971, Montague, 1973, Tichý, 1971, Gabbay, 1989, etc.)
- Method of Temporal Arguments: "add a t"
  $\exists t[hungy(john, t) \wedge t < now]$
  (Russell, 1903, Kim, 1966, McCarthy and Hayes, 1969, Allen, 1983, etc.)

Interval Relations for Temporal Ordering

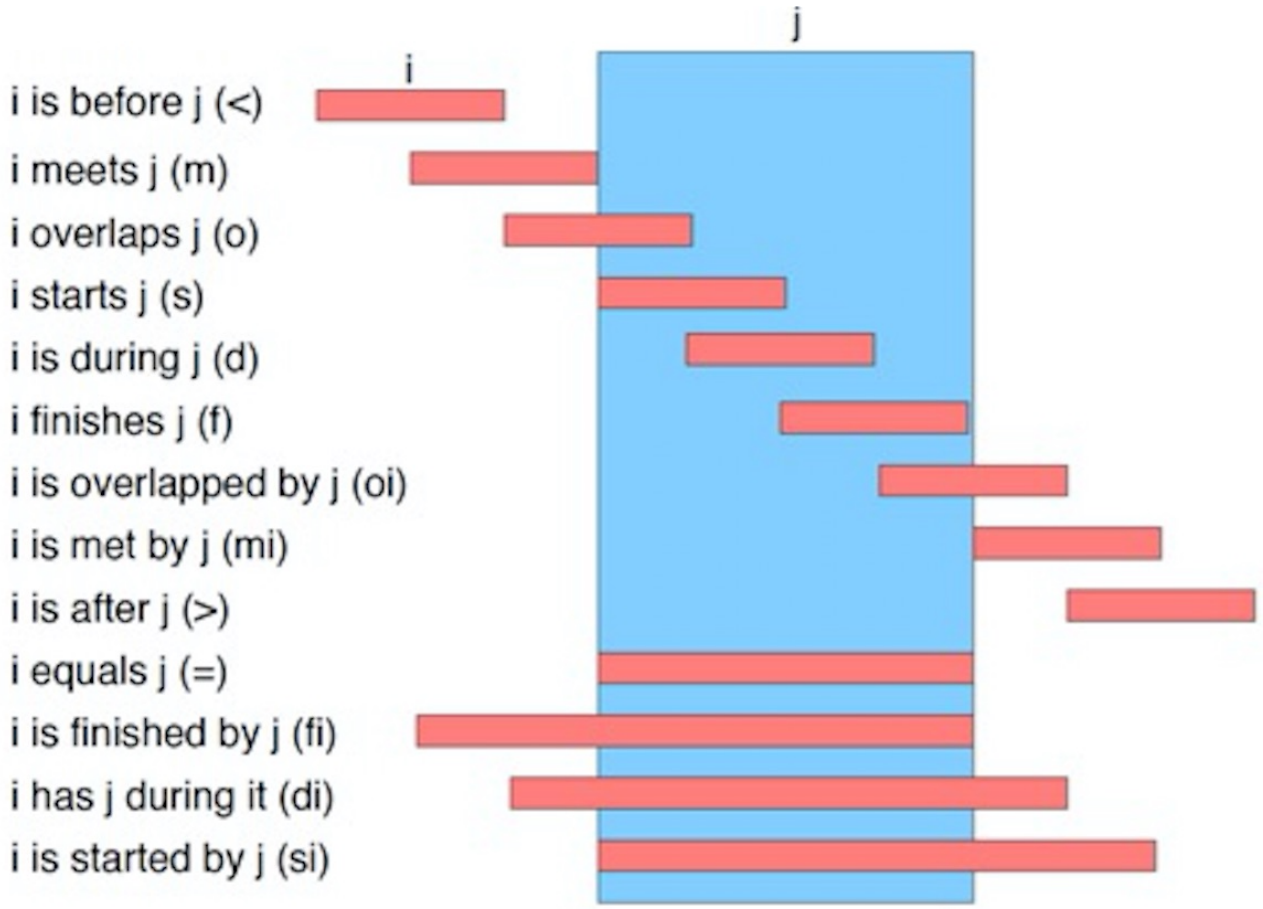


Figure: The interval relations as defined by Allen (1984)

Tense and Aspect 1/2

- Tense is a k-partitioning of the temporal domain, $\mathcal{D}_{\mathcal{T}}$.
  it is nominalized (past, present, future)
  and is ordered.
- Aspect is a binary partitioning relative to this first partition.
- Reichenbach's Reference time can be compared to Temporal Frames of Reference:
  (Moore, 2009, Tenbrink, 2011, Evans, 2012)

Tense and Aspect 2/2

- Simple Past: $E = R$, $R < S$.
  John **ate**$_{E,R}$ dinner.
- Past Perfect. $E < R$, $R < S$.
  John **had eaten**$_E$ dinner before noon$_R$.
- Past Progressive: $R \subseteq E$, $E < S$.
  John [**was eating**$_E$]$_R$ dinner.

Where do Events Happen

- The party was in the basement.
- The committee held a vote in the conference room.
- A poster is taped onto the wall.
- The dog walked on the carpet with his dirty paws.
- Sophie danced in her bedroom.

Not all Events are Situated

- Are there events that are timeless?
  *When is oxygen part of water?
- Are there events thare are spaceless?
  *Where did Obama win the Nobel Prize?

How Events Refer to Space

- Semantic Type: Position and Posture verbs: *stand*, *lean*, *hunch over*
- Argument Selection: *fill*, *wipe*, *cover*, *leave*, *enter*
  wipe the table, *erase the whiteboard*
  enter the room, *leave the party*

Spatial Properties of Events

- Mereotopological relations: touches, inside, disconnected
- Orientation (Projective): above, left-of, in front of
- Metric space: near, far
- Movement: walk, fall, leave
- Shape: curved, straight

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:

$$loc(e, l)$$

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

(1) 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)

(2) a. EVENT $\rightarrow$ STATE | PROCESS | TRANSITION
b. STATE: $\rightarrow$ e
c. PROCESS: $\rightarrow$ $e_1 \ldots e_n$
d. TRANSITION$_{ach}$: $\rightarrow$ STATE STATE
e. TRANSITION$_{acc}$: $\rightarrow$ PROCESS STATE

