Multimodal Semantic Simulations of Linguistically Underspecified Motion Events

Nikhil Krishnaswamy and James Pustejovsky

Department of Computer Science, Brandeis University
415 South Street, Waltham, MA 02453, USA
{nkrishna,jamesp}@brandeis.edu
http://www.voxicon.net/

Abstract. *This paper details the technical functionality of VoxSim, a system for generating three-dimensional visual simulations of natural language motion expressions. We use a rich formal model of events and their participants to generate simulations that satisfy the minimal constraints entailed by an utterance and its minimal model, relying on real-world semantic knowledge of physical objects and motion events. This paper outlines technical considerations of such a system, and discusses the implementation of the aforementioned semantic models as well as VoxSim's suitability as a platform for examining linguistic and spatial reasoning questions.*

Keywords: spatial cognition; spatial reasoning; spatial language; event semantics; simulation semantics; spatial information representation; spatial information processing; underspecification

1 Introduction

Spatial expressions in natural language rely on a wealth of world knowledge and contextual information about the properties of objects and events discussed in order to arrive at a complete interpretation of the utterance in question, making them difficult to translate into visuals. Linguistic predicates encode a certain level of knowledge that affords using them for spatial reasoning, but the level of spatial information varies for each predicate, such that many expressions leave certain parameters underspecified.

Existing work in visualization from natural language has largely focused on object placement in static scenes [7,9,43]. We have recently introduced a focus on motion verbs, using a rich formal model of events, and relying on philosophical and cognitive science approaches to linguistic interpretation, to integrate dynamic semantics into our system of event visualization, in simulations of the associated actions [37,38].

In philosophy, "mental simulation" theory attempts to model everyday human psychological competence [29], providing a *process* driven theory of mind [22]. In cognitive linguistics, "simulation" has come to mean a mental instantiation of a linguistic utterance, playing a functional role in language understanding [4,14], based on the notion of an agent's *embodiment*, as also discussed by

Narayanan and others [33]. This provides a shared semiotic structure exploitable by both a computer (as a minimal model [6,19]) and a human (via sensory interpretation). Finally, both Qualitiative Spatial Reasoning (QSR) and gaming-style AI approaches have been used to develop simulators for training driven by interactive narratives [10,16], which makes procedural simulation from language expressions a natural candidate for fast-prototyping of new scenarios.

In a dynamic semantics approach, verbs are treated as programs or processes [27] and so although the computational linguistics and cognitive linguistics communities do not often reference each other, in our opinion there is fertile ground for cross-pollination, starting with the approach of Pustejovsky and Moszkowicz [40], and for implementing language-based reasoning in a QSR framework, leveraging temporal and spatial calculi such as the Allen Temporal Relations [2] and the Region Connection Calculus (RCC) [17,18,42].

We previously presented a method for visualizing natural language expressions in a 3D environment built on the Unity game engine [37]. The goal of that work was to evaluate, through visualization, the semantic presuppositions inherent in differing lexical choices. Thus, we assert that the amount and nature of spatial information encoded in a predicate can be revealed through simulation, of which visualization is just one modality of expression. We developed VoxML [38], a modeling language which encodes object and event semantic information into *voxemes* or "visual object concepts," structured and stored in a *voxicon* (the "lexicon" of voxemes). This approach enables procedural simulation generation from semantic knowledge of an event and its participants. Using a notion of action verbs as programs [34,40], our system, given an utterance and scene containing all referenced nominals as 3D objects, enacts the verbal program over them.

This method of abstracting and composing objects and programs allows us to take a semantically complex natural language predicate, such as "lean" or "switch," enacted over arbitrary objects within the system's vocabulary, and immediately generate a visualization if such a verbal program can be executed over the mentioned objects.

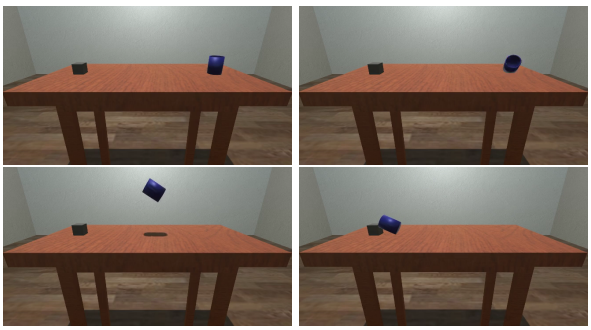


Fig. 1. Visualization of "lean the cup on the block"

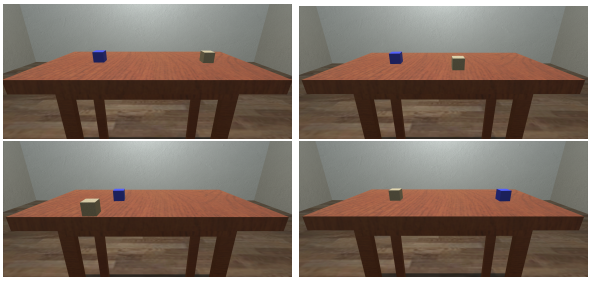


Fig. 2. Visualization of "switch the blocks"

The remainder of this article describes the technical functionality of this system, **VoxSim**, and its utility as a platform for experimentation in linguistic and context-based reasoning.

2 Architecture

VoxSim uses the Unity game engine [23] for graphics and I/O processing.[1] Input is a simple natural language sentence, which is part-of-speech tagged, dependency-parsed, and transformed into a simple predicate-logic format. These Natural Language Processing (NLP) tasks are currently handled by external applications networked to the simulator: we have interfaces for several parsers and resources, including the ClearNLP parser [8], SyntaxNet [3], and the TRIPS parser [15]. 3D assets and VoxML-modeled nominal objects and events (created with other Unity-based tools) are loaded externally, either locally or from a web server. Commands to the simulator may be input directly to the software UI, sent over a generic network connection, or selected within the **VoxSim Commander** application, a companion app for iOS. A diagram of the VoxSim architecture is shown in Fig. 3, and the front-end UI is shown in Fig. 4.

