Orientation Configuration and Dimensionality Markup Language

Annotation Guidelines Version 1.0

Cynthia Goodman Jasper Phillips Cory Massaro Zachary Yocum

March 12, 2013

Contents

1 Introduction

In this annotation task annotators must identify spatially relevant entities within descriptive texts and categorize them according to their intrinsic spatial properties, as well as specify how they are configured relative to one another. In this section we briefly outline the task goal. Section 2 enumerates the extent tags and their attributes. Section 3 explains the link tag relations and their attributes. Section 4 provides discussion of fully annotated example sentences. Finally, Section 5 explains the phased approach for this annotation task.

1.1 Goal

The annotation task is intended to capture the intrinsic spatial orientation and dimensionality of spatial entities for the purpose of making inferences about their participation in spatial relations. The ultimate aim is to be able to classify spatial entities according to the ways in which they are able to participate within spatial relations and, ideally, also be able to supply an inventory of spatial configurations based on these classes. E.g., we would want to be able to classify a hominid as a bipedal entity, which possesses an intrinsic left, right, front, back, top and bottom bounding extrema that can stand vertically 'bottom-to-top', or lay horizontally 'side-to-top', in relation to some supporting surface. The primary applications in spatial reasoning that OCDML would be suited to are text-to-scene generation applications. OCDML may also be able to augment other qualitative spatial schemes that capture topological and directional relations.

2 Extent Tags

2.1 Spatial Entity

The `spatial_entity` tag in OCDML is intended to identify participants of spatial relations. In order to be considered a participant in a spatial relation, the entity must be located in real-space. For this annotation task, annotators should ignore any entities that exist within metaphorical spaces. For instance, a metaphrical-space is introduced in Example (1a), so, although *song* and *charts* could be considered spatial entities, they should be ignored for this task. Note, however, that metaphorical-spaces are distinct from fictional-spaces, such as in Example (1b). For the purposes of this task, annotators should treat the fictional-space that is associated with the diagesis of a play, film or other literary work to be real-space. For instance, in Example (1b), annotators should consider *PAMELA*, *door*, and *staircase* to be spatial entities.

(1) a. The hit song is on top of the charts.

b. Enter [**PAMELA**$_{se1}$] from the [**door**$_{se1}$] in front of the [**staircase**$_{se1}$], ...

2.1.1 Spatial Entity Extents

For this task, the textual extents that should be tagged with the SPATIAL_ENTITY tag will be nouns. In terms of grammatical dependencies, only the heads of noun phrases should be captured with the SPATIAL_ENTITY tag. Spatial entities may be referred to by nominal or named forms, and both are valid extents for this tag type. In the case of multi-word proper names, the entire extent of the name should be included in the tag. In terms of a dependency phrase-structure, sister words and

phrases of head nouns should not be included in SPATIAL_ENTITY tag extents. The SPATIAL_ENTITY tag is allowed to be non-consuming. In cases of implicit spatial entities that are not directly referred to in the text, annotators should create non-consuming tags. Refer to Example (2) in section 2.1.3 for illustrations of various SPATIAL_ENTITY extent tags.

Note:

2.1.2 Spatial Entity Attributes

The OCDML SPATIAL_ENTITY tag inherits some attributes from other annotation schemes, but only a subset of these attributes are relevant for this task. The attributes which annotators should annotate for this task are `dimensionality`, `line_type`, `area_type`, `volume_type`, `left_right`, `front_back`, `top_bottom`, and `c-sec_axis`. These attributes are listed in Table 1. Annotators do not need to fill the other attributes that are defined in the Document Task Definition, including `form`, `latLong`, `mod`, `countable`, `amount`, `quant`, and `scopes`.