Dynamic Extensions to GL

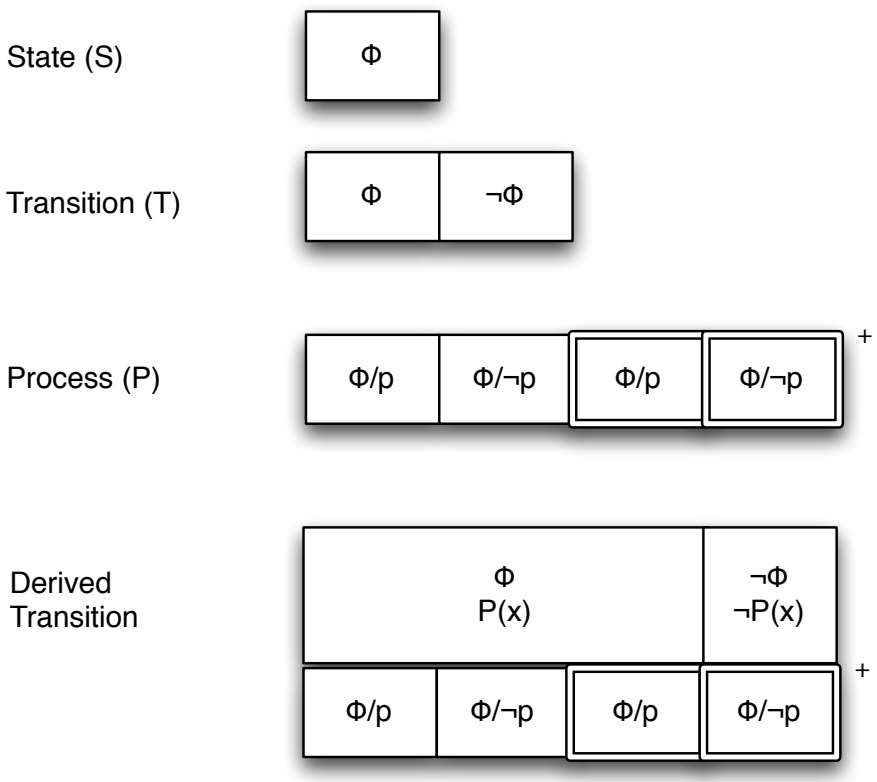
- Qualia Structure: Can be interpreted dynamically
- Dynamic Selection: Encodes the way an argument participates in the event
- Tracking change: Models the evolution of argument attributes

GL Feature Structure

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

Frame-based Event Structure

Pustejovsky (2009, 2013)



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).

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

(3) 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:

(4) $e_1 \xrightarrow{\alpha} e_3$

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:

(5) $\boxed{\phi} \xrightarrow{\alpha} \boxed{\neg\phi}$

State Transition

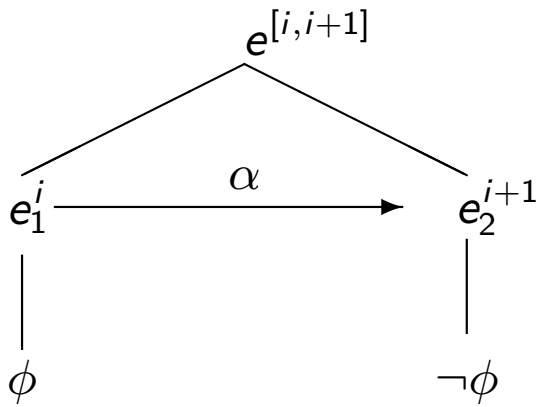
Frame-based representation:

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

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

Dynamic Event Structure

(8)

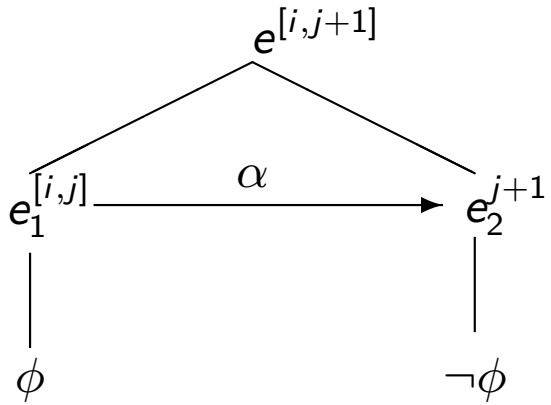


Dynamic Event Structure

(9) 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", $j + 1$.

(10)



Dynamic Event Structure

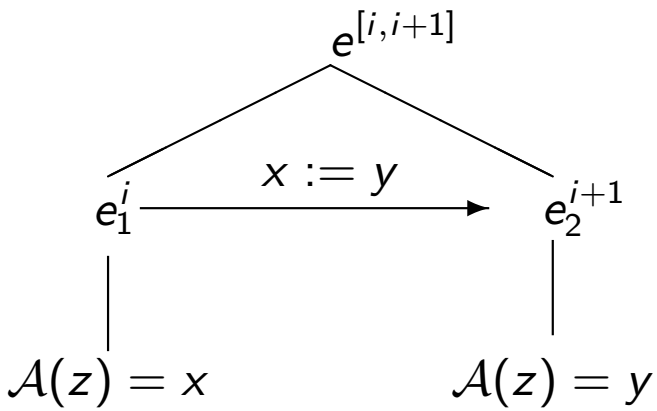
(11) $x := y$ (ν-transition)

"x assumes the value given to y in the next state."

$\langle \mathcal{M}, (i, i+1), (u, u[x/u(y)]) \rangle \models x := y$

iff $\langle \mathcal{M}, i, u \rangle \models s_1 \wedge \langle \mathcal{M}, i+1, u[x/u(y)] \rangle \models x = y$

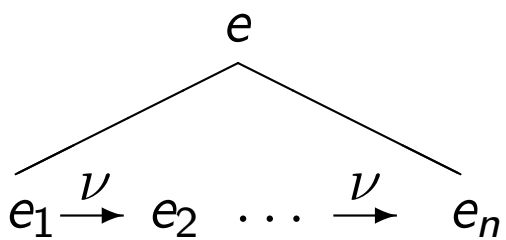
(12)



Processes

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

(13)



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

(14) 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

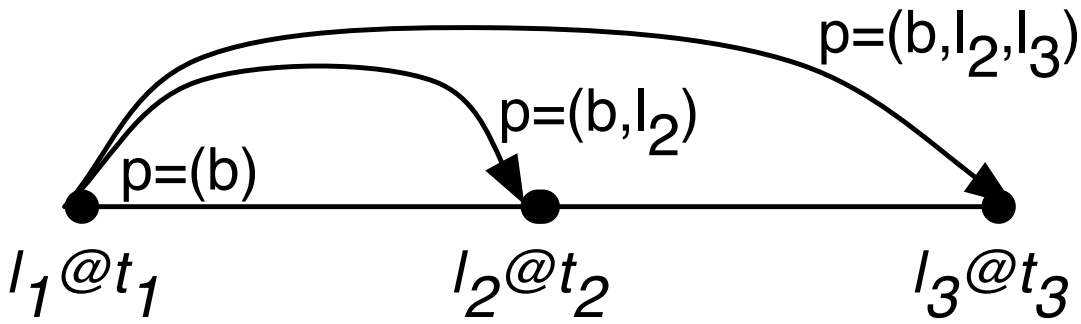


Figure: Directed Motion leaving a Trail

(15) 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

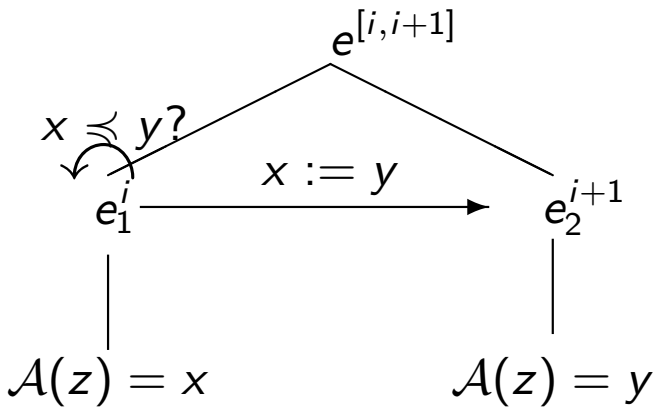
(16) $\overset{x \neq y?}{\curvearrowleft}\boxed{loc(z) = x}_{e_1} \xrightarrow{\nu} \boxed{loc(z) = y}_{e_2}$

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

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

Directed Motion

(18)



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

(19) a. John built a table.
b. Mary walked to the store.

