International Standard

ISO 24617-10

First edition
2024-08

Language resource management — Semantic annotation framework (SemAF) —

Part 10: Visual information

Gestion des ressources linguistiques - Cadre d'annotation sémantique —

Partie 10: informations visuelles (VoxML)

Reference number
ISO 24617-10:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 37, *Language and terminology,* Subcommittee SC 4, *Language resource management*.

A list of all parts in the ISO 24617 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document standardizes the specification of a semantic annotation scheme for visual information, based on a modelling language for constructing three-dimensional (3D) visualizations of concepts denoted by natural language (NL) expressions. This modelling language serves as a semantic basis of interpreting the semantic forms of annotation structures model-theoretically by constraining the models for interpretation. This document focuses on the introduction of the modelling language as a semantic basis for interpretation, since the syntactic specification of the annotation scheme for visual information is a simplified formulation based on the abstract specification of the spatio-temporal annotation schemes, such as those specified in ISO 24617-1, ISO 24617-7 and ISO 24617-14. These three standards lay a theoretical basis for this document, which specifies ways of annotating visual information involving motions and actions that are spatio-temporally characterized.

The modelling language, named "VoxML" (visual object concept structure modelling language), where "Vox" abbreviates "visual object concept structure" (VOCS), can be used as the platform for creating multimodal semantic simulations in the context of human-computer communication. VoxML encodes semantic knowledge of real-world objects represented as 3D models, and of events and attributes related to and enacted over these objects. VoxML is intended to overcome the limitations of existing 3D visual markup languages by allowing for the encoding of a broad range of semantic knowledge that can be exploited by a variety of systems and platforms, leading to multimodal simulations of real-world scenarios using conceptual objects that represent their semantic values.

NOTE 1 The main content of this document is based on References [1] and [2]. Reference [1] was developed by the Brandeis University Computer Science Department in the context of communicating with computers (CwC), a Defence Advanced Research Projects Agency (DARPA) effort to identify and construct computational semantic elements, for the purpose of carrying out joint plans between a human and computer through NL discourse.

NOTE 2 This document adopts VoxML as a semantic basis for enriching the model for interpreting the descriptions of objects, actions and relations involving dynamic visual information.

This document outlines a specification:

a) to formulate the annotation scheme for visual information;

b) to represent semantic knowledge of real-world objects represented as 3D models.

It uses a combination of parameters that can be determined from the object's geometrical properties as well as lexical information from NL, with methods of correlating the two where applicable. This information allows for visualization and simulation software to fill in information missing from the NL input and allows the software to render a functional visualization of programs being run over objects in a robust and extensible way. Currently, a voxicon, which is the structured repository of visual object concepts, contains 500 object (noun) voxemes, lexemes or entries of the voxicon, and 10 program (verb) voxemes.

NOTE 3 As this library of available voxemes continues to grow, the specification elements will operationalize an increasingly large library of various and more complicated programs. A voxeme library and visualization software where users will be able to conduct visualizations of available behaviours driven by VoxML after parsing and interpretation is available from Reference [25].

Language resource management — Semantic annotation framework (SemAF) —

Part 10: Visual information

1 Scope

This document specifies an annotation language for visual information, based on VoxML (visual object concept structure modelling language), a modelling language for the visualizations of concepts and actions denoted by natural language (NL) expressions in three dimensions (3D).

The specification of the VoxML-based annotation scheme conforms to the requirements given in ISO 24617-1, ISO 24617-7 and ISO 24617-14. The adoption of VoxML, specified in ISO 24617-14 as a semantic basis, is necessary for the 3D simulation and visualization of actions and motions taken by both human and artificial agents in real-life situations.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 24610-1:2006, *Language resource management — Feature structures — Part 1: Feature structure representation*

ISO 24617-1, *Language resource management — Semantic annotation framework (SemAF) — Part 1: Time and events (SemAF-Time, ISO-TimeML)*

ISO 24617-7, *Language resource management — Semantic annotation framework — Part 7: Spatial information*

ISO 24617-14, *Language resource management — Semantic annotation framework (SemAF) — Part 14: Spatial semantics*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
affordance
affordance structure
set of specific actions, described along with the requisite conditions, that the object may take part in

3.1.1
Gibsonian affordance
GA
set of specific actions that an agent can perform with an object that is presented to the agent

EXAMPLE Hold, grasp, move.

3.1.2
telic affordance
set of goal-oriented or intentionally situated actions of an agent on an object presented to the agent

EXAMPLE An agent *eating* an apple when it is presented to the agent.

3.2
habitat
representation of an object situated within a partial minimal model

3.3
minimal embedding space
MES
three-dimensional (3D) region within which the state is configured, or the event unfolds

3.4
qualia
qualia structure
QS
relational forces or aspects of a lexical item or concept

3.5
telic
purpose or function *qualia* (3.4) of an object

3.6
voxeme
basic entries in *voxicon* (3.7)

3.7
voxicon
lexicon or list of basic visual object concepts of VoxML (visual object concept structure modelling language)

4 Abbreviated terms

3D	three dimensional
A	agentive role
ARG	argument
AS	atomic structure
AS_{visML}	annotation scheme for visual information markup language
$ASyn_{visML}$	abstract syntax for visual information markup language
$CSyn_{visML}$	concrete syntax for visual information markup language
C	constitutive property
F	formal property
GA	Gibsonian affordance

ID	identifier
MES	minimal embedding space
NL	natural language
NLP	natural language processing
QS	qualia structure
T	telic role
Vox	visual object concept structure
VoxML	visual object concept structures modelling language
XML	extensible markup language

5 Basic semantic assumptions — Habitats and affordances

Before introducing the VoxML specification, this document reviews two basic assumptions regarding the semantics underlying the model. Following the Generative Lexicon,[3] lexical entries in the object language are given a feature structure consisting of a word's basic type, its parameter listing, its event typing and its qualia structure. In accordance with ISO 24610-1:2006, each feature structure shall be typed, consisting of pairs of features (attributes) and values, either atomic or complex. If a value is a variable, then it is bound either universally, existentially, or by the lambda operator, as shown in Example 1.