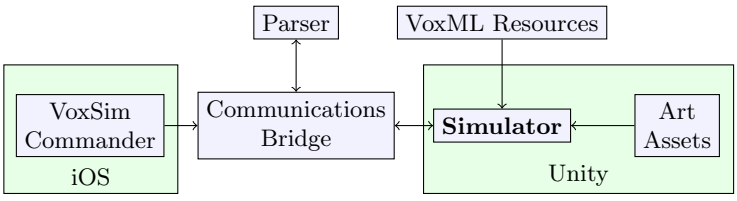


Fig. 3. VoxSim architecture schematic

[1] The VoxSim Unity project and source may be found at https://github.com/VoxML/VoxSim/. The latest stable builds are posted at http://www.voxicon.net/.

Fig. 4. VoxSim UI with sample scene

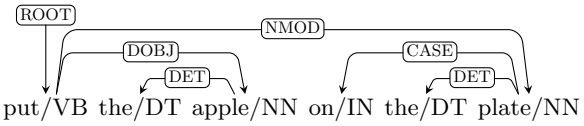
Objects from the library may be part of pre-built scenes made in the Unity Editor or may be placed in the scene by users at runtime using the "Add Object" menu. Users may enter input in the upper left or over network connection via VoxSim Commander. Output from VoxSim is printed in the upper right.

Given a tagged and dependency-parsed input sentence as shown in Fig. 5, we can transform it into predicate-logic format using the root of the parse as the VoxML PROGRAM, which accepts as many arguments as are specified in its type structure, and subsequently enqueuing any arguments that are either constants (i.e., instances of VoxML OBJECTs) or evaluate to constants at runtime (all other VoxML entity types, applied over OBJECTs). Other non-constant VoxML entity types are treated similarly to PROGRAMs, though they usually accept only one argument. Thus, the dependency arc CASE($plate, on$), for example, becomes $on(plate)$. The resulting predicate-logic formula is evaluated from the innermost first-order predicates outward until a single first-order representation is reached.

2.1 VoxML Overview

VoxML (Visual Object Concept Markup Language), is a modeling language for constructing 3D visualizations of natural language expressions [38]. It forms the scaffold used to link lexemes to their visual instantiations, or *voxemes*. There may be many-to-many correspondences between lexemes and their associated voxemes. For example, the lexeme *plate* may be visualized as a square plate or a round plate, which would be different voxemes, and a single visual object may be referred to by multiple lexemes.

Each voxeme is linked to an object geometry (nouns are OBJECTs in VoxML), a dynamic logic program (verbs are VoxML PROGRAMs), an attribute set (VoxML ATTRIBUTEs), or a transformation algorithm (VoxML RELATIONs or FUNCTIONs).



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*)))	

Fig. 5. Dependency parse for *Put the apple on the plate* and transformation to predicate-logic form.

VoxML is used to specify the "episemantic" information beyond that which can be directly inferred from the linked geometry, Dynamic Intervsal Temporal Logic (DITL) program [40], or attribute properties.

An OBJECT voxeme's semantic structure provides *habitats*, or situational contexts or environments which condition the object's *affordances*, or attached behaviors describing what can be done to the object [36]. Habitats established by previous actions may activate or deactivate certain affordances, as when flipping a cup over prevents objects from going inside it.

3 Linguistic and Semantic Analysis

From tagged and parsed input text, all noun phrases are indexed to objects in the scene. A reference to *a ball* causes the simulator to attempt to locate a voxeme instance in the scene whose lexical predicate is "ball," while an occurrence of *a block* prompts an attempt to locate a voxeme with the lexical predicate "block".

Attributive adjectives impose a sortal scale on their heads [38], so *small block* and *big block* single out two separate blocks if they exist in the scene, and the VoxML-encoded semantics of "small" and "big" discriminates the blocks based on their relative size. *red block* vs. *green block* results in a distinction based on color, a nominal attribute, while *big red block* and *small red block* introduce scalar attribution, and can be used to disambiguate two distinct red blocks by iteratively evaluating each interior term of a formula such as *big(red(block))* until the reference can be resolved into a single object instance in the scene that has all the signaled attributes[2]. The system may ask for clarification (e.g., *"Which block?"*) if the object reference is still ambiguous.

[2] See [39] for details on discriminating and referencing objects through sortal and scalar descriptions.

$$
\begin{bmatrix}
\textbf{ball} & \\
\text{LEX} = & \begin{bmatrix} \text{PRED} = \textbf{ball} \\ \text{TYPE} = \textbf{physobj, artifact} \end{bmatrix} \\
\text{TYPE} = & \begin{bmatrix} \text{HEAD} = \textbf{ellipsoid[1]} \\ \text{COMPONENTS} = \textbf{nil} \\ \text{CONCAVITY} = \textbf{convex} \\ \text{ROTATSYM} = \{X, Y, Z\} \\ \text{REFLECTSYM} = \{XY, XZ, YZ\} \end{bmatrix} \\
\text{HABITAT} = & \begin{bmatrix} \text{INTR} = \ldots \\ \text{EXTR} = \ldots \end{bmatrix} \\
\text{AFFORD\_STR} = & \begin{bmatrix} \text{A}_1 = H \rightarrow [grasp(x, [1])]hold(x, [1]) \\ \text{A}_2 = H \rightarrow [roll(x, [1])] \end{bmatrix} \\
\text{EMBODIMENT} = & \begin{bmatrix} \text{SCALE} = <\textbf{agent} \\ \text{MOVABLE} = \textbf{true} \end{bmatrix}
\end{bmatrix}
$$

Fig. 6. *Ball* voxeme markup. VoxSim references voxemes by their lexical predicates.