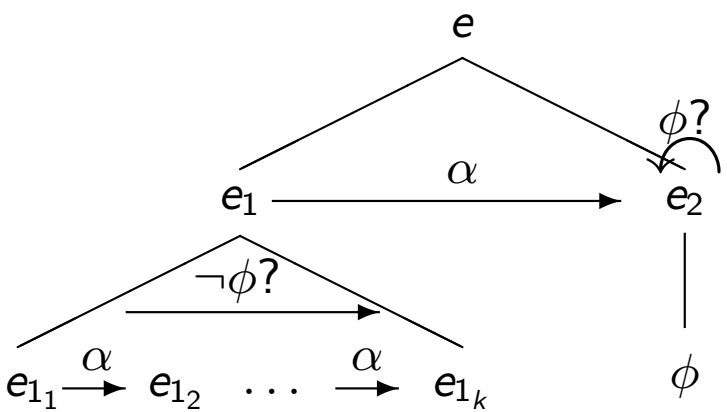
$build(x,z,y)$	$build(x,z,y)^{+}$	$build(x,z,y), y = v$
$\neg table(v)$		$table(v)$

$\langle i,j \rangle$

Table: Accomplishment: parallel tracks of changes

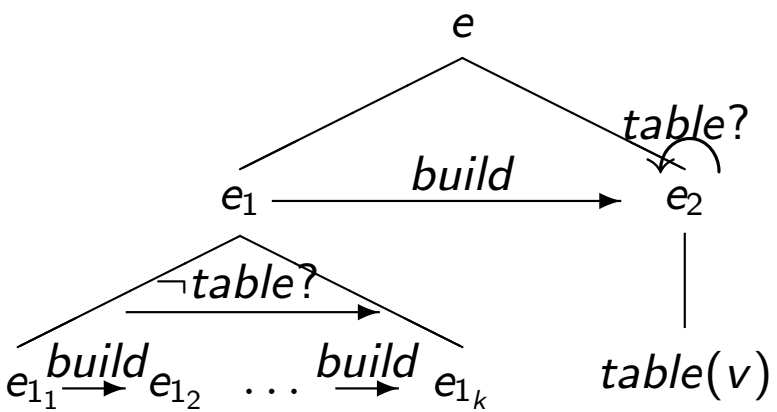
Dynamic Event Structure for Accomplishment

(20)



Parallel Scales define an Accomplishment

(21)



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, $\mathcal{D}_S$. 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: **EC**(l_e, l_r), **end**$(l_r, \hat{p})$.
  John **arrived**$_{l_e}$ home$_{l_r}$.
  John **walked**$_{l_e}$ to the park$_{l_r}$
- Ingressive Aspect: **EC**(l_r, l_e), **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: $\forall e \forall R_e \forall \mu [[ES(R_e, e) \wedge Min(\mu, R_e)] \leftrightarrow [\mu \subseteq R_e \wedge \forall y [y \subseteq R_e \rightarrow \mu \subseteq y]]]$.

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.

(22) 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:

(23) 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

(24) $\mathcal{C} \rightarrow [\pi]\mathcal{R}$

The TELIC of *sandwich*:

(25) $\lambda x \left[\begin{array}{l} \textbf{sandwich} \\ \text{AS} = \left[\text{ARG1} = x : e \right] \\ \text{QS} = \left[\begin{array}{l} \text{F} = \mathit{phys}(x) \\ \text{T} = \lambda y \lambda e[\mathcal{C} \rightarrow [\mathit{eat}(e, y, x)]\mathcal{R}_{\mathit{eat}}(x)]] \\ \text{A} = \exists z[\mathit{make}(z, x)] \end{array} \right] \end{array} \right]$

Habitats and Simulations

Pustejovsky (2013), Pustejovsky and Krishnaswamy (2014)

- 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.

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

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

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,...,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

(26)
$$\left[\begin{array}{l}\textbf{OBJECT} \\ \text{LEX} = \left[\begin{array}{l}\text{PRED} = \textbf{...} \\ \text{TYPE} = \textbf{...}\end{array}\right] \\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{...} \\ \text{COMPONENTS} = \textbf{...} \\ \text{CONCAVITY} = \textbf{...} \\ \text{ROTATSYM} = \{...\} \\ \text{REFLECTSYM} = \{...\} \\ \text{CONSTR} = \{...\}\end{array}\right] \\ \text{HABITAT} = \left[\begin{array}{l}\text{INTR} = ... \\ \text{EXTR} = ...\end{array}\right] \\ \text{AFFORD\_STR} = \left[\text{A}_n = H_{[\#]} \rightarrow [E(a_{1..n})]R(a_{1..n})\right] \\ \text{EMBODIMENT} = \left[\begin{array}{l}\text{SCALE} = \textbf{...} \\ \text{MOVABLE} = \textbf{...}\end{array}\right]\end{array}\right]$$

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

(27)
$$\left[\begin{array}{l}\textbf{plate} \\ \text{LEX} = \left[\begin{array}{l}\text{PRED} = \textbf{plate} \\ \text{TYPE} = \textbf{physobj}\end{array}\right] \\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{sheet} \\ \text{COMPONENTS} = \textbf{surface, base} \\ \text{CONCAVITY} = \textbf{concave} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLECTSYM} = \{XY, YZ\}\end{array}\right] \\ \text{HABITAT} = \left[\begin{array}{l}\text{INTR} = [1]\left[\begin{array}{l}\text{UP} = \mathit{align}(Y, \mathcal{E}_Y) \\ \text{TOP} = \mathit{top}(+Y)\end{array}\right] \\ \text{EXTR} = \ldots\end{array}\right] \\ \text{AFFORD\_STR} = \left[\begin{array}{l}\text{A}_1 = H[1] \rightarrow [\mathit{put}(x, y)]\mathit{hold}(y, x) \\ \text{A}_2 = \textbf{...} \\ \text{A}_3 = \textbf{...}\end{array}\right] \\ \text{EMBODIMENT} = \left[\begin{array}{l}\text{SCALE} = < \textbf{agent} \\ \text{MOVABLE} = \textbf{true}\end{array}\right]\end{array}\right]$$