The semantic structure of an object shall be analysed into the following four sub-structures:

a) atomic structure (formal): objects expressed as basic nominal types;

b) subatomic structure (constitutive): mereo-topological structure of objects;

c) event structure (telic) and (agentive): origin and functions associated with an object;

d) macro-object structure: how objects fit together in space and through coordinated activities.

Objects can be partially contextualized through their qualia structure. For example, a food item has an atelic value of "eat"; an instrument for writing has a telic value of "write"; a cup has a telic value of "hold", etc. As a further example, the lexical semantics for the noun "chair" carries a telic value of "sit_in":

EXAMPLE 1

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

where

AS	is an atomic structure;
QS	is a qualia structure;
ARG1	is argument 1;
F	is a formal property;
T	is a telic role.

While an artefact is designed for a specific purpose (its telic role), this can only be achieved under specific circumstances. Reference [4] introduces the notion of an object's "habitat", which encodes these circumstances. References [5] and [6] further define the notion of habitat and how it interacts with affordances. It is assumed that for an artefact, x, given the appropriate context C, performing the action π will result in the intended or desired resulting state, R, i.e. $C \rightarrow [\pi]R$. That is, if a context C (a set of contextual

factors) is satisfied, then every time the activity of π is performed, the resulting state *R* will occur. It is necessary to specify the precondition context *C* since this enables the local modality to be satisfied.

Using this notion, a habit is defined as representing an object situated within a partial minimal model; it is a directed enhancement of the qualia structure. Multi-dimensional affordances determine how habitats are deployed and how they modify or augment the context, and compositional operations include procedural (simulation) and operational (selection, specification, refinement) knowledge.

The habitat for an object is built by first placing it within an embedding space and then contextualizing it. For example, to use a table, the top must be oriented upward, the surface must be accessible, etc. A chair also must be oriented up, the seat must be free and accessible, it must be able to support the user, etc. An illustration of how the resulting knowledge structure for the habitat of a chair is shown in Example 2.

EXAMPLE 2

$$\lambda x \left[\begin{array}{l} \textbf{chair}\,hab \\ \mathrm{F} = [phys(x),\ on(x, y_1), in(x, y_2), orient(x,up)] \\ \mathrm{C} = [sit(x_1), back(x_2), leg(x_3), clear(x_1)] \\ \mathrm{T} = [\lambda z \lambda e[C \rightarrow [sit(e, z, x)]\ \boldsymbol{R} sit(x)] \\ \mathrm{A} = [made(e', w, x)] \end{array}\right]$$

where

F is a formal property;

C is a constitutive property;

T is a telic role;

A is an agentive role.

As described in more detail in 6.4, event or action simulations are constructed from the composition of object habitats, along with some constraints imposed by the dynamic event structure inherent in the verb itself, when interpreted as a program.

The final step in contextualizing the semantics of an object is to operationalize the telic value in its habitat. This effectively means identifying the "affordance structure" for the object.[7][8] The affordance structure available to an agent, when presented with an object, is the set of actions that can be performed with it. These are referred to as "Gibsonian affordances" and they include "grasp", "move", "hold", "turn", etc. This is to distinguish them from more goal-directed, intentionally situated activities, referred to as "telic affordances".

6 VoxML specification

6.1 Metamodel and VoxML elements

The spatio-temporal annotation schemes given in ISO 24617-1, ISO 24617-7 and ISO 24617-14 shall apply.

The metamodel, graphically depicted by Figure 1, represents a small world of basic elements modelled in VoxML. These elements form a set of categories:

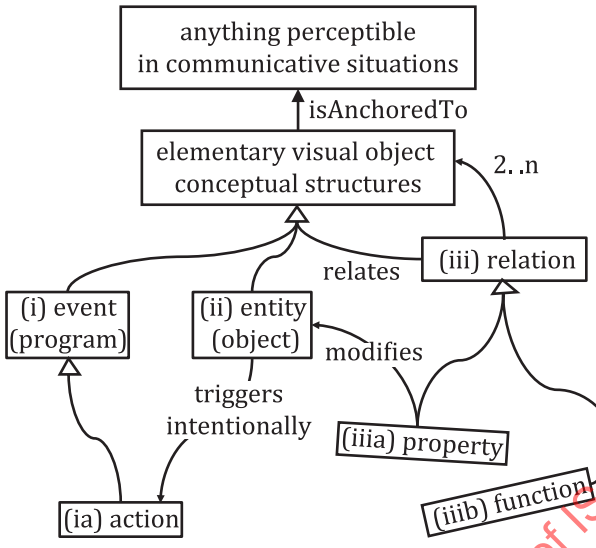
a) event (program);

b) entity (object);

c) relation over them.

Events, especially actions, work as programs while taking simple objects or spatio-temporally localized objects as arguments. Entities as objects are individuals or groups that may behave as agents. Relations can be divided into properties, often referred to as "attributes", and functions as subcategories. Attributes and relations evaluate to states, and functions evaluate to geometric regions. These elements can then compose into visualizations ns of NL concepts and expressions.

The metamodel of VoxML, presented in Figure 1, has no regions or times. These are introduced by functions such as *loc* and *τ*. The function *loc,* for instance, maps an object *x to* the region *loc(x)* to which it is anchored. Likewise, *τ(x)* maps an event to an event time, the time of its occurrence. Similarly, the function *seq* or the function *vec* maps a set of regions to a path or a vector. Thereby, the ontology of VoxML is enriched with spatio-temporal entities and dynamic paths.



NOTE 1 The empty triangular head of an arrow represents a subcategorization relation. Each directed arrow with a smaller filled-in arrowhead relates one element to one or other more elements while its labelling specifies such a relation. An entity as an agent, for example, triggers intentionally an action, while the action is a subcategory of an event, treated as a program.

NOTE 2 SOURCE: Reference [2], reproduced with the permission of the authors.

Figure 1 — Metamodel

6.2 Representation of VoxML structures

This document follows the convention of the current version of VoxML and Voxicon (see Reference [1]). Basic VoxML structures called “voxemes” are conventionally represented as feature structures, each consisting of a set of attribute-value specifications, conforming to ISO 24610-1. Voxemes are mostly formed by complex feature structures, having at least one of their substructures embedded in them as a feature structure, as illustrated in this clause.