We use a basic set of primitive programs to represent verbs, from which we build more complex programs. There have been many previous attempts to group verbs into distinct clusters, based on syntactic behavior [26], or to associate verbs with specific spatial semantic primitives [32]. Frameworks from the computer vision and AI communities include work on representing traffic configurations and behavior [20] and in human-robot communication [13,44]. VoxML follows a similar model-theoretic approach, using an underlying semantics of a hybrid dynamic logic, Dynamic Interval Temporal Logic (DITL) [40]. Program content is then operationalized with the intent of decomposing the verbal and relational semantics while leaving object category labels intact, similar to approaches that seek to overcome descriptive constraints limited by robotic perception [31,41].

In a 3D environment, any complex motion can be decomposed into series and compositions of translations and rotations, making those obvious verbal primitives in our visualization system. Other primitives include commonly-repeating subevents of other motions, such as "grasp," a fine-grained motion of the fingers that is difficult to decompose but appears as a subevent of nearly any human-object interaction. We can then assemble complex events out of primitive motions, as in "put" in Fig. 8, and then macro-complex events, such as "stack" as a sequence of "put" events.

3.1 Habitats and Affordances

We assume that every voxeme exists within an intrinsic "habitat" [36,30], an encoding of the environment in which the object must exist simply to avoid violating any physical constraints, such as gravity, and conditions under which an object typically exists in the world.

An object's habitat introduces two parameters: it specifies how it is situated within a minimal embedding space (local embodiment); and by so doing this, it contextualizes the object, by making reference to the object's "affordances," i.e., the correlations between an agent who acts on an object and systematic or

prototypical effects. For instance, a pencil at rest must be laying flat and not resting on its tip. While a table can be situated in almost any orientation, it has an intrinsic "top," its surface, as can be inferred from common utterances such as "the table is upside down". In a typical configuration, an object's inherent top will be aligned to the world's upward vector, so we denote this in VoxML with TOP = $top(+Y)$. Being able to identify this as a habitat allows VoxSim to then reason about what behaviors and resultant states are afforded by that object in that configuration.

A voxeme's semantic structure provides both "Gibsonian" and "telic" *affordances* [21,35,36], or attached behaviors, which the object either facilitates by its geometry (Gibsonian) [21], or purposes for which it is intended to be used (telic) [35]. For example, a Gibsonian affordance for a cup is "grasp," while a telic affordance is "drink from." Following from the convention that agents of a VoxML PROGRAM must be explicitly singled out in the associated implementation by belonging to certain entity classes (e.g., humans), affordances describe what *can be done to* the object, and not what actions it *itself* can perform. Thus an affordance is notated as HABITAT $\rightarrow$ [EVENT]RESULT, and an instance such as $H_{[2]} \rightarrow [put(x, on(\mathbf{[1]}))]support(\mathbf{[1]}, x)$ can be paraphrased as "In habitat-2, an object x can be put on component-1, which results in component-1 supporting x." This procedural reasoning from habitats and affordances, executed in real time, allows VoxSim to infer the complete set of spatial relations between objects at each state and track changes in the shared context between human and computer. Thus, simulation becomes a way of tracing the consequences of linguistic spatial cues through a narrative.

3.2 Predicate-Argument Interaction

VoxML treats objects (NPs) and events (predicates) in terms of a dynamic event semantics, DITL [40]. The verbal semantics is based on a type system that encodes how a given formula ϕ and proposition π are executed/tested during the execution of a verbal program; programs can be a `state`, `process`, `transition`, `assignment`, or `test`, all of which are translated into DITL and operationalized differently.

This dynamic interpretation of events allows VoxSim's VoxML implementation to map linguistic expressions directly into simulations through an operational semantics that interprets predicates relative to their arguments' semantic encoding. Argument-sensitive distinctions in the operationalization of predicates themselves exploit the HEAD of the argument's type structure—a selected subset of geometric faces wholly or partially coterminous with the entire object geometry. This can be illustrated through the respective encodings for *on* and *in*. As shown in Fig. 7, both are configurational relations, but *on* requires the objects to be EC while *in* requires the figure object to be wholly or partially contained by the ground object. These relate directly to the affordances of an item. For instance, putting some object *on* a cup results in a support relation between the two, while putting an object *in* the cup results in a containment relation.

Additionally, the two relations specify different constraints. The spatial relation *on* requires the point of contact between the two objects to be located along the axis defined by a formula that will be found in the intrinsic habitat of the ground object and denoted with the notation *align*, subject to the full set of constraints that define that habitat. One such example would be the habitat constraints $\{align(Y, \mathcal{E}_Y), top(+Y)\}$ for a cup, where placing an object x $on(cup)$ requires contact between x and *cup* to be along the positive Y-axis (either of the cup or the world). Meanwhile, to put some object x on an object with an inherent orientation defined relative to a different surface, such as a wall with a facing surface or a TV with a screen, VoxSim exploits that object's habitat constraints $\{align(Y, \mathcal{E}_Y), front(+Z)\}$, and aligns x's Y-axis with the destination object's Y-axis, and x's front (if any defined), with the destination's front. However, the point of contact between the two objects in this situation is not concretely defined. The VoxML encoding leaves both faces along the +Y axis and faces along the ground object's +Z axis as viable candidates for contact points. That is, assuming regions respectively satisfying these mutually exclusive constraints are both unobstructed, an item such as a picture could be "on" a television or a wall in either configuration. Such an open question becomes a prime area for using simulation as an experimentation platform. This is discussed in section 4.

Meanwhile, the spatial relation *in* enforces a constraint that requires a test against the current situational context before a value assignment can be made. An example is given in Fig. 9 of a knife in a cup. If the "placed object" is too large to fit inside the object it is to be placed in, VoxSim conducts a series of calculations to see if the object, when reoriented along any of its three orthogonal axes, will be situated in a configuration that allows it to fit inside the region bounded by the ground object's containing area. The containing area is situated relative to one of the ground object's orthogonal axes, and which axis and orientation this is is encoded in the ground object's VoxML type semantics. For example, a typical cup has some rotational symmetry around its Y-axis, as well as reflectional symmetry across its XY- and YZ-planes. These parameters compose with the cup's specification as a concave object to encode a concavity opening along the Y-axis, and the additional $top(+Y)$ habitat constraint further situates this opening along the object's *positive* Y-axis. Thus, as seen in Fig. 9, the knife must be reoriented so that its world-space bounding box aligning with the cup's Y-axis is smaller than the bounds of the cup's opening in that same configuration.