Plate

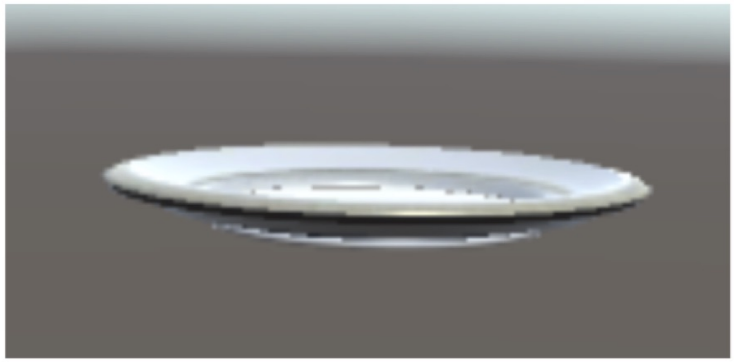


Figure: Plate voxeme instance

VoxML for cup

$$
\begin{bmatrix}
\textbf{cup} \\
\text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{cup} \\ \text{TYPE} = \textbf{physobj} \end{bmatrix} \\
\text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{cylindroid[1]} \\ \text{COMPONENTS} = \textbf{surface,interior} \\ \text{CONCAVITY} = \textbf{concave} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLECTSYM} = \{XY, YZ\} \end{bmatrix} \\
\text{HABITAT} = \begin{bmatrix} \text{INTR} = {}_{[2]}\begin{bmatrix} \text{UP} = align(Y, \mathcal{E}_Y) \\ \text{TOP} = top(+Y) \end{bmatrix} \\ \text{EXTR} = \ldots \end{bmatrix} \\
\text{AFFORD\_STR} = \begin{bmatrix} \text{A}_1 = H[2] \rightarrow [put(x, on([1]))]support([1], x) \\ \text{A}_2 = H[2] \rightarrow [put(x, in([1]))]contain([1], x) \\ \text{A}_3 = H[2] \rightarrow [grasp(x, [1])] \end{bmatrix} \\
\text{EMBODIMENT} = \begin{bmatrix} \text{SCALE} = \textbf{<agent} \\ \text{MOVABLE} = \textbf{true} \end{bmatrix}
\end{bmatrix}
$$

VoxML for spoon

(28)
$$\left[\begin{array}{l}\textbf{spoon}\\ \text{LEX} = \left[\begin{array}{l}\text{PRED} = \textbf{spoon}\\ \text{TYPE} = \textbf{physobj, artifact}\end{array}\right]\\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{sheet[1]}\\ \text{COMPONENTS} = \textbf{handle[2], bowl[3]}\\ \text{CONCAVITY} = \textbf{concave}\\ \text{ROTATSYM} = \textit{nil}\\ \text{REFLECTSYM} = \{YZ\}\end{array}\right]\\ \text{HABITAT} = \left[\begin{array}{l}\text{INTR} = [4]\left[\begin{array}{l}\text{CONSTR} = \{Z > X, Z \gg Y\}\\ \text{UP} = \textit{align}(Y, \mathcal{E}_Y)\\ \text{FRONT} = \textit{top}(+Y)\end{array}\right]\\ \text{EXTR} = [5]\left[\text{UP} = \textit{align}(Y, \mathcal{E}_{\perp Y})\right]\end{array}\right]\\ \text{AFFORD\_STR} = \left[\begin{array}{l}\text{A}_1 = H_{[4]} \rightarrow [\textit{put}(x, \textit{in}([3]))]\textit{contain}([3], x)\\ \text{A}_2 = H_{[4]} \rightarrow [\textit{grasp}(x, [2])]\\ \text{A}_1 = H_{[4]} \rightarrow [\textit{put}([1], \textit{in}(x))]\textit{contain}(x, [1])\\ \text{A}_1 = H_{[5]}\textit{contain}(x, [1]) \rightarrow [\textit{stir}([1], x)]\end{array}\right]\end{array}\right]$$

VoxML for book

(29)

$$\left[\begin{array}{l}\textbf{book} \\ \text{LEX} = \left[\begin{array}{l}\text{PRED} = \textbf{book} \\ \text{TYPE} = \textbf{physobj, artifactj}\end{array}\right] \\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{rectangular\_prism[1]} \\ \text{COMPONENTS} = \textbf{cover[2]+, page[3]+} \\ \text{CONCAVITY} = \textbf{flat} \\ \text{ROTATSYM} = \textbf{nil} \\ \text{REFLECTSYM} = \{XY\}\end{array}\right] \\ \text{HABITAT} = \left[\begin{array}{l}\text{INTR} = [4]\left[\begin{array}{l}\text{UP} = \mathit{align}(Y, \mathcal{E}_Y) \\ \text{TOP} = \mathit{front}(+Y)\end{array}\right] \\ \text{EXTR} = \ldots\end{array}\right] \\ \text{AFFORD\_STR} = \left[\begin{array}{l}\text{A}_1 = H \rightarrow [\mathit{grasp}(x, [2]), \\ \qquad \mathit{move}(x, [2], \mathit{away}(\mathit{from}([3])))]\mathit{open}(x, [1]) \\ \text{A}_2 = H \rightarrow [\mathit{grasp}(x, [2]), \\ \qquad \mathit{move}(x, [2], \mathit{toward}([3]))]\mathit{close}(x, [1])\end{array}\right]\end{array}\right]$$

VoxML Template: Program

(30)
$$\left[\begin{array}{l}\textbf{PROGRAM} \\ \text{LEX} = \left[\begin{array}{l}\text{PRED} = \textbf{...} \\ \text{TYPE} = \textbf{...}\end{array}\right] \\ \text{TYPE} = \left[\begin{array}{l}\text{HEAD} = \textbf{...} \\ \text{ARGS} = \left[\text{A}_1 = \textbf{x:a}\right] \\ \text{BODY} = \left[\text{E}_n = E(a_{1..n})\right]\end{array}\right]\end{array}\right]$$

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

(31)
$$\left[\begin{array}{l}\textbf{put} \\ \text{LEX} = \left[\begin{array}{l}\text{PRED} = \textbf{put} \\ \text{TYPE} = \textbf{transition\_event}\end{array}\right] \\ \text{TYPE} = \left[\begin{array}{ll}\text{HEAD} = & \textbf{transition} \\ \text{ARGS} = & \left[\begin{array}{l}\text{A}_1 = \textbf{x:agent} \\ \text{A}_2 = \textbf{y:physobj} \\ \text{A}_3 = \textbf{z:location}\end{array}\right] \\ \text{BODY} = & \left[\begin{array}{l}\text{E}_1 = \mathit{grasp}(x,y) \\ \text{E}_2 = [\mathit{while}(\mathit{hold}(x,y), \mathit{move}(y)] \\ \text{E}_3 = [\mathit{at}(y,z) \rightarrow \mathit{ungrasp}(x,y)]\end{array}\right]\end{array}\right]\end{array}\right]$$

VoxML for flip

(32)

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

VoxML for in

$$\left[\begin{array}{l} \textbf{in} \\ \text{LEX} = \left[\text{PRED} = \textbf{in}\right] \\ \text{TYPE} = \left[\begin{array}{l} \text{CLASS} = \textbf{config} \\ \text{VALUE} = \textbf{ProperPart} \,\|\, \textbf{PO} \\ \text{ARGS} = \left[\begin{array}{l} A_1 = \textbf{x:3D} \\ A_2 = \textbf{y:3D} \end{array}\right] \\ \text{CONSTR} = \textbf{...} \end{array}\right] \end{array}\right]$$