NOTE 1 ISO 24610-1 avoids the use of the term “attribute-value”. Instead, it uses the term “feature-value”, thus defining a feature structure as a function from a set of features to a set of values.

In the concrete syntax, adopted for representing these feature structures of VoxML in this document, the names of its attributes are represented in all uppercase characters, while the names of elements start with their first character in upper case (e.g. the attribute LEX for the element Object as in Figure 2).

NOTE 2 This document follows the convention of the current version of VoxML and Voxicon for representing attribute names in upper case characters.

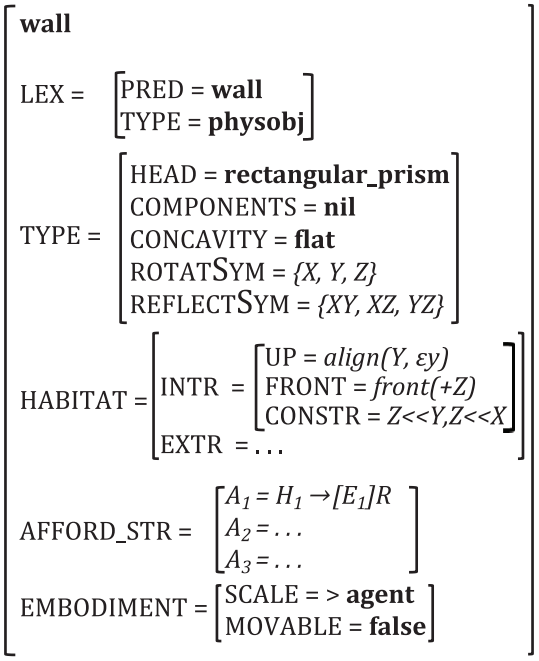


Figure 2 — Voxeme structure of a wall

6.3 Objects

The element Object in VoxML is used for modelling nouns. The current set of Object attributes is shown in Table 1.

Table 1 — Object attributes

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

The attribute LEX in Table 1 contains a substructure, specified by two attributes: PRED and TYPE. The attribute PRED in the substructure specifies the predicate lexeme denoting the Object, and the attribute TYPE in the substructure specifies the Object's type according to the Generative Lexicon[3] (see Figure 2).

There are two different sorts of the attribute TYPE, as shown in Figure 2. The first sort refers to the attribute TYPE of the element Object. In contrast, the second sort refers to the attribute TYPE of the substructure of the attribute LEX, which contains information to define the object geometry in terms of primitives. This attribute TYPE has an attribute HEAD in its substructure, which specifies a primitive 3D shape that roughly describes the object's form (such as calling an apple an "ellipsoid"), or the form of the object's most semantically salient subpart. Possible values for the attribute HEAD are grounded in, for completeness, mathematical formalism defining families of polyhedra[9], and, for the annotator's ease, common primitives found across the "corpus" of 3D artwork and 3D modelling software.

NOTE Mathematically curved surfaces such as spheres and cylinders are in fact represented, computed and rendered as polyhedra by most modern 3D software.[10]

Using common 3D modelling primitives as convenience definitions provides some built-in redundancy to VoxML, as is found in an NL description of structural forms. For example, a "rectangular_prism" is the same as a "parallelepiped" that has at least two defined planes of reflectional symmetry, meaning that an object whose Head is a rectangular_prism can be defined in two ways, an association which a reasoner can unify axiomatically. Possible values for the attribute HEAD are given in Table 2

Table 2 — Possible values for the attribute HEAD

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

These values are not intended to reflect the exact structure of a particular geometry, but rather a cognitive approximation of its shape, as is used in some image-recognition work.[11]

The substructures of an object are enumerated in its attribute COMPONENTS. In Figure 2, the attribute COMPONENTS embedded in the attribute TYPE has its value nil. Concavity can be concave, flat or convex and refers to any concavity that deforms the Head shape. ROTATSYM, or rotational symmetry, defines any of the world's three orthogonal axes around which the object's geometry may be rotated for an interval of less than 360° and retain identical form as the unrotated geometry. A sphere may be rotated at any interval around any of the three axes and retain the same form. A rectangular prism may be rotated 180° around any of the three axes and retain the same shape. An object such as a ceiling fan would only have rotational symmetry around the y-axis. Reflectional symmetry, or REFLECTSYM, is defined similarly. If an object can be bisected by a plane defined by two of the world's 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. A sphere or rectangular prism has reflectional symmetry across the XY, XZ and YZ planes. A wine bottle only has reflectional symmetry across the XY and YZ planes.

The possible values of ROTATSYM and REFLECTSYM are intended to be world-relative, not object-relative. That is, because objects are only being discussed when situated in a minimal embedding space (MES), even an otherwise empty one, wherein all coordinates are given Cartesian values, the axis of rotational symmetry or plane of reflectional symmetry are those denoted in the world, not of the object. Thus, a tetrahedron (which in isolation has seven axes of rotational symmetry, no two of which are orthogonal) when placed in the MES such that it cognitively satisfies all "real-world" constraints, is situated with one base downward (a tetrahedron placed any other way will fall over). Thus, reducing the salient in-world axes of rotational symmetry to one: the world's y-axis. When the orientation of the object is ambiguous relative to the world, the world is assumed to provide the grounding value.

The Habitat element 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 some specified actions to take place. Intrinsic faces of an object can be defined in terms of its geometry and axes. The model of a computer monitor, when axis-aligned according to 3D modelling convention, aligns the screen with the world's Z-axis facing the direction of increasing Z values. When discussing the object "computer monitor", the lexeme "front" singles out the screen of the monitor as opposed to any other part. The lexeme can therefore be correlated with the geometrical representation by establishing an intrinsic habitat of the computer monitor of *front*(*+Z*). The terminology of "alignment" of an object dimension, $d \in \{x, y, z\}$, is adopted with the dimension, d', of its embedding space, $\mathcal{E}$, as follows: *align* (d, $\mathcal{E}$, d').

The attribute AFFORD_STR describes the set of specific actions, along with the requisite conditions, that the object can potentially take part in. There are low-level affordances, called "Gibsonian", which involve manipulation or manoeuvre-based actions (grasping, holding, lifting, touching). There are also telic affordances,[3] which link directly to what goal-directed activity can be accomplished, by means of the Gibsonian affordances.