Attribute	**Value**
`id`	`se1`, `se2`, `se3`, ...
`dimensionality`	POINT, LINE, AREA, VOLUME
`mod`	A spatially relevant modifier
`line_type`	SEGMENT, RAY, LINE, LOOP, OTHER
`area_type`	3-GON, 4-GON, DISC, ANNULUS, OTHER
`volume_type`	TRI_PRISM, RECT_PRISM, PYRAMID, SPHERE, TORUS, CYLINDER, CONE, BIPED, QUADRUPED, OTHER
`left_right`	INTRINSIC or RELATIVE
`front_back`	INTRINSIC or RELATIVE
`top_bottom`	INTRINSIC or RELATIVE
`c-sec_axis`	LEFT_RIGHT or FRONT_BACK or TOP_BOTTOM or OTHER

Table 1: SPATIAL_ENTITY Attributes

`dimensionality` This attribute is used to designate the number of spatial dimensions occupied by the entity. For the purposes of OCDML annotation, take spatial entities to be point-sets. As such, a value of POINT indicates a 0-dimensional entity consisting of a single point with no edges or surfaces. A value of LINE indicates a 1-dimensional entity with up to two bounding points, or vertices, with a single edge and no surfaces. A value of AREA indicates a 2-dimensional entity with some number of bounding edges, points and two surfaces. A value of VOLUME indicates a 3-dimensional entity with at least one surface and possibly a number of bounding edges and points.

If a value of POINT is specified for a SPATIAL_ENTITY tag, then it is not necessary to specify the other attributes, since they are not relevant to 0-dimensional entities. If a value of LINE is specified, then the `line_type` attribute must be filled. If a value of AREA is specified then at least the `area_type` attribute must be filled, and the `line_type` could be filled with whatever would be appropriate if the entity were coerced to a line for the purposes of participating in a relation. Finally, if the the VOLUME type is specified, then at least the `volume_type` attribute must also be filled, and the `area_type` and `line_type` attributes might also be filled.

Note: Although annotators must choose a single value for the `dimensionality` attribute, spatial entities may be coerced to different dimensionalities depending on the spatial relations they are participating in. E.g., a *door* might be considered volumetric, however it may be coerced to 2-dimensions when participating in some relationships. In that case, it would be appropriate to fill both the `volume_type` and `area_type` attributes. Refer to section 3.1 for more discussion of this type of coercion.

`line_type` This attribute is used to classify the entity based on a set of 1-dimensional primitive types. Some lexical items that are good candidates for the `line_type` value of SEGMENT would be a *clothesline*, a piece of *wire*, or a singular strand of *hair*. These are examples of linear entities which have distinguishable endpoints. We include the RAY type to capture entities such as a *ray* of light that would have only one distinguishable endpoint. The LINE type would be appropriate for a *row* or *queue* that may have no particularly distinguishable endpoints, but also is not a loop. The LOOP type would be appropriate for an *equator* or other latitudinal *line* of demarcation on a planet, or for the *border* of a contiguous region, which has no endpoints, but also loops back on itself. A value of OTHER may be specified when none of the previously mentioned values are appropriate.

`area_type` This attribute is used to classify the entity based on a set of 2-dimensional primitive types. The 3-GON type is used to indicate that the entity is triangular, possessing three distinguishable bounding edges and points such as a triangular *slice* of pizza. Similarly, the 4-GON type is used to indicate a 2-dimensional primitive with four bounding edges and points, such as a rectangular *piece* of paper. The DISC type is used for 2-dimensional entities with a single bounding edge, and no distinguishable points, such as a round *coaster*. The ANNULUS type is appropriate for a 2-dimensional entity, such as a compact *disc*, that has two distinguishable bounding edges and no distinguishable bounding points. A value of OTHER may be specified when none of the previously mentioned values are appropriate.

`volume_type` This attribute is used to classify the entity based on a set of 3-dimensional primitive types:

TRI_PRISM should be used for 3-dimensional entities that can be thought of as 3-gons that are extruded into a 3-dimensional prism and have five distinguishable bounding surfaces and six distinguishable bounding points. A *wedge* of cheese or a *box* of Toblerone chocolates would both be appropriate entities to be tagged with the TRI_PRISM type.