VoxML for Spatial Relations

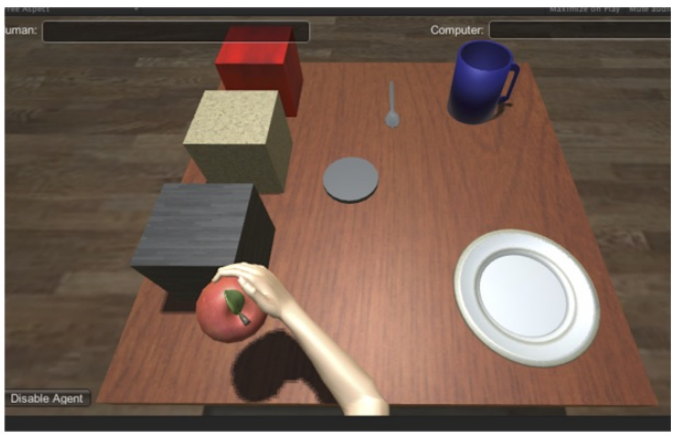
$$\left[\begin{array}{l} \textbf{in} \\ \text{LEX} = \left[\text{PRED} = \textbf{in}\right] \\ \text{TYPE} = \left[\begin{array}{l} \text{CLASS} = \textbf{config} \\ \text{VALUE} = \textbf{ProperPart} \,\|\, \textbf{PO} \\ \text{ARGS} = \left[\begin{array}{l} A_1 = \textbf{x:3D} \\ A_2 = \textbf{y:3D} \end{array}\right] \\ \text{CONSTR} = \textbf{...} \end{array}\right] \end{array}\right]$$

Modeling Action in VoxML

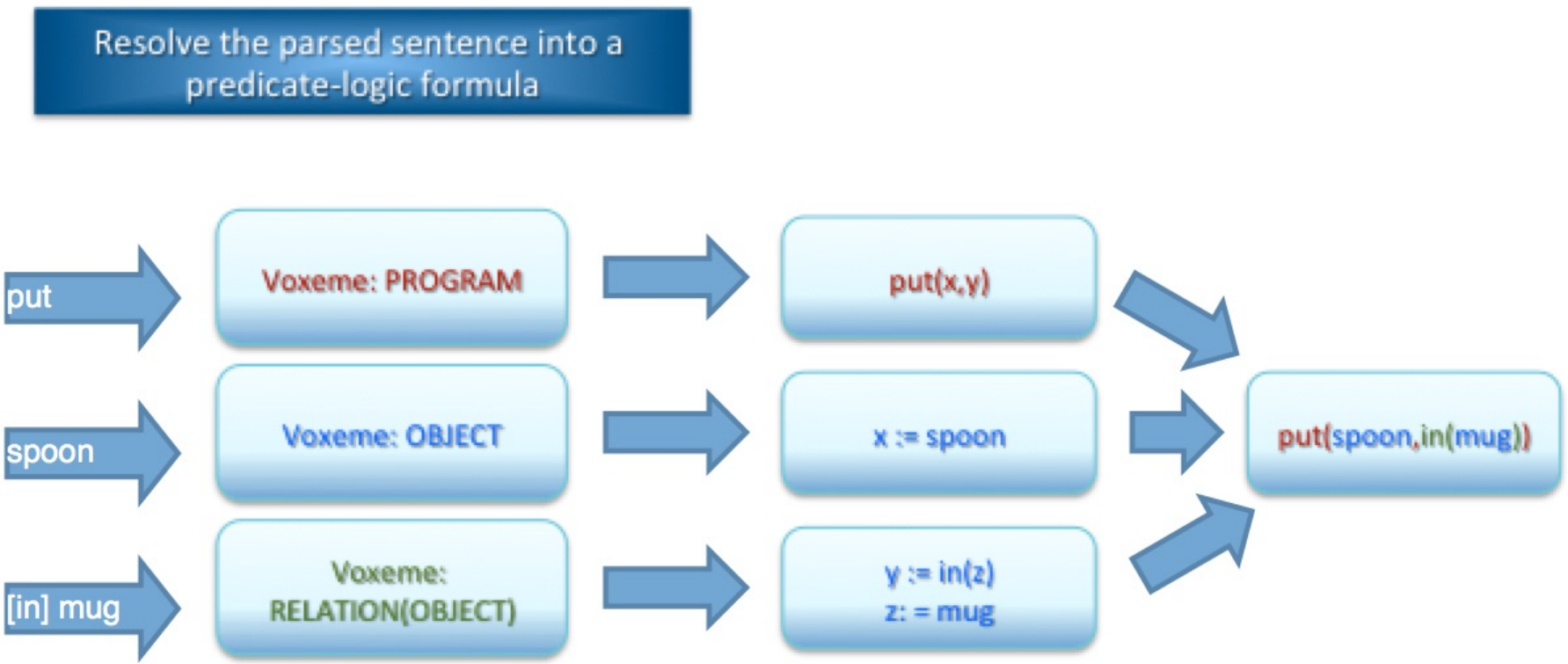
- Object Model: State-by-state characterization of an object as it changes or moves through time.
- Action Model: State-by-state characterization of an actor?s motion through time.
- Event Model: Composition of the object model with the action model.

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?

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

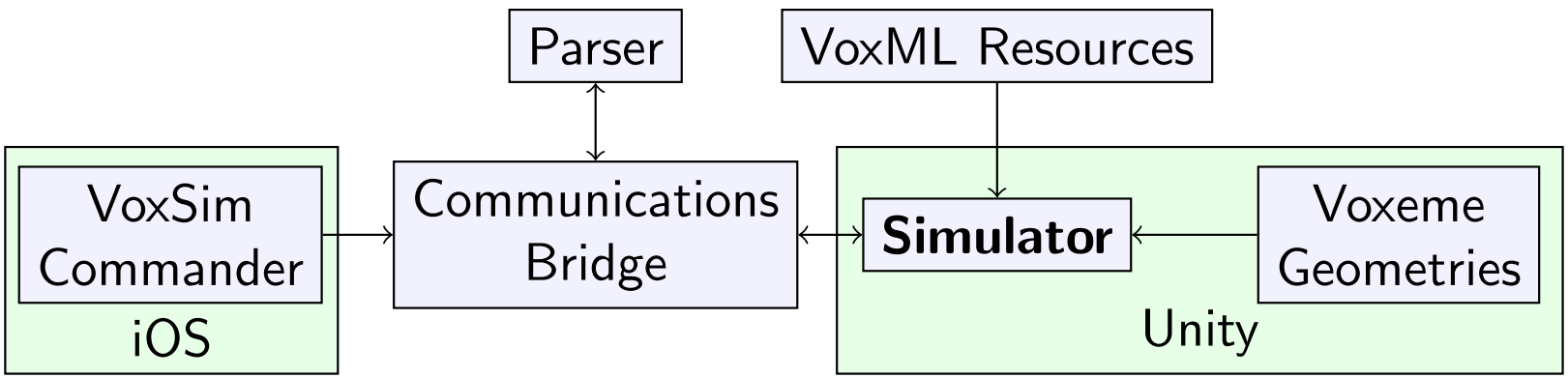
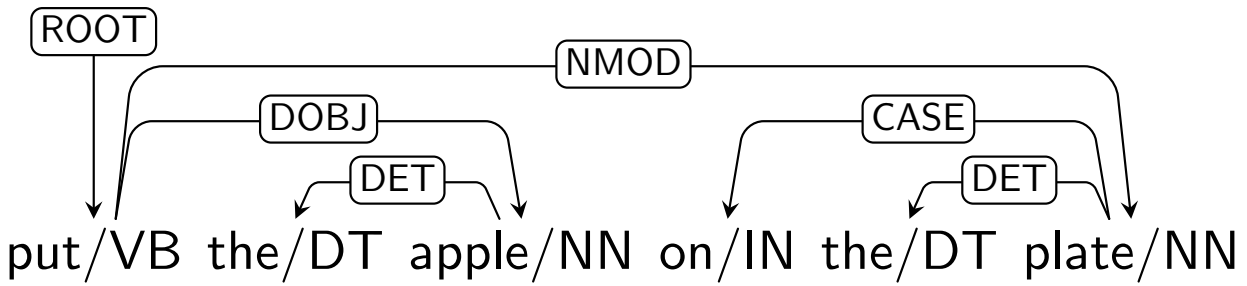


Figure: VoxSim architecture schematic

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.