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

6.4 Actions as programs

Actions are treated as programs in VoxML for modelling eventualities. The current set of Program attributes is shown in Table 3.

Table 3 — Program attributes

LEX	Program's lexical information
TYPE	Program's event typing
EMBEDDING_SPACE	Program's embedding space as a Function of the participants and their changes over time

Just like the elements Objects, a Program's LEX attribute contains the subcomponents "PRED", the predicate lexeme denoting the program, and "TYPE", the program's type as given in a lexical semantic resource, e.g. its Generative Lexicon[3] type.

The attribute TYPE contains the attribute HEAD, its base form; Args, reference to the participants; and Body, any sub-events that are executed in the course of the program's operation. Top-level values for a Program's attribute HEAD are given in Table 4.

Table 4 — Program's attribute HEAD

HEAD	state, process, transition, assignment, test

The attribute TYPE of a program, as in Table 3, is given in terms of how the visualization of the action is realized. Basic program distinctions, such as "test" versus "assignment", are included within this typology and further distinguished through subtyping.

6.5 Relations

6.5.1 General

Like Objects and Programs, all relational categories contain a Lex attribute and a Pred parameter that denote the lexeme related to the VoxML representation. The distinction between Properties (Attributes), proper Relations and Functions lies mainly in differences in their Type structure, discussed in 6.5.2 to 6.5.4.

The terms "property" and "attribute" are used interchangeably as modifying objects. The use of the term "attribute" in 6.5.2 should not be confused with the term "Attribute" in VoxML as forming a feature structure, represented in the Attribute-Value matrix format. To avoid such a confusion, ISO 24610-1 uses the term "feature" for the term "attribute" as forming a feature structure or a feature-value structure, instead of using the term "attribute-value matrix".

6.5.2 Properties (Attributes)

Adjectival modification and predication involve reference to an Attribute in VoxML, along with a specific value for that attribute. Often, but not always, this attribute is associated with a family of other attributes, structured according to some set of constraints, called "Scale". The least constrained association is a conventional sortal classification, and its associated attribute family is the set of pairwise disjoint and non-overlapping sortal descriptions (non-super types). Following References [12] and [13], this classification is called a "nominal" scale, and it is the least restrictive scale domain over which an individual can be predicated. "Binary" classifications are a two-state subset of this domain. When more constraints are imposed on the values of an attribute, more structured domains are reached. For example, by introducing a partial ordering over values, transitive closure is obtained, assuming all orderings are defined. This is called an "ordinal" scale. When fixed units of distance are imposed between the elements on the ordering, an "interval" scale is obtained. Finally, when a zero value is introduced, a scalar structure called a "ratio" scale is obtained.

In reality, of course, there are many more attribute categories than the four listed above, but the goal in VoxML is to use these types as the basis for an underlying cognitive classification for creating measurements from different attribute types. In other words, these scale types are models of cognitive strategies for structuring values for conceptual attributes associated with NL expressions involving scalar values. VoxML encodes how attributes and the associated programs that change their values can be grouped into these scalar domains of measurement. As VoxML is intended to model visualizations of physical objects and programs, only first-order attributes are being examined, and not those that require any subjective interpretation relative to

their arguments (that is, VoxML is intended to model “the image is red” but not “the image is depressing”). Examples of different scale types are shown in Table 5.

Table 5 — Examples of different scale types

original	DIMENSION	*big, little, large, small, long, short*
binary	HARDNESS	*hard, soft*
nominal	COLOUR	*red, green, blue*
rational	MASS	*1 kg, 2 kg, etc.*
interval	TEMPERATURE	*0 °C, 100 °C, etc.*

VoxML also denotes an attribute’s arity, or the relative of the attribute to the object it describes compared to other instances of the same object class. Transitive attributes are considered to describe object qualities that require comparison to other object instances (e.g. “the small cup” versus “the big cup”), whereas intransitive attributes do not require that comparison (a “red cup” is not red compared to other cups; it is objectively red, i.e. its redness can be assessed quantitatively). Finally, every attribute is realized as applying to an object, so attributes require an ARG, a variable representing said object and the typing thereof. This is denoted identically to the individual ARGs of VoxML Programs.

6.5.3 Relations

A Relation’s TYPE structure specifies a binary class of the relation: “configuration” or “force-dynamic”, describing the nature of the relation. These classes themselves have sub-values. For configurational relations, these are values from the region connection calculus.[14] Also specified are the arguments participating in the relations. These specifications are represented as typed variables.

6.5.4 Functions

Functions’ typing structures take as the attribute ARG over which the Object voxeme is being computed. Referent takes any sub-parameters of the ARG that are semantically salient to the function, such as the voxeme’s HEAD. If unspecified, the entire voxeme should be assumed as the referent. Mapping is set to a denotation of the type of transformation the function performs over the object, such as dimensionality reduction, as represented as *dimension(n)*: *n-1* for a function that takes in an object of *n* dimensions and returns a region of *n-1*. Finally, the attribute ORIENTATION provides the following three values:

— space, which notes if the function is performed in world space or object space;

— axis, which notes the primary axis and direction of the function Boolean value of a specified input variable *x*[*y*];

— intransitive, which denotes a function that returns intransitive if the value of *y* in *x* is true.

Definitions of transitive and intransitive follow those for Attributes.

7 Examples of voxemes

7.1 General

This clause illustrates the representational capabilities of the specification by briefly presenting example voxeme entries from the current VoxML voxicon. VoxML Object representations are intended to correspond with specific voxeme geometries: each voxeme structure (e.g. Figure 3) is thus followed by a voxeme instance (e.g. Figure 4). In cases where a representation instance is given independently of a geometry, it should be assumed to denote a prototypical or “default” representation of the voxeme’s associated lexeme.

7.2 Objects

This subclause illustrates the visual object concept modelling capabilities for objects by differentiating between properties of the object's type, habitat, affordance structure and how it is embodied. Consider the voxeme structures for "wall" and "table", as represented by Figures 3 to 6.