A typical wall with uniform surface topology affords no containment in either a Gibsonian or telic sense: however, something like a picture hanging on the wall may be interpreted as being contained by the bounds of the wall's surface, and this involves coercing the wall to its surface as a 2D region in order to satisfy one of the eight basic 2D relations in the Region Connection Calculus, whereas the wall voxeme taken as a whole is a 3D object. On the other hand, something interpenetrating the wall (perhaps by application of enough force), as in the left image of Fig. 10, would satisfy the *PO* configuration required by *in*.

In Fig. 8, "put" is given arguments `agent`, `obj1`, and `rel(obj2)`, where `rel` refers to the class of spatial functions including `on` and `in`. The typing of both

$$\begin{bmatrix} \textbf{on} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{on} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{CLASS} = \textbf{config} \\ \text{VALUE} = \textbf{EC} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:3D} \\ \text{A}_2 = \textbf{y:3D} \end{bmatrix} \\ \text{CONSTR} = \textbf{y}\rightarrow\text{HABITAT}\rightarrow \\ \quad \text{INTR}[align] \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textbf{in} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{in} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{CLASS} = \textbf{config} \\ \text{VALUE} = \textbf{PO} \parallel \textbf{TPP} \parallel \\ \quad \textbf{NTPP} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{x:3D} \\ \text{A}_2 = \textbf{y:3D} \end{bmatrix} \\ \text{CONSTR} = \textbf{y}\rightarrow\text{HABITAT}\rightarrow \\ \quad \text{INTR}[align]\textbf{?} \end{bmatrix} \end{bmatrix}$$

Fig. 7. VoxML structures for "on" and "in," showing distinction in configurational value and constraints. The question mark in the typing of "in" denotes the test mentioned previously.

$$\begin{bmatrix} \textbf{put} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{put} \\ \text{TYPE} = \textbf{transition\_event} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{transition} \\ \text{ARGS} = \begin{bmatrix} \text{A}_1 = \textbf{agent} \\ \text{A}_2 = \textbf{obj1} \\ \text{A}_3 = \textbf{rel(obj2)} \end{bmatrix} \\ \text{BODY} = \begin{bmatrix} \text{E}_1 = grasp(A_1, A_2) \\ \text{E}_2 = [while(hold(A_1, A_2), move(A_2))] \\ \text{E}_3 = [at(A_2, A_3) \rightarrow ungrasp(A_1, A_2)] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 8. VoxML structure for "put"

"put" and "on" encode the calculation of such parameters as object trajectory and destination location (e.g., the position denoted by `on(block)` vs. `on(plate)` or `on(wall)`). As seen in BODY, object A_2 is moved to location A_3, calculated by operationalizing the specified relation over `obj2`. The results, depending on the arguments, may be configurations such as those shown in Figs. 9-11.

Fig. 9. "In the cup" vs. "on the cup"

In Figs. 9-11 we can see the visualizations of the composition of these typing distinctions. The expression `on(wall)` selects for a vertical face of the object while in all other examples, `on` selects for the object's top. The location of

Fig. 10. "In the wall" vs. "on the wall"

Fig. 11. "On the table" vs. "on the wall" vs. "on the plate"

"top" is computed based on the object's VoxML type structure. The top of a plate, a slightly concave object when situated in its default habitat, is located slightly lower on the Y-axis than the plate's highest overall point. The expression `in(cup)` selects the interior surface of the cup, an object with a telic affordance of containment, while `in(wall)` explicitly interpenetrates the wall, as "wall" lacks the containment telic role. The system attempts to maximally satisfy the constraints placed upon it by the NL expression. For instance, placing the knife object `on(cup)` lays the knife across the rim of the upright cup. In the same (horizontal) orientation, the knife cannot be placed at the location computed by `in(cup)`, so the system must transform the knife so that the RCC representation of `in(cup)`, $PO(knife, cup)$, can be satisfied without violating any of the physical or structural constraints of the cup or knife objects.

As events proceed, VoxSim maintains the current set of relations that exist between every pair of objects in the scene. The currently-implemented reasoning approach is built on top of RCC, particularly variants for relations in 3D space [1], but it can easily be extended to other QSR approaches, including the situation calculus [5], and the Intersection Calculus [25,28]. Where the spatial reasoning calculus may leave certain parameters underspecified, the object, relation, and event VoxML encodings allow additional parameters to inform VoxSim's reasoner in attempting to visualize a situation in accordance with the mental instantiation of the utterance provided, such as determining the angle of entry for an object into a cup, as in the example discussed in Section 3.2.

In some cases, even the combined force of spatial reasoning calculi and VoxML will not be enough to unambiguously supply all information missing from the

linguistic utterance to create an adequate simulation: this is, of course, fertile ground for experimentation.

4 Underspecification

When constructing minimal models, it is common to leave certain parameters underspecified [19]. That is, it is permissible for a model to contain the proposition "the ball rolled" without providing information such as direction, speed, size of the ball, friction between the ball and the supporting surface, etc.; this information *can* be specified, but the model is still considered complete without it. On the other hand, when a ball rolls in the real world, these model-unspecified parameters all have values assigned to them, even if those values are not specifically measured. Likewise, a *simulation* of the same event requires that certain of these parameters be specified in order for the verbal program enacted over the arguments actually to be executed from frame to frame. Certain parameters, such as the precise location indicated by a relation over an object, can be calculated from the composition of the program and objects as discussed above in Section 3, while others are either not encoded in the semantics at all or the semantic composition leaves the value imprecise or ambiguous.