Flow of Control

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

$$
\left[\begin{array}{l}
\textbf{put} \\
\text{TYPE} = \left[\begin{array}{ll}
\text{HEAD} = & \textbf{process} \\
\text{ARGS} = & \left[\begin{array}{ll} A_1 = & \textbf{agent} \\ A_2 = & \textbf{physobj} \\ A_3 = & \textbf{location} \end{array}\right] \\
\text{BODY} = & \left[\begin{array}{ll} E_1 = & grasp(A_1, A_2) \\ E_2 = & [while(hold(A_1, A_2), \\ & \quad move(A_2))] \\ E_3 = & [at(A_1, A_3) \rightarrow \\ & \quad ungrasp(A_1, A_2)) \end{array}\right]
\end{array}\right]
\end{array}\right]
\left[\begin{array}{l}
\textbf{in} \\
\text{TYPE} = \left[\begin{array}{ll}
\text{CLASS} = & \textbf{config} \\
\text{VALUE} = & \textbf{ProperPart} \parallel \textbf{PO} \\
\text{ARGS} = & \left[\begin{array}{ll} A_1 = & \textbf{x:3D} \\ A_2 = & \textbf{y:3D} \end{array}\right]
\end{array}\right]
\end{array}\right]
\left[\begin{array}{l}
\textbf{cup} \\
\text{TYPE} = \left[\begin{array}{ll}
\text{HEAD} = & \textbf{cylindroid[1]} \\
\text{COMPONENTS} = & \textbf{surface,interior} \\
\text{CONCAVITY} = & \textbf{concave} \\
\text{ROTATSYM} = & \{Y\} \\
\text{REFLECTSYM} = & \{XY, YZ\}
\end{array}\right] \\
\text{HABITAT} = \left[\begin{array}{ll} \text{INTR} = & [2]\left[\begin{array}{ll} \text{UP} = & align(Y, \mathcal{E}_Y) \\ \text{TOP} = & top(+Y) \end{array}\right] \end{array}\right] \\
\text{AFFORD\_STR} = \left[\begin{array}{ll} A_1 = & H[2] \rightarrow \\ & [put(x, on([1]))] \\ & support([1], x) \\ A_2 = & H[2] \rightarrow \\ & [put(x, in([1]))] \\ & contain([1], x) \\ A_3 = & H[2] \rightarrow \\ & [grasp(x, [1])] \end{array}\right]
\end{array}\right]
$$