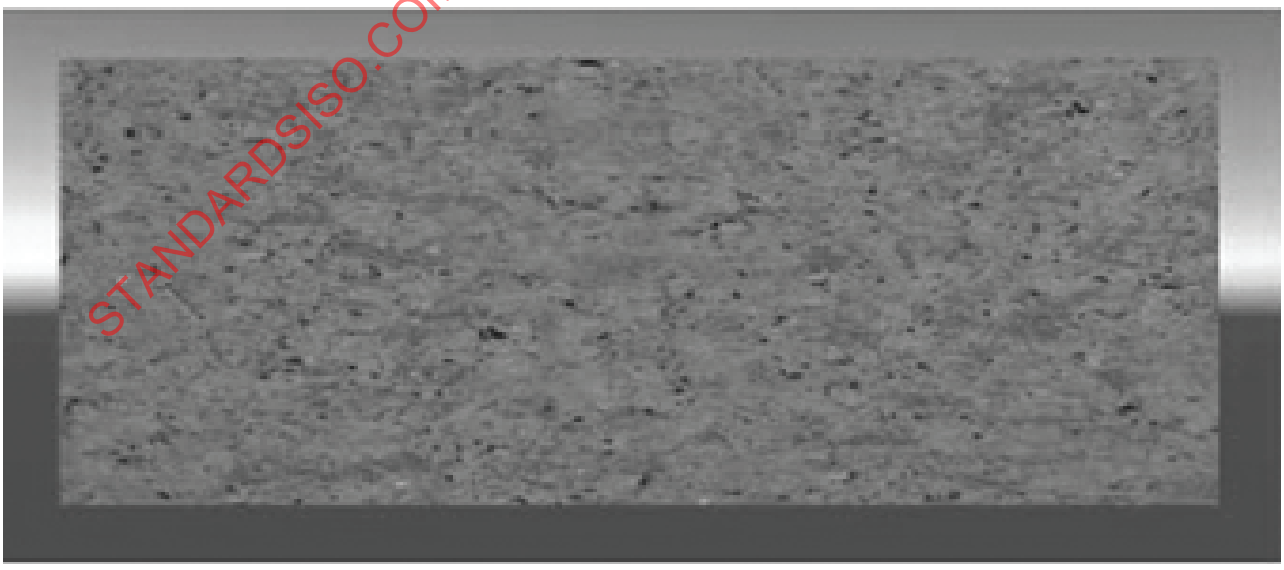
NOTE Figure 3 is the same as Figure 2.

While walls and tables are both geometries that have habitats which are upwardly aligned, tables have a head geometry of "sheet", which is a special case of "rectangular_prism" where the Y dimension is significantly less than X or Z. This head is identified in the habitat as the top of the object, facing up.

$$\begin{bmatrix} \textbf{wall} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{wall} \\ \text{TYPE} = \textbf{physobj} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{rectangular\_prism} \\ \text{COMPONENTS} = \textbf{nil} \\ \text{CONCAVITY} = \textbf{flat} \\ \text{ROTATSYM} = \{X, Y, Z\} \\ \text{REFLECTSYM} = \{XY, XZ, YZ\} \end{bmatrix} \\ \text{HABITAT} = \begin{bmatrix} \text{INTR} = \begin{bmatrix} \text{UP} = \textit{align}(Y, \varepsilon y) \\ \text{FRONT} = \textit{front}(+Z) \\ \text{CONSTR} = Z\!<\!<\!Y, Z\!<\!<\!X \end{bmatrix} \\ \text{EXTR} = \ldots \end{bmatrix} \\ \text{AFFORD\_STR} = \begin{bmatrix} A_1 = H_1 \rightarrow [E_{\downarrow}]R \\ A_2 = \ldots \\ A_3 = \ldots \end{bmatrix} \\ \text{EMBODIMENT} = \begin{bmatrix} \text{SCALE} = > \textbf{agent} \\ \text{MOVABLE} = \textbf{false} \end{bmatrix} \end{bmatrix}$$

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 3 — Voxeme structure of a wall



NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 4 — Voxeme instance of a wall

$$\begin{bmatrix} \textbf{table} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{table} \\ \text{TYPE} = \textbf{physobj} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{sheet[1]} \\ \text{COMPONENTS} = \textbf{surface[1], leg+} \\ \text{CONCAVITY} = \textbf{flat} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLECTSYM} = \{XY, YZ\} \end{bmatrix} \\ \text{HABITAT} = \begin{bmatrix} \text{INTR} = \begin{bmatrix} \text{UP} = align(Y, \varepsilon y) \\ \text{TOP} = top(+Y) \end{bmatrix} \\ \text{EXTR} = \ldots \end{bmatrix} \\ \text{AFFORD\_STR} = \begin{bmatrix} A_1 = H_1 \rightarrow [E_1]R \\ A_2 = \ldots \\ A_3 = \ldots \end{bmatrix} \\ \text{EMBODIMENT} = \begin{bmatrix} \text{SCALE} = > \textbf{agent} \\ \text{MOVABLE} = \textbf{true} \end{bmatrix} \end{bmatrix}$$

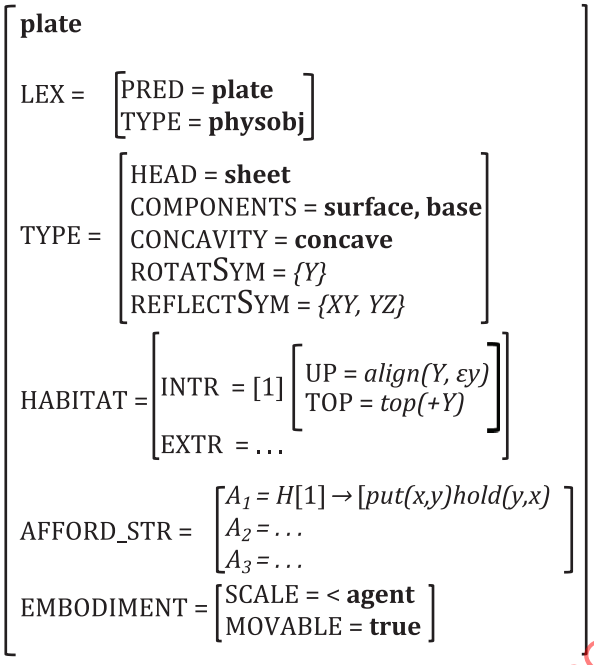
NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 5 — Voxeme structure of a table