RECT_PRISM is used for 3-dimensional entities that are 3-dimensionally extruded 4-gons with six distinguishable bounding surfaces and eight boinding points. This type includes entities such as a standard six-sided *die*, a closed *book* or *tome*, or a prototypical *box*.

PYRAMID is used for 3-dimensional entities with five distinguishable bounding surfaces, including a 4-gon and four 3-gons, in addition to five bounding points, including a single apex point.

SPHERE is given to 3-dimensional spheroidal entities with no distinguishable bounding edges or vertices, such as a *globe*, *orange*, or *ball*.

TORUS is used for 3-dimensional entities with no distinguishable bounding edges or vertices, but is topologically distinct from a SPHERE type by virtue of having a hole. The distinction is analogous to the difference between the 2-dimensional DISC and ANNULUS primitive types, with SPHERE corresponding to the former and TORUS corresponding to the latter. A *doughnut*, a hoola *hoop*, or a wedding *band* would all be examples of entities that would take a value of TORUS for their `volume_type`.

CYLINDER indicates a 3-dimensional entity with three distinguishable surfaces, two distinguishable bounding edges, and zero vertices. A soda *can*, a dumbbell *bar*, and a AA *battery* are all examples that fall under this volume type.

CONE should be given for 3-dimensional entities with two distinguishable surfaces, one bounding edge, and a single apex point. An ice-cream *cone*, a *funnel*, or a coniferous *tree* would all be appropriate entities to annotate with the CONE type.

BIPED is used for 3-dimensional entities with two distinguishable feet. An *ostrich*, a *kangaroo*, or a *person* are all excellent candidates for this `volume_type`.

QUADRUPED is used for 3-dimensional entities with four distinguishable feet. Animals such as *horses*, *cats* or *dogs* are good examples of quadrupedal spatial entities.

The `volume_type` attribute is not intended to capture every entity perfectly. Rather, it is intended to identify distinguishable topological features which are accessed when the entity participates within a spatial configuration relation. If none of the previously described values are appropriate, a value of OTHER may be specified.

`left_right` This attribute is a bit that indicates whether the entity possesses an intrinsic axis of orientation whose polar extremes are 'left' and 'right'. An entity should be considered to possess an intrinsic left-right axis if it has left-hand and right-hand sides that are distinguishable independent of any frame of reference. One heuristic which may help to determine whether a spatial entity possesses an intrinsic `left_right` axis is the linear-array-test. The linear-array-test can be employed by asking "Could duplicates of this entity be arranged in a 1-dimensional array from 'left-to-right', i.e., such that the 'left' boundary belonging to each entity in the array abuts the 'right' boundary of another?" If the answer is "no", then the `left_right` attribute should probably be annotated as RELATIVE; if "yes", the value would likely be INTRINSIC.

`front_back` This attribute is similar to the `left_right` attribute. A value of INTRINSIC indicates the entity possesses an intrinsic axis of orientation whose polar extremes are 'front' and 'back'. A value of RELATIVE indicates the opposite. The linear-array-test heuristic can be modified for this case such that arrangement of the array would be considered 'front-to-back'.

`top_bottom` This attribute is similar, again, to both `left_right` and `front_back`. A value of INTRINSIC indicates the entity possesses an intrinsic axis of orientation whose polar extremes are 'top' and 'bottom'. The linear-array-test can be applied for this attribute as well to test if the entities can possibly be stacked 'top-to-bottom'.