Flow of Control

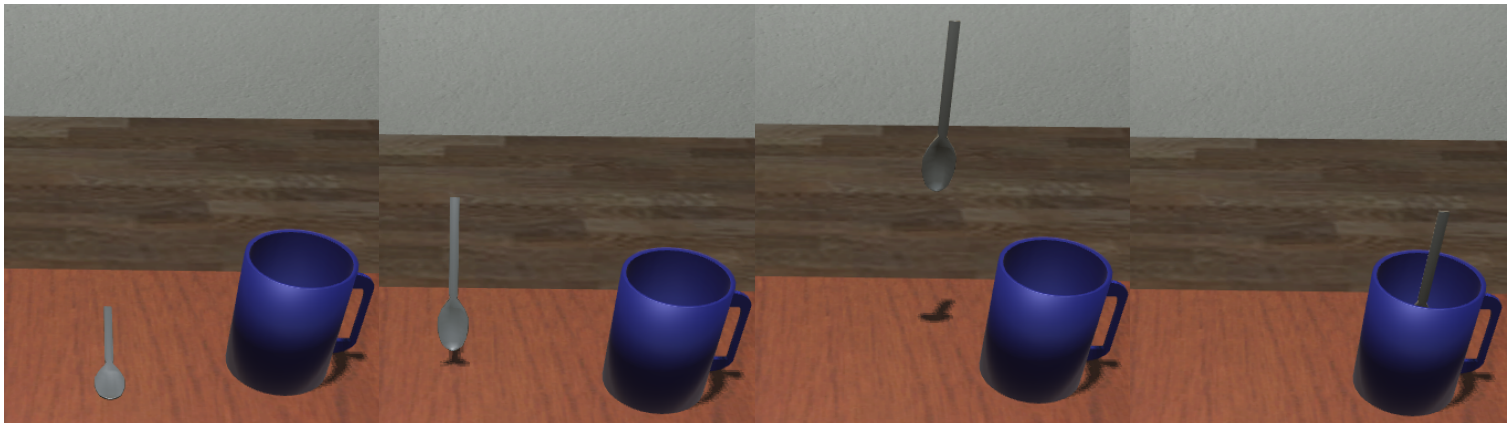


Figure: Object properties impose constraints on motion

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

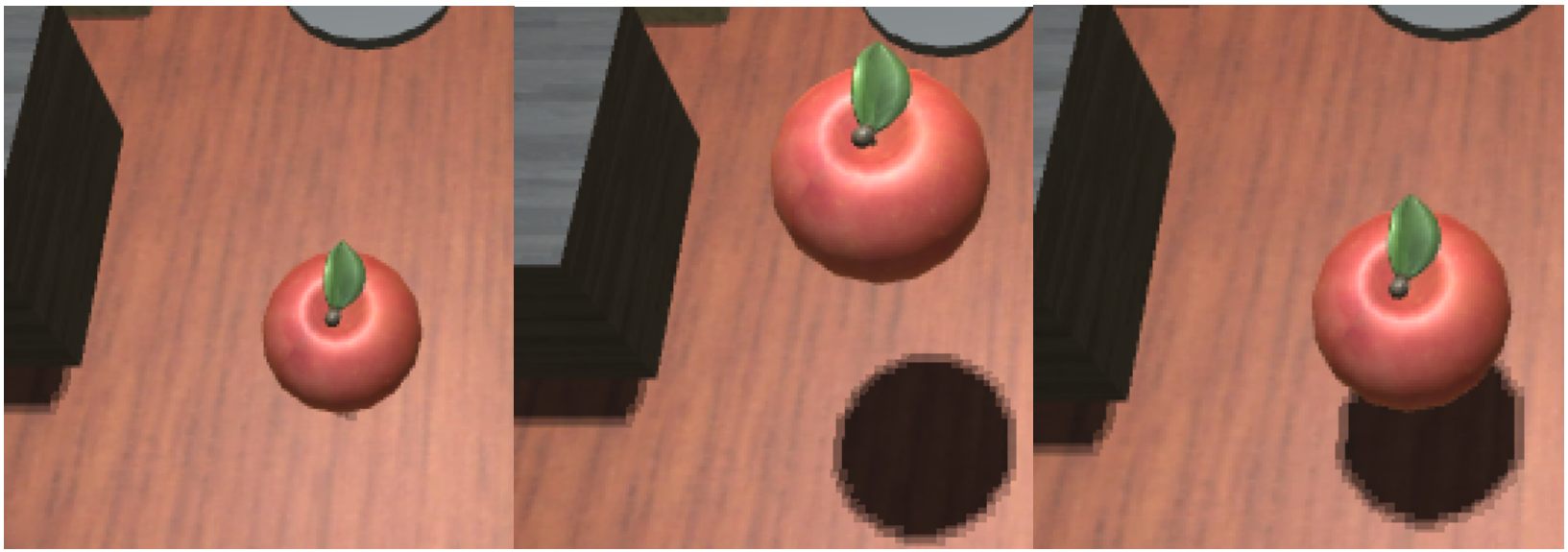


Figure: Object model of lifting and dropping an object

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$\rightarrow$[grasp(x, y)]hold(x, y)
  - H, hold(x, y)$\rightarrow$[move(x, y)]
- Removing grasper removes "hold" condition and "ungrasp" subevent, $ungrasp(x, y) \rightarrow \neg move(x, y)$
- $at(y, z) \rightarrow \neg move(x, y) \Longrightarrow \neg at(y, z) \vee \neg move(y)$ by CNF conversion
- One condition must be true, so "while" constraint holds
- $grasp(x, y)$, $[while(hold(x, y), move(x, y)]$, $at(y, z) \rightarrow ungrasp(x, y) \Longrightarrow [while(\neg at(y, z), move(y))]$

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

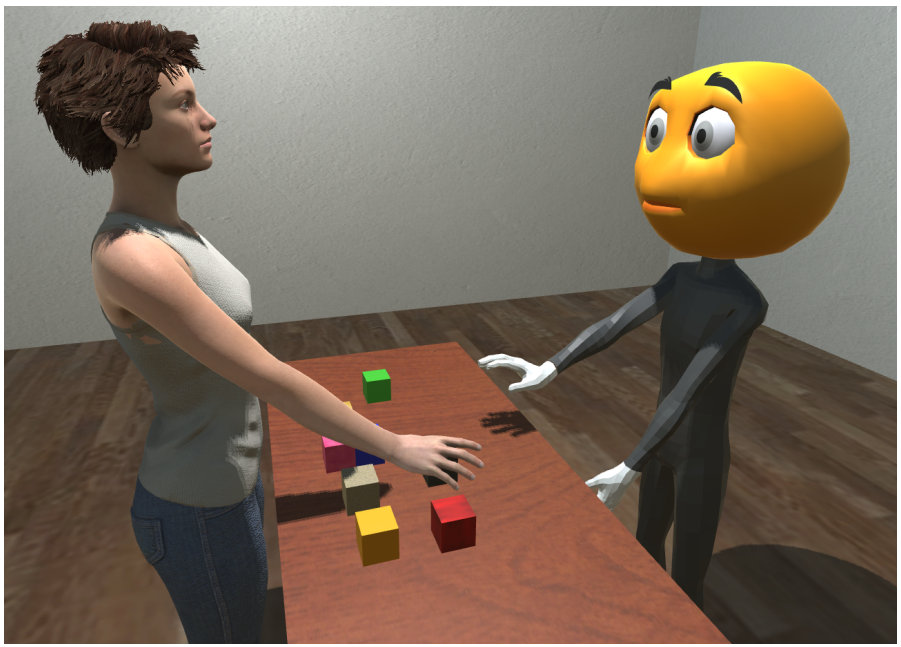


Figure: Sample gesture and object manipulation interaction

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.

The 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)

Aspects of Gesture Configuration

- Hand shape
- Palm orientation
- Finger orientation
- Hand location
- Hand movement
- Gesture is a function from *config* → *config*

Gesture as Interpreted Programs

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

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

Gesture Modeling with CSU Data (Bruce Draper)

Focus on four common gestures that do not require hand poses and appear in human subject studies both with and without audio communication. The gestures are:

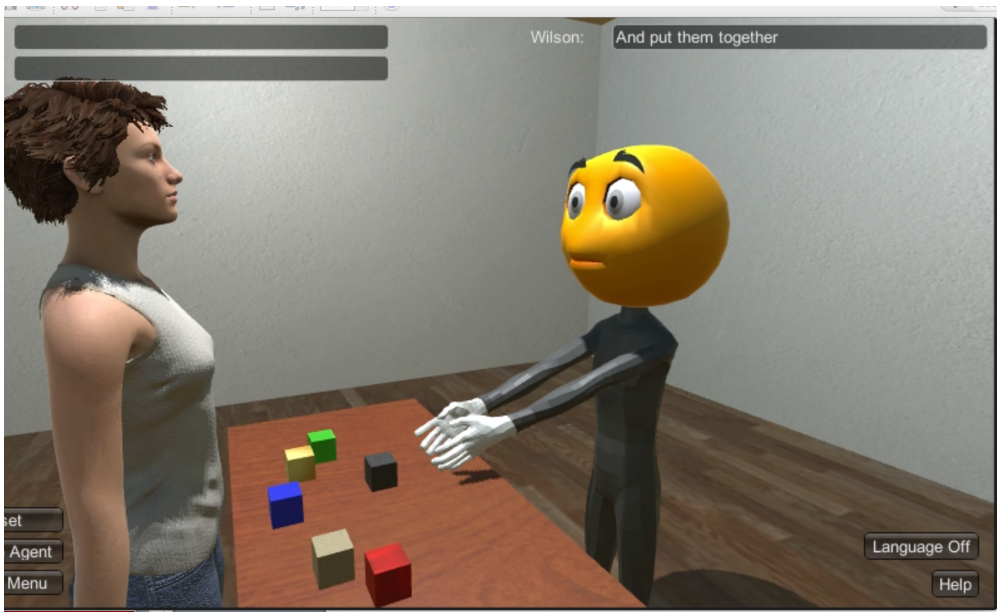
1. head nod (one form of agreement)
2. right arm down (signaling where to place something)
3. arms together (gesturing to slide blocks together)
4. two arms moving front, hands down & gripping (gesturing to grab two blocks)

Gesture as Interpreted Programs

Deixis: Wilson points to a block

Wilson Points to another block

Wilson Gestures to move them together



Wilson Claws and Moves hand (pick up)

Wilson Claws and Moves hand (pick up)

Object Model

Closing a bottle

Foo

Movie

Action satisfying Constraints

Foo

Models of Synthetic Vision

- Robotics and Computational Vision
- Game Design and Virtual Environments
- Modal Logic of Knowledge and Belief

Synthetic Vision in Robots

- Renault, Noser (1995) and others have used false color renderings from an agent's perspective to generate lists of visible obstacles and agents.
- Tu and Terzopoulos (1994), Shao (2005) used ray casting on a raster grid to detect obstacles. Agents were sensed using standard radial awareness searches.
- Niederberger (2003) implements vision as a sensor in an agent's sense-think-act cycle.