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

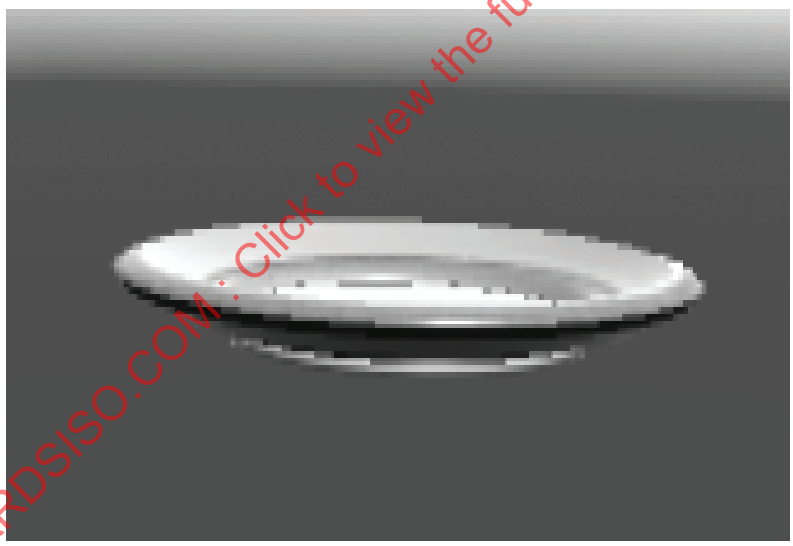
Figure 6 — Voxeme instance of a table

The voxemes for the objects “plate” and “apple” are considered in Figures 6 to 9. These illustrate how the habitat feeds into activating the affordance structure associated with the object. Namely, if the appropriate conditions are satisfied, then the attribute AFFORD_STR for telic affordance associated with a plate, for instance, is activated; every placement of *x* on *y* results in *y* holding *x*.



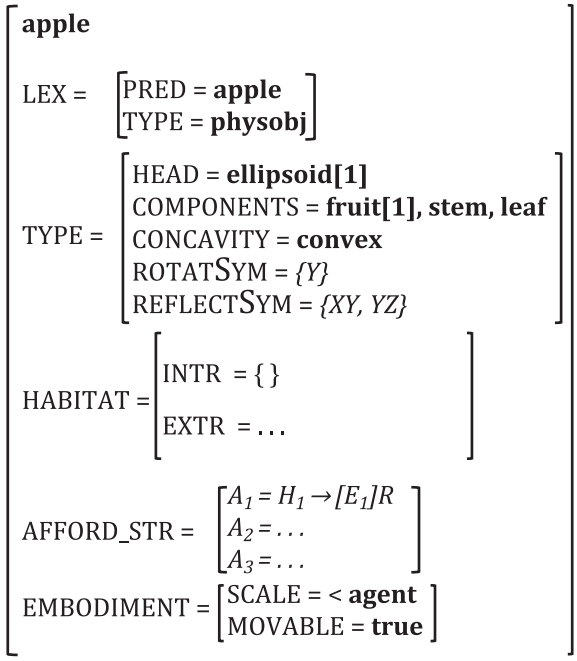
NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 7 — Voxeme structure of a plate



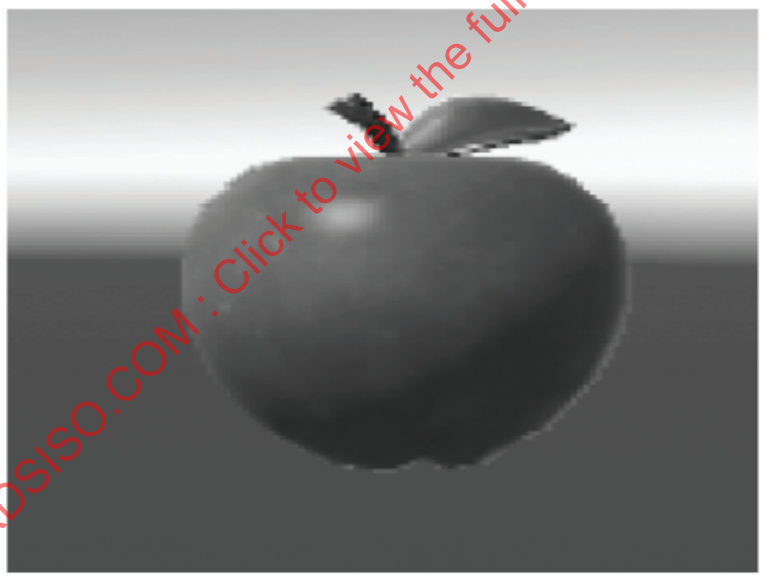
NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 8 — Voxeme instance of a plate



NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 9 — Voxeme structure of an apple



NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 10 — Voxeme instance of an apple

7.3 Eventualities as programs

Events are interpreted as programs, moving an object or changing an object property or relation from state to state, as described in more detail in 7.4. Program structure derives largely from the lexical semantics of the verb in the Generative Lexicon[3]. However, the semantics of the predicative change over the event structure is interpreted operationally. This is illustrated with two predicates as programs: “slide” and “put” in Figure 11 and Figure 12, respectively.

$$\begin{bmatrix} \textbf{slide} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{slide} \\ \text{TYPE} = \textbf{process} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{process} \\ \text{ARGS} = \begin{bmatrix} A_1 = \textbf{x:physobj} \\ A_2 = \textbf{y:physobj} \end{bmatrix} \\ \text{BODY} = [\text{E1} = \text{while(EC(x,y)), move(x)}] \end{bmatrix} \end{bmatrix}$$