4.1 Current Results of Simulation Output

From the composition of VoxML and dynamic event semantics, we can achieve a model of an event that is "filled out" with information to an extent far greater than that provided by a minimal model. The kind of *forward composition* illustrated thus far can take a minimal model of an event and augment it with general-domain lexical world knowledge about the event predicate and its participants, which allows VoxSim to create a more extensive informational context than a minimal model provides, which it can then share with its human user. Visualization, an intuitively accessible modality for most humans, becomes a medium through which to share that context. From the minimal model, VoxSim's interpretation of encoded VoxML and dynamic semantic knowledge allows it to define the interpretation of the event predicate as a logic program with further specification than the logic program does in isolation, by drawing on and composing knowledge about the arguments, relations, and preconditions involved in the event. For example, in the simple utterance "roll the ball," we as language interpreters know that there must be a surface to roll the ball over, even though no such surface is mentioned in the utterance given. The VoxML type encoding for *roll* (Fig. 12) makes this explicit, representing the surface as A_3.

$$\left[\begin{array}{l} \textbf{roll} \\ \text{LEX} = \ldots \\ \text{TYPE} = \left[\begin{array}{ll} \text{HEAD} = & \textbf{process} \\ \text{ARGS} = & \left[\begin{array}{ll} \text{A}_1 = & \textbf{x:agent} \\ \text{A}_2 = & \textbf{y:physobj} \\ \text{A}_3 = & \textbf{z:physobj} \end{array}\right] \\ \text{BODY} = & \ldots \end{array}\right] \end{array}\right]$$

Fig. 12. Abbreviated VoxML type structure for "roll."

As mentioned above, in a minimal model, the nature of the surface (shape, consistency, etc.) can be left out completely. With forward composition and VoxML, VoxSim fills in some of these parameters by composing the event with the surface object. Object habitats and configurations therefore become *conditioning environments* that enforce constraints on the minimal model.

While the integration of context, spatially conditioning environments, and real-world knowledge may provide sufficient information to determine the *nature* of a spatial constraint or set of constraints, the precise *instantiated value*, down to the 3D coordinates and rotation of an object at each step of program execution, may still be left imprecise. For example, through forward composition, VoxSim may be able to determine the region within which an object must be placed to satisfy the completion of an event it is commanded to simulate, but the precise location within that region where the object should be by the end of the event is both left underspecified *and* there is no method from the lexical semantics to determine exactly what value that location should take. Nonetheless, in order for any platform such as VoxSim to execute the fully specified program at runtime, *all* required parameters must have a value assigned, including those that are potentially never mentioned in the linguistic utterance linked to the event and never raised in the additional encoding used by forward composition.

For instance, given a cup sitting upright (in the proper orientation), "put the lid on the cup" clearly describes an end state where the lid closes the cup's opening. However, if the cup is on its side, we find that, if the end location is chosen at random, such as by a Monte Carlo method, from possible configurations left available by the currently operating set of constraints and configurations, both a lid closing the opening of the cup, and one that is touching the cup on the positive Y-axis (i.e., explicitly "on top of" the cup independent of orientation) can be computed as satisfying the command "put the lid on the cup" (Fig. 13).

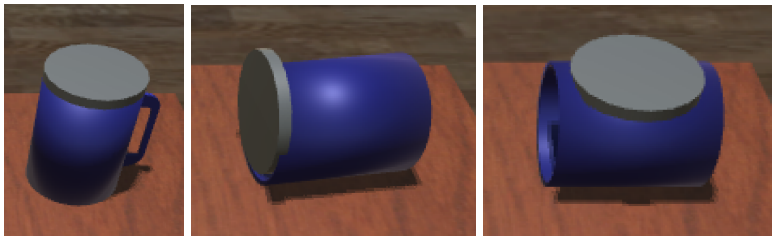


Fig. 13. Orientation-dependent visualizations of "put the lid on the cup"

Ambiguity of resultant configuration occurs even in default habitats. To say that something is "on the TV" can usually be interpreted specifically relative to the type of the object (e.g. using a type system such as Generative Lexicon [35]), where a physical object on the TV is *on top* of it while an image is on *the screen*, the TV's semantic head per VoxML.[3] However, in the case where the

[3] We ignore here the idiomatic non-spatial reading of *on TV*, denoting "the information content available through the medium 'TV'."

object on the TV is something, such as a sheet of paper, that could be placed on either surface, the ambiguity remains. When VoxSim must make a *simulation assignment* of values through random choice, either result may be generated as the end state of the event (as shown in Fig. 14), and a human is required to judge the appropriateness of either choice.

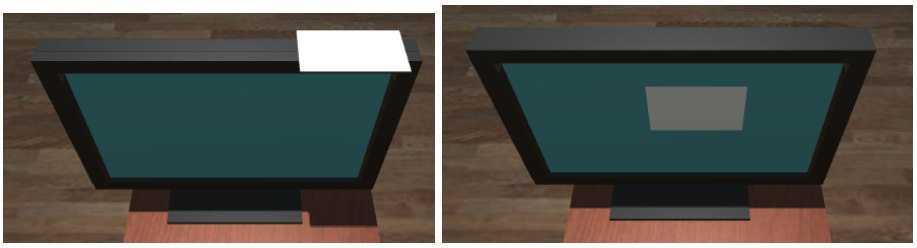


Fig. 14. Results from "put the paper on the TV"

Underspecification can thus arise in both post-conditions and in the action sequence of the event itself, such as when parameters like speed and direction of motion are left unknown. Additionally, depending on the motion predicate being simulated, the manner of motion itself may be left underspecified. "Move the cup to the center of the table" implies a path by the specification of the destination, but the manner by which the cup should be moved is not specified. Forward composition may narrow the search space—in this example, the table provides a surface to be moved over, which may make sliding or rolling (if the cup lacks a handle) preferable to other forms of motion—but ultimately the predicate itself contains unspecified parameters and so may be replaced in a simulation with another motion predicate that satisfies the same basic constraints and imposes others on top of them. Some examples are given below in Table 1, with *move* as the least specified motion predicate.