Synthetic Vision in Robots

- Octrees: used to partition a three dimensional space by recursively subdividing it into eight octants. Matrix based (MX) octrees: the subdivision point is implicitly the center of the space the node represents
- Scene Graphs
- Binary Vision Partitioning (BSP) Trees

Synthetic Vision for Virtual Humans

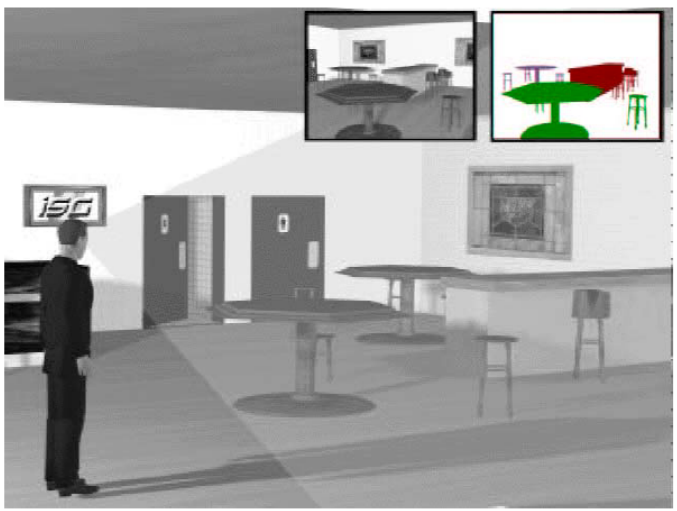


Fig. 2. Examples of synthetics vision. In the figure on the left, the picture at the bottom right corner shows the character's vision. In the figure on the right, the smaller picture on the left shows the scene rendered from the viewpoint of the character. The smaller picture on the right shows the same scene using the false colouring technique to check proximity between objects.

Synthetic Vision for Virtual Humans

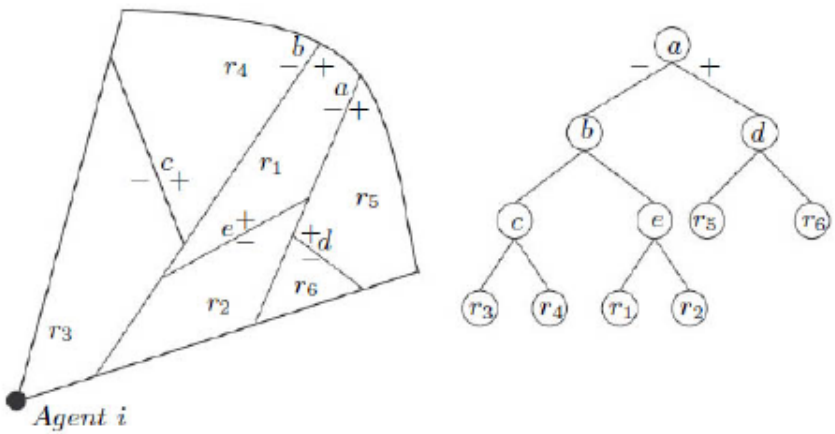


Fig. 3. The figure shows a setting where the verification of visibility via BSP and the construction of a space of subsets via the relation of proximity R_i^p are performed

Epistemic Logic

φ is a formula of Epistemic Logic ($\mathcal{L}$) if it is of the form

$$\varphi \;:=\; p \mid \neg\varphi \mid \varphi \wedge \psi \mid K\varphi$$

$K(p \to q)$: "Ann knows that p implies q"

$Kp \vee \neg Kp$: "either Ann does or does not know p"

$Kp \vee K\neg p$: "Ann knows whether p is true"

$L\varphi$: "φ is an epistemic possibility"

$KL\varphi$: "Ann knows that she thinks φ is possible"

Perception Logic (Lineworld)

Schwarzentruber (2010)

$$\varphi ::= a \rhd b \mid \bot \mid \neg\varphi \mid (\varphi \vee \varphi) \mid K_a\varphi$$

where $a, b \in AGT$. The formula $a \rhd b$ is read "agent a sees agent b" and is called a perception literal. The formula $K_a\varphi$ is read "agent a knows that φ is true". As usual $\top =^{\text{def}} \neg\bot$, $(\varphi \wedge \psi) =^{\text{def}} \neg(\neg\varphi \vee \neg\psi)$, $\hat{K}_a\varphi =^{\text{def}} \neg K_a\neg\varphi$, $(\varphi \to \psi) =^{\text{def}} (\neg\varphi \vee \psi)$ and $(\varphi \leftrightarrow \psi) =^{\text{def}} ((\varphi \to \psi) \wedge (\psi \to \varphi))$. We follow the standard rules for omission of parentheses.

Definition 15 (truth conditions)
We define $w \models \varphi$ by induction on φ:

- $w \models a \rhd b$ iff either ($\vec{dir}(a) = \texttt{Left}$ and $b < a$) or ($\vec{dir}(a) = \texttt{Right}$ and $a < b$);
- $w \not\models \bot$;
- $w \models \neg\varphi$ iff $w \not\models \varphi$;
- $w \models \varphi \vee \psi$ iff $w \models \varphi$ or $w \models \psi$.

The semantics of $w \models a \rhd b$ is intuitive: agent a sees agent b iff either b is on the left of a and a's direction is left or b is on the right of a and a's direction is right.

Lineworld Logic

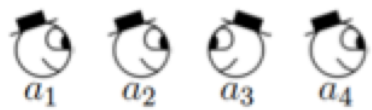


Figure 4.1: Example of a lineworld

Example 9 *Consider the lineworld w depicted in Figure 4.1. We have $w \models K_{a_1} a_1 \rhd a_3$, $w \models \neg K_{a_2} a_1 \rhd a_3$ and $w \models K_{a_2} a_3 \rhd a_1$.*

A Logic for Synthetic Vision

- Previous logics are extramission theories of vision
- Rays of light emitted from the eyes to the object: R(agent,object)
- Modal introduces accessibility relation between agent and visible objects:
  - K_i^s is a modal operator meaning "knows by seeing". $K_i^s \phi$ means: *Agent i knows ϕ by seeing it.*
- We need a Logic of Sensing
- Formally acknowledge the environment as active part of the model
- Treat the perceiver as a sensor, not an agent.
- Perception is not knowledge

Synthetic Vision as Sensing

- There is a medium, $\mathcal{M}$, which initiates or terminates the accessibility relations.
- Objects in the environment *transmit* to other objects: $\mathcal{R}_T(x, y)$.
- Agents $(a_1, a_2, \ldots, a_n)$ have sensors, s_i, that *receive* an object's transmission (optic flow) under appropriate conditions:
- For an agent, a_1, with sensor s_1, if $R_T(x, s_1)$ then we say $S_{a_1}x$.

Conclusion

- Conceptual and frame-based approaches to events are rich sources for modeling
- Mental models and schemas can be formally modeled
- Multimodal simulations builds on GL and dynamic semantics.
- VoxML is intended to be a rich modeling language for creating simulations
- VoxSim provides for such a simulation environment
- Providing a rich framework for developing and experimenting with semantic theories

Conclusion

- Conceptual and frame-based approaches to events are rich sources for modeling
- Mental models and schemas can be formally modeled
- Multimodal simulations builds on GL and dynamic semantics.
- VoxML is intended to be a rich modeling language for creating simulations
- VoxSim provides for such a simulation environment
- Providing a rich framework for developing and experimenting with semantic theories

THANK YOU!