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 11 — Voxeme structure of the program "slide"

$$\begin{bmatrix} \textbf{put} \\ \text{LEX} = \begin{bmatrix} \text{PRED} = \textbf{put} \\ \text{TYPE} = \textbf{transition-event} \end{bmatrix} \\ \text{TYPE} = \begin{bmatrix} \text{HEAD} = \textbf{transition} \\ \text{ARGS} = \begin{bmatrix} A_1 = \textbf{x:agent} \\ A_2 = \textbf{y:physobj} \\ A_3 = \textbf{z:location} \end{bmatrix} \\ \text{BODY} = \begin{bmatrix} E_1 = \textit{grasp}(x, y) \\ E_2 = [\textit{while}(\textit{hold}(x, y)), \\ \quad \textit{move}(x)] \\ E_3 = [\textit{at}(y, z) \rightarrow \textit{ungrasp}(x, y)] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 12 — Voxeme structure of the program "put"

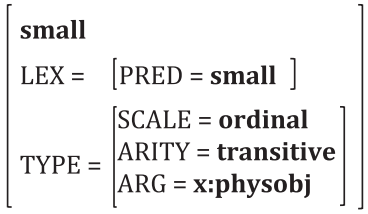
7.4 Properties

Unlike physical objects, which can be associated with specific geometries, properties (attributes) are abstract predications over distinct domains and can only be simulated by application to an element of this domain. Figure 13 and Figure 14 each provide an example of the nominal attributive interpretation of the adjective "brown", and the ordinal attributive interpretation of the adjective "small", respectively.

$$\begin{bmatrix} \textbf{brown} \\ \text{LEX} = [\text{PRED} = \textbf{brown}] \\ \text{TYPE} = \begin{bmatrix} \text{SCALE} = \textbf{nominal} \\ \text{ARITY} = \textbf{intransitive} \\ \text{ARG} = \text{x:physobj} \end{bmatrix} \end{bmatrix}$$

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 13 — Voxeme structure of the adjective "brown"

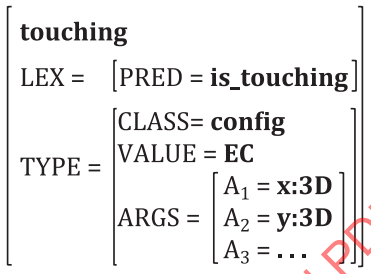


NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 14 — Voxeme structure of the adjective "small"

7.5 Relations

Spatial relations are propositional expressions, contributing configurational information about two or more objects in a state. Figure 15 shows such a spatial relation.

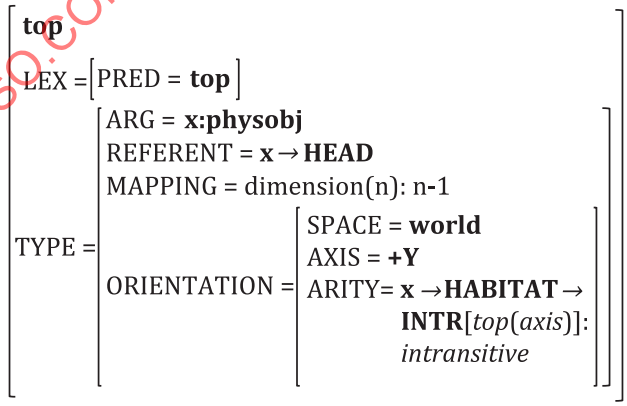


NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 15 — Voxeme structure of the relation of "touching"

7.6 Functions

This subclause illustrates the semantics of spatial functions with "top" with Figure 16. This applies to an object of dimensionality *n*, returning an object of dimensionality *n-1*. For example, the top of a cube is a plane; the top of a rectangular sheet is a line; and the top of a line is a point.



NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 16 — Voxeme structure of the spatial function "top"

8 Using VoxML for simulation modelling of language

VoxML treats objects and events in terms of dynamic event semantics, dynamic interval temporal logic (DITL).[15] The advantage of adopting a dynamic interpretation of events is that linguistic expressions can be mapped directly into simulations through operational semantics.[16][17] Models of processes using updating typically make reference to the notion of a state transition.[18][19] This is done by distinguishing between formulae, φ, and programs, π. A formula is interpreted as a classical propositional expression, with assignment of a truth value in a specific model. In this document, the dynamics of actions are represented in terms of labelled transition systems (LTSs).[18]

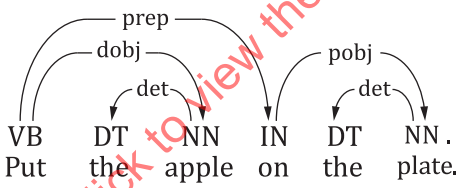
NOTE 1 This is consistent with the approach developed in References [20] and [21]. This approach to a dynamic interpretation of change in language semantics is also inspired by Reference [22].

An LTS consists of a triplet, $< S, Act, \rightarrow >$, where S is the set of states, Act is a set of actions and $\rightarrow$ is a total transition relation: $\rightarrow \subset or = S \times Act \times S$. An action, $\alpha \in Act$, provides the labelling on an arrow, making it explicit what brings about a state-to-state transition. As a shorthand for $(e_1, \alpha, e_2) \in \rightarrow$, the following is also used: $e_1 \stackrel{\alpha}{\rightarrow} e_2$.

Simulation generation begins by parsing a natural English sentence, currently using the parser named ClearNLP. [23]

NOTE 2 This document contributes to the integrating of the lexical entries with a rule-based parser that provides for a compositionally derived interpretation of the sentence being parsed.

The dependency parse is then transformed into a predicate-argument set representation using the root of the parse as the main predicate and its dependencies as arguments. Each predicate can have more than one argument and arguments may themselves be predicates (thus higher-order predicates are permissible). This predicate-argument set formula is then evaluated from the innermost first-order predicates outward until a single first-order representation is reached.



NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 17 — Dependency parser

1. pred := put(x, y) *2. x := apple* *3. y := on(z)* *4. z := plate*
put(apple, on(plate))

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 18 — Transformation to logical form

The complex argument “on(plate)” of the predicate “put” in Figure 18 evaluates to the coordinates as in: *put (apple,<1, 2,3, –0,8>).*

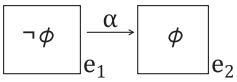
put(obj, y)
loc(obj) := x; target(obj) := y; b := x; x := w; y ≠ w; d(b,x) < d(b,w); d(b,y) > d(y,w)+

NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 19 — DITL expressions for *put*(*obj, y*)

The evaluation of predicates entails the composition of the voxemes involved. Since logical type voxemes (relations, attributes, functions) and object voxemes (nominals) are allowed for program voxemes (verbs), evaluation involves executing a snippet of code that operationalizes a program voxeme using the geometric and VoxML-encoded semantic information from the voxeme(s) indicated by the input predicate's argument(s).