Program	**Underspecified parameters**
put(y,z)	speed of motion
slide(y)	speed, direction of translocation
roll(y)	speed, direction of rotation
turn(y)	speed, direction of rotation speed, direction of translocation
move(y)	manner, speed, direction of motion

Table 1. Sample motion predicates and underspecified parameters

4.2 Experimentation and Early Results

Where a program's DITL formula says nothing about the nature of a parameter (e.g., it states that b_{n+1} is farther from b_0 than b_n is, but it does not state in which direction), we can use VoxSim to generate a set of visualizations of that event with randomly assigned values for the underspecified parameter, and have

human judges evaluate the results of this Monte Carlo simulation generation to determine what, if any, "prototypical" values exist for that parameter.

In order to better determine the nature of these values, we designed a series of experiments in which we asked human judges on the Amazon Mechanical Turk platform to make similarity judgements between a set of input sentences and simulations generated from them with randomly chosen values for parameters requiring value assignment.

1. From one input sentence, we generate three visualizations of the event and ask the judge to determine which visualization(s) best depict the event described by the input sentence. Multiple choices are allowed, and the judge may respond that none of the provided visualizations adequately depicts the event described.
2. From a single visualization of an event, we ask the judge to determine which of three sentences—one of which is the original input and two of which use different predicates in place of the original one—best describes the event depicted. Multiple choices are allowed, and the judge may respond that none of the provided sentences adequately describes the event depicted.

Programs		**Objects**	
move x	put x on y	block	grape
turn x	put x in y	ball	banana
roll x	lean x on y	plate	bowl
slide x	lean x against y	cup	knife
spin x	flip x on edge	disc	pencil
lift x	flip x at center	book	paper sheet
stack x	close x	blackboard	
put x near y	open x	bottle	
put x touching y		apple	

Table 2. Test set of verbal programs and objects

We captured a total of 3357 videos of individual events (3 each for 1119 input sentences generated from combining the objects and events in Table 2). For each event generated, the underspecified parameters were logged to a SQL database to facilitate retrieval and evaluation. Videos were uploaded to Amazon Mechanical Turk to be evaluated by 8 workers for each individual task according to the guidelines listed above. We received a total of 8952 individual evaluations on the first task listed (8 evaluations per each of the 1119 input sentences), and a total of 26,856 individual evaluations on the second task (8 evaluations per each of 3357 videos). A full analysis of each predicate is currently in progress as of this writing, but we have completed evaluation on the input classes "put x near y" and "put x touching y."

"Touching" was randomly specified as one of the qualitative spatial relations "left," "right," "in front," "behind," or "on" with the addition of an RCC external connection (EC) constraint. Thus the acceptability judgment of the visualized event may be conditioned on the relation between the two objects at the

end of the event, or on the motion of the moving object between configurations relative to the stationary object.

QSR	**P(accept|QSR)**
behind(y)	0.5474
in_front(y)	0.5816
left(y)	0.4995
right(y)	0.5560
on(y)	0.6683

Movement (M)	**P(accept|M)**	**Movement (M)**	**P(accept|M)**
behind→behind(y)	0.5347	*left→behind(y)*	0.5732
behind→in_front(y)	0.4758	*left→in_front(y)*	0.5853
behind→left(y)	0.5014	*left→left(y)*	0.5266
behind→right(y)	0.4888	*left→right(y)*	0.5211
behind→on(y)	0.7453	*left→on(y)*	0.6492
in_front→behind(y)	0.4523	*right→behind(y)*	0.5406
in_front→in_front(y)	0.6447	*right→in_front(y)*	0.5786
in_front→left(y)	0.4601	*right→left(y)*	0.4777
in_front→right(y)	0.5756	*right→right(y)*	0.5847
in_front→on(y)	0.6234	*right→on(y)*	0.7081

Table 3. Results for visualizations of "put x touching y"

We observe a lower likelihood for visualizations to be judged acceptable when the moving object moves from behind the still object to in front of it, and vice versa. Visualizations where the moving object ends to the left of the still object are also less likely to be judged acceptable. This is a weaker correlation than the dispreference for behind-to-front/front-to-behind motion, but is still noticeable, falling about 1.16σ below the mean likelihood of acceptance. We also see a strong preference for the "on" specification of "touching." Like the apparent inclination against the left side, this seems to be independent of the moving object's starting location relative to the still object. These preferences may be factors of point of view or frame of reference, conjectures which will be the subject of further experimentation and evaluation.

Unlike "touching," specification for "near" must fall in a continuous range as it is difficult to define discrete relations that can further specify "near." The distribution of distances between the two objects at the end of the event was partitioned using quintiles (the interval from 0 to the first quintile being the least distance between the objects), which conditioned the acceptability judgement. QSR relations were also taken into account, but without the EC constraint implied by "touching."

Evaluators unsurprisingly preferred visualizations where the two objects ended up nearer relative to each other. In the first three distance intervals, we observe a slight preference for events where the moving object ends up behind the still object. This may be an effect of foreshortening caused by the point of view, as with some of the "touching" specifications. When conditioning on the joint distribution of the distance interval and the QSR relation, there is some apparent

Interval	**P(accept|QU)**
First	0.7523
Second	0.6207
Third	0.3890
Fourth	0.3655
Fifth	0.1295

Interval	**QSR**	**P(accept|QU,QSR)**	**Interval**	**QSR**	**P(accept|QU,QSR)**
First	*behind(y)*	0.7730	Third	*left(y)*	0.3945
First	*in_front(y)*	0.7349	Third	*right(y)*	0.3825
First	*left(y)*	0.7338	Fourth	*behind(y)*	0.1713
First	*right(y)*	0.7712	Fourth	*in_front(y)*	0.4308
Second	*behind(y)*	0.6701	Fourth	*left(y)*	0.2093
Second	*in(y)*	0.5797	Fourth	*right(y)*	0.4699
Second	*left(y)*	0.6675	Fifth	*behind(y)*	0.0972
Second	*right(y)*	0.5819	Fifth	*in_front(y)*	0.1401
Third	*behind(y)*	0.4151	Fifth	*left(y)*	0.1250
Third	*in_front(y)*	0.3644	Fifth	*right(y)*	0.1348

Table 4. Results for visualizations of "put x near y"

confusion in judgements of events in the fourth distance interval, where σ for the population of P(accept|QSR) is greater than .15, where in all other intervals σ for P(accept|QSR) falls between .019 and .051. This is possibly a factor of workers being unable to judge purely from the visuals whether an object that began its movement from a position in the fourth distance interval relative to the still object, actually ended the motion nearer than it began, whereas in preceding intervals, the resulting location was more likely to be unambiguously "near" regardless of starting location. These factors may be revealed by conditioning on starting location or distance.