`c-sec_axis` This attribute is intended to identify the salient cross-sectional axis of the spatial entity. Only entities with at least one intrinsic axis of orientation, should have a `c-sec_axis` value specified. The point of specifying the cross-sectional axis is to distinguish between spatial entities such as a typical twelve-ounce *can* of soda and a military-style *submarine*. Each of these entities would be classified as volumetric, cylindrical primitive types, and both possess intrinsic left, right, front, back, top, and bottom boundaries, yet they possess salient cross-sectional axes that do not correspond to one another. For the soda *can*, the salient cross-sectional axis—the axis along which the 2-dimensional circular base would be extruded—is the TOP_BOTTOM axis. For the *submarine*, contrastively, it is the FRONT_BACK axis which corresponds to the cross-sectional axis along which the cylindrical primitive form would be extruded. Figure 1 illustrates this distinction. Another way to conceptualize the salient cross-sectional axis would be to imagine slicing the entity into pieces as if to skewer the pieces with a kebab. Under this conceptualization, the axis that is aligned with the imaginary kebab is the axis which should be filled for the `c-sec_axis` attribute.

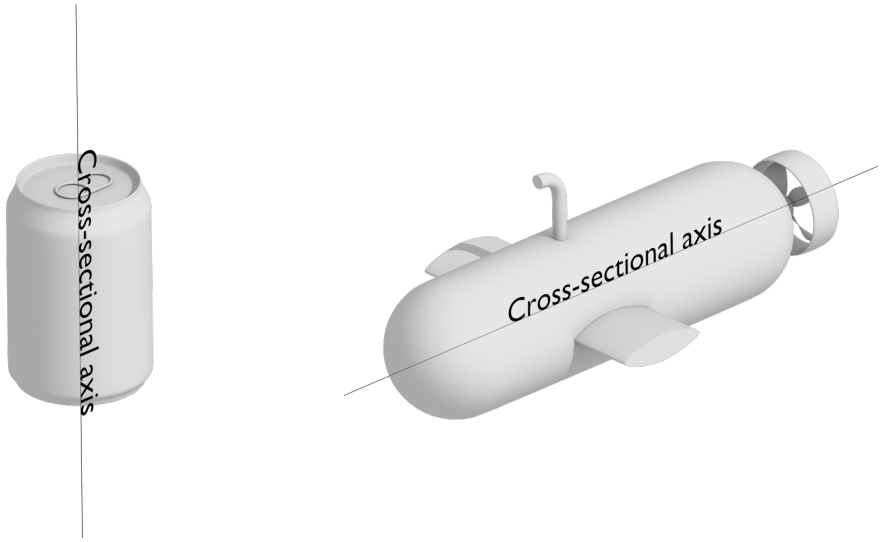

Figure 1: Salient Cross-sectional Axes

2.1.3 Spatial Entity Examples

SPATIAL_ENTITY tag extents have been identified in Example (2) followed by pseudo-XML annotations for each tag's attributes.[1]

(2) a. On the large [**table**$_{se1}$] stands a lighted [**lamp**$_{se2}$].

SPATIAL_ENTITY (`id`=`se1`, `dimensionality`=VOLUME, `mod`="large", `line_type`=∅, `area_type`=4-GON, `volume_type`=RECT_PRISM, `left_right`=RELATIVE, `front_back`=RELATIVE, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

SPATIAL_ENTITY (`id`=`se2`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=DISC, `volume_type`=CYLINDER, `left_right`=RELATIVE, `front_back`=RELATIVE, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

[1]The symbol ∅ is used in these examples to explicitly indicate an empty, or unspecified value.

b. [$_{se0}$] Lies down on [**sofa**$_{se3}$].

Note: In this example, `se0` is a non-consuming SPATIAL_ENTITY tag that is configured relative to the *sofa*.

SPATIAL_ENTITY (`id`=`se0`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=∅, `volume_type`=∅, `left_right`=∅, `front_back`=∅, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

SPATIAL_ENTITY (`id`=`se3`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=4-GON, `volume_type`=RECT_PRISM, `left_right`=INTRINSIC, `front_back`=INTRINSIC, `top_bottom`=INTRINSIC, `c-sec_axis`=LEFT_RIGHT)

c. [**MAURICE**$_{se4}$] and [**HENRIETTE**$_{se5}$] are in evening dress and sit facing each other ...