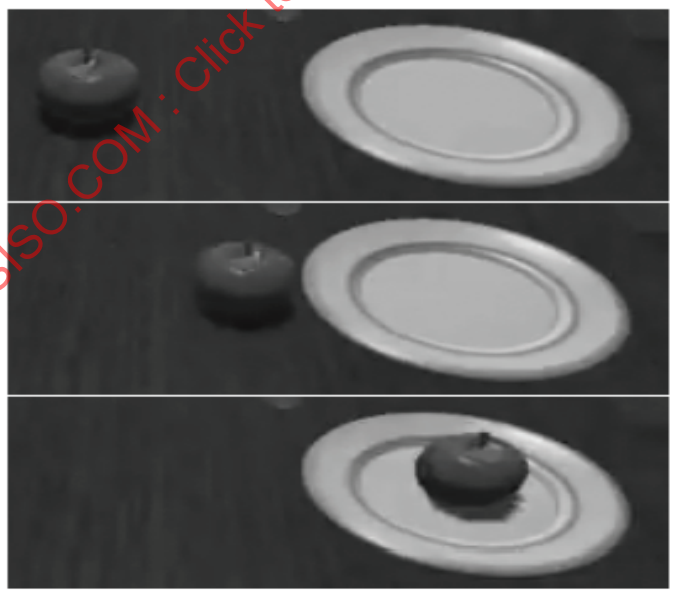
In Figure 18, the predicate "on(plate)" is evaluated to a specific location that satisfies the constraint denoted by "on the plate", using information about the "plate" object, specifically its dimensions and concavity. The program "put" can then be realized as a DITL program that moves the object "apple" toward that location until the apple's location and the location evaluated for "on(plate)" are equal, and executed as a simple state transition, as represented by Figure 20.



NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 20 — State transition

Given a 3D scene that contains voxemes for all nominals referred to in the text, the program can be operationalized in code and the state transition can be executed over the geometries visualized in 3D space. An updated version of the simulator was built for use in the context of Reference [24] and the C# language to generate visualizations like that shown in Figure 21.



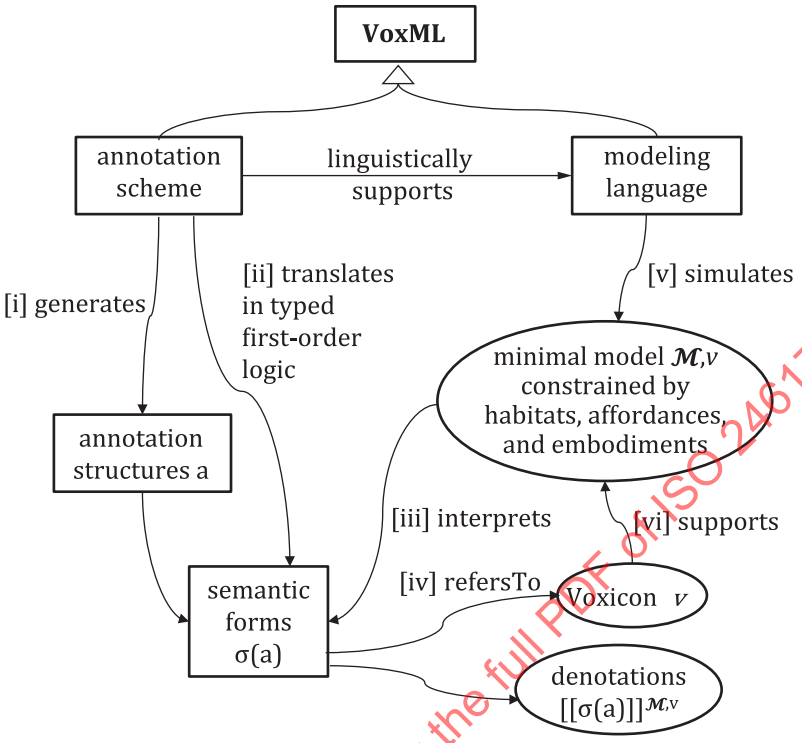
NOTE SOURCE: Reference [1], reproduced with the permission of the authors.

Figure 21 — State transition automatically generated

9 VoxML-based annotation scheme

9.1 Overview

As represented in Figure 22, VoxML provides a semantic basis for the specification of an annotation scheme.



NOTE SOURCE: Reference [2], reproduced with the permission of the authors.

Figure 22 — Annotation scheme based on VoxML

The VoxML-based annotation scheme generates annotation structures **a,** while converting them into semantic forms σ(a). Then, these semantic forms are interpreted with respect to a minimal model ***M***, constrained by habitats, affordances, and embodiments by referring the dictionary of voxemes, called "Voxicon v". Such a design of the annotation schemes provides a simplified annotation scheme with a rich semantics of interpreting annotation structures, adequate enough to make artificial agents such as robots take appropriate actions on the information thus provided.

9.2 Annotation scheme

9.2.1 Abstract specification

The annotation scheme AS_{visML} for visual information, semantically based on VoxML, is formally specified in abstract set-theoretic terms with an abstract syntax $ASyn_{visML}$. Given a set *D* of primary data, consisting of visually perceptible objects and dynamic events with various types of *n*-ary relations, including properties and functions, over them, the abstract syntax for the annotation of visual information is formulated as a triplet <*M, C,* @>, as in Definition 1.

Definition 1: The abstract syntax $ASyn_{visML}$ is formally defined as <*M, C*, @>, where:

a) *M* is a set of markable expressions in *D,* delimited by base categories in *C*;

b) *C* is a set of categories, classed into base categories, *B*, and relational (link) categories, *L*, such that *B* consists of three categories, event, entity and relation, as depicted by the metamodel (Figure 1), while *L* is an empty set, relying on the other annotation schemes such as those specified by ISO 24617-1 or ISO 24617-7;