Evaluation for the remaining predicates is proceeding using similar qualitative and quantitative methods.

4.3 Further Experimentation

While continuing to evaluate results of the above experiments, we are also augmenting them with with an automatic evaluation task intended to integrate VoxML and VoxSim with machine learning approaches.

- We are evaluating machine learning methods using the sparse feature vectors generated during the event capture process, training models to select the correct sentence originally used to generate the visualization in question from three candidates provided. As in the human evaluation tasks, it will be possible for the automatic evaluation to rate multiple sentences as equally correct, should the probabilities of multiple candidates come out equal in the model, or to rate none as correct, should no probability come out high enough. This evaluation is essentially a machine learning-based version of the

evaluation done by humans in Task #2 discussed above and can be compared to those same results.

To date, we have established a baseline on this automatic evaluation task using a maximum entropy logistic regression classifier using generalized iterative scaling. Over a 10-fold cross-validation of the test set, the MaxEnt classifier achieves **48.50%** accuracy on selecting the correct event predicate alone, and **45.64%** when selecting the correct sentence in its entirety.

Predicting Predicate Only			**Predicting Full Sentence**		
Total	**Correct**	**Incorrect**	**Total**	**Correct**	**Incorrect**
3357	1628	1729	3357	1532	1825
μ **Accuracy**	σ	σ^2	μ **Accuracy**	σ	σ^2
48.50%	0.29066	0.08448	45.64%	0.02424	0.00059

Table 5. Accuracy tables for baseline automatic evaluation

The baseline results when selecting the predicate alone display a much higher variance than the results when selecting the entire sentence, pointing to the existence of some "confusing" features when judging the predicate by itself, or indicating some extra information provided by object features resulting in more consistent results across folds. Nevertheless, this baseline exhibits only 12-15% improvement over random chance in a three-way classification task and we expect more sophisticated machine learning methods will easily beat this baseline. We are exploring options available through Google's Tensor Flow framework, including the newly-released `tf-seq2seq` sequence-to-sequence model.

5 Discussion and Future Work

We have presented here a method for incorporating motion and dynamic spatial semantics into a visualization framework. We have shown the underlying processing pipeline from natural language input to minimal model to simulation to rendering, incorporating encodings of real-world semantic knowledge from DITL and VoxML to augment the minimal model. Resulting simulations show how composing knowledge of objects and events allows a computer system, VoxSim, to create visualizations that accord with human understanding of motion events, and we have presented some areas where the full level of compositional knowledge provided still leaves some ambiguity that prevents a single simulation from being generated until underspecified parameters are given values.

Following on this, we have outlined in Section 4.3 a further set of experiments to determine prototypical or "best" values for some of these commonly-occurring parameters. Preliminary results are presented above, and a full evaluation is forthcoming [24].

We are also developing methods for automatically composing complex behaviors from primitives, based on DITL, as well as building a corpus of linked

simulations and event-annotated video in order to train algorithms to discriminate events based on their participants' motions [11,12].

Finally, we are planning on building links to lexical semantic resources such as VerbNet to allow us to leverage existing datasets for macro-program composition, and to expand the semantic processing to encompass event sequences, allowing us to generate single-input simulations of narratives beyond the sentence level.

VoxSim provides a method not only for generating 3D visualizations using an intuitive natural language interface instead of specialized skillsets (a primary goal of programs such as WordsEye [9]), but also a platform on which researchers may conduct experiments on the discrete observables of motion events while evaluating semantic theories, thus providing data to back up theoretical intuitions. We believe that visual simulation provides an intuitive way to trace the entailments that inhere in spatial expressions through a narrative, enabling a broader study of event and motion semantics.

Experiments such as those outlined above may be used by researchers to gather data on the semantic presuppositions humans hold regarding the prototypical realization of motion events, conditioned on variables like objects involved and point of view. Experiments of this kind can easily be segmented across population groups or across languages with the addition of appropriate vocabulary and NLP packages. We also believe that the automatic evaluation described can serve machine learning researchers in determining the salient features of motion events and object semantics from a computational and deep learning perspective, offering insight into the differences between human and artificial spatial cognition. A full description of those results will be forthcoming shortly.

Acknowledgements

We would like to thank the reviewers for their perceptive and helpful comments. This work is supported by a contract with the US Defense Advanced Research Projects Agency (DARPA), Contract W911NF-15-C-0238. Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We would like to thank Scott Friedman, David McDonald, Marc Verhagen, and Mark Burstein for their dicussion and input on this topic. All errors and mistakes are, of course, the responsibilities of the authors.

References

1. Albath, J., Leopold, J.L., Sabharwal, C.L., Maglia, A.M.: Rcc-3d: Qualitative spatial reasoning in 3d. In: CAINE. pp. 74–79 (2010)
2. Allen, J.: Towards a general theory of action and time. Arificial Intelligence 23, 123–154 (1984)
3. Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., Collins, M.: Globally normalized transition-based neural networks. arXiv preprint arXiv:1603.06042 (2016)