Note: Normally the pronoun *other* would be tagged as a SPATIAL_ENTITY, but in this example only the extents for *MAURICE* and *HENRIETTE* have been tagged. In cases where a spatial entity is referenced anaphorically via a pronominal form it is normally appropriate to tag the extent of the pronoun with the SPATIAL_ENTITY tag. However, if a split antecedent occurs where each part of the antecedent participates in a spatial configuration, acting as both a figure and ground relative to others, then it is safe to only tag the antecedents. If *MAURICE* and *HENRIETTE* were both facing something else then that 3rd entity would be tagged as a SPATIAL_ENTITY.

SPATIAL_ENTITY (`id`=`se4`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=∅, `volume_type`=BIPED, `left_right`=INTRINSIC, `front_back`=INTRINSIC, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

SPATIAL_ENTITY (`id`=`se5`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=∅, `volume_type`=BIPED, `left_right`=INTRINSIC, `front_back`=INTRINSIC, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

d. In the foreground a [**table**$_{se6}$] is spread, with [**flowers**$_{se7}$] in the centre, [**bowls**$_{se8}$] full of [**fruit**$_{se9}$], [**wine**$_{se10}$] in [**decanters**$_{se11}$], [**oysters**$_{se12}$] on [**platters**$_{se13}$] ...

SPATIAL_ENTITY (`id`=`se6`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=4-GON, `volume_type`=RECT_PRISM, `left_right`=RELATIVE, `front_back`=RELATIVE, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

SPATIAL_ENTITY (`id`=`se7`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=∅, `volume_type`=OTHER, `left_right`=RELATIVE, `front_back`=RELATIVE, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

SPATIAL_ENTITY (`id`=`se8`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=DISC, `volume_type`=OTHER, `left_right`=RELATIVE, `front_back`=RELATIVE, `top_bottom`=INTRINSIC, `c-sec_axis`=TOP_BOTTOM)

SPATIAL_ENTITY (`id`=`se9`, `dimensionality`=VOLUME, `mod`=∅, `line_type`=∅, `area_type`=∅, `volume_type`=OTHER, `left_right`=RELATIVE, `front_back`=RELATIVE, `top_bottom`=RELATIVE, `c-sec_axis`=∅)

SPATIAL_ENTITY (id=se10, dimensionality=VOLUME, mod=∅, line_type=∅, area_type=∅, volume_type=OTHER, left_right=RELATIVE, front_back=RELATIVE, top_bottom=RELATIVE, c-sec_axis=∅)

SPATIAL_ENTITY (id=se11, dimensionality=VOLUME, mod=∅, line_type=∅, area_type=DISC, volume_type=OTHER, left_right=RELATIVE, front_back=RELATIVE, top_bottom=INTRINSIC, c-sec_axis=TOP_BOTTOM)

SPATIAL_ENTITY (id=se12, dimensionality=VOLUME, mod=∅, line_type=∅, area_type=∅, volume_type=OTHER, left_right=INTRINSIC, front_back=INTRINSIC, top_bottom=INTRINSIC, c-sec_axis=TOP_BOTTOM)

SPATIAL_ENTITY (id=se13, dimensionality=VOLUME, mod=∅, line_type=∅, area_type=DISC, volume_type=CYLINDER, left_right=RELATIVE, front_back=RELATIVE, top_bottom=INTRINSIC, c-sec_axis=TOP_BOTTOM)

2.2 Orientation Signal Tag

An orientation signal is taken to be some word or phrase that restricts the potential configurational interpretations of a spatial entity. Orientation signals always trigger configuration links, so, if an annotator tags an orientation signal, then they are committing to also creating a configuration link.

Sentences in Example (3) have had ORIENTATION_SIGNAL extents identified.