4. Bergen, B.K.: Louder than words: The new science of how the mind makes meaning. Basic Books (2012)
5. Bhatt, M., Loke, S.: Modelling dynamic spatial systems in the situation calculus. Spatial Cognition and Computation (2008)
6. Blackburn, P., Bos, J.: Computational semantics. THEORIA. An International Journal for Theory, History and Foundations of Science 18(1) (2008)
7. Chang, A., Monroe, W., Savva, M., Potts, C., Manning, C.D.: Text to 3d scene generation with rich lexical grounding. arXiv preprint arXiv:1505.06289 (2015)
8. Choi, J.D., McCallum, A.: Transition-based dependency parsing with selectional branching. In: ACL (1). pp. 1052–1062 (2013)
9. Coyne, B., Sproat, R.: Wordseye: an automatic text-to-scene conversion system. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 487–496. ACM (2001)
10. Dill, K.: A game ai approach to autonomous control of virtual characters. In: Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) (2011)
11. Do, T., Krishnaswamy, N., Pustejovsky, J.: Ecat: Event capture annotation tool. Proceedings of ISA-12: International Workshop on Semantic Annotation (2016)
12. Do, T., Pustejovsky, J.: Fine-grained event learning of human-object interaction with lstm-crf. In: Proceedings of European Symposium on Artificial Neural (ESANN) 2017 (2017)
13. Dzifcak, J., Scheutz, M., Baral, C., Schermerhorn, P.: What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. pp. 4163–4168. IEEE (2009)
14. Feldman, J.: From molecule to metaphor: A neural theory of language. MIT press (2006)
15. Ferguson, G., Allen, J.F., et al.: Trips: An integrated intelligent problem-solving assistant. In: AAAI/IAAI. pp. 567–572 (1998)
16. Forbus, K.D., Mahoney, J.V., Dill, K.: How qualitative spatial reasoning can improve strategy game ais. IEEE Intelligent Systems 17(4), 25–30 (2002)
17. Galton, A.: Towards an integrated logic of space, time, and motion. In: Bajcsy, R. (ed.) Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93). pp. 1550–5. Morgan Kaufmann, San Mateo, CA (1993)
18. Galton, A.: Qualitative Spatial Change. Oxford University Press, Oxford (2000)
19. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP. vol. 88, pp. 1070–1080 (1988)
20. Gerber, R., Nagel, H.H.: Representation of occurrences for road vehicle traffic. Artificial Intelligence 172(4), 351–391 (2008)
21. Gibson, J.J., Reed, E.S., Jones, R.: Reasons for realism: Selected essays of James J. Gibson. Lawrence Erlbaum Associates (1982)
22. Goldman, A.I.: Simulating minds: The philosophy, psychology, and neuroscience of mindreading. Oxford University Press (2006)
23. Goldstone, W.: Unity Game Development Essentials. Packt Publishing Ltd (2009)
24. Krishnaswamy, N.: Monte-Carlo Simulation Generation Through Operationalization of Spatial Primitives. Ph.D. thesis, Brandeis University (2017)
25. Kurata, Y., Egenhofer, M.: The 9+ intersection for topological relations between a directed line segment and a region. In: Gottfried, B. (ed.) Workshop on Behaviour and Monitoring Interpretation. pp. 62–76. Germany (September 2007)
26. Levin, B.: English verb class and alternations: a preliminary investigation. University of Chicago Press (1993)

27. Mani, I., Pustejovsky, J.: Interpreting Motion: Grounded Representations for Spatial Language. Oxford University Press (2012)
28. Mark, D., Egenhofer, M.: Topology of prototypical spatial relations between lines and regions in english and spanish. In: Proceedings of the Twelfth International Symposium on Computer- Assisted Cartography. vol. 4, pp. 245–254 (1995)
29. Markman, K.D., Klein, W.M., Suhr, J.A.: Handbook of imagination and mental simulation. Psychology (2012)
30. McDonald, D., Pustejovsky, J.: On the representation of inferences and their lexicalization. In: Advances in Cognitive Systems. vol. 3 (2014)
31. Moratz, R., Fischer, K., Tenbrink, T.: Cognitive modeling of spatial reference for human-robot interaction. International Journal on Artificial Intelligence Tools 10(04), 589–611 (2001)
32. Muller, P.: A qualitative theory of motion based on spatio-temporal primitives. In: Cohn, A.G., Schubert, L., Shapiro, S.C. (eds.) KR'98: Principles of Knowledge Representation and Reasoning, pp. 131–141. Morgan Kaufmann, San Francisco, California (1998)
33. Narayanan, S.S.: KARMA: Knowledge-based active representations for metaphor and aspect. University of California, Berkeley (1997)
34. Naumann, R.: A dynamic approach to aspect: Verbs as programs. University of Düsseldorf, submitted to Journal of Semantics. (1999)
35. Pustejovsky, J.: The Generative Lexicon. MIT Press, Cambridge, MA (1995)
36. Pustejovsky, J.: Dynamic event structure and habitat theory. In: Proceedings of the 6th International Conference on Generative Approaches to the Lexicon (GL2013). pp. 1–10. ACL (2013)
37. Pustejovsky, J., Krishnaswamy, N.: Generating simulations of motion events from verbal descriptions. Lexical and Computational Semantics (* SEM 2014) p. 99 (2014)
38. Pustejovsky, J., Krishnaswamy, N.: VoxML: A visualization modeling language. In: Chair), N.C.C., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), Paris, France (May 2016)
39. Pustejovsky, J., Krishnaswamy, N.: Envisioning language: The semantics of multimodal simulations (forthcoming)
40. Pustejovsky, J., Moszkowicz, J.: The qualitative spatial dynamics of motion. The Journal of Spatial Cognition and Computation (2011)
41. Raman, V., Lignos, C., Finucane, C., Lee, K.C., Marcus, M.P., Kress-Gazit, H.: Sorry dave, i'm afraid i can't do that: Explaining unachievable robot tasks using natural language. In: Robotics: Science and Systems. vol. 2, pp. 2–1. IEEE (2013)
42. Randell, D., Cui, Z., Cohn, A.: A spatial logic based on regions and connections. In: Kaufmann, M. (ed.) Proceedings of the 3rd Internation Conference on Knowledge Representation and Reasoning. pp. 165–176. San Mateo (1992)
43. Siskind, J.M.: Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. J. Artif. Intell. Res.(JAIR) 15, 31–90 (2001)
44. Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., Brock, D.: Spatial language for human-robot dialogs. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 34(2), 154–167 (2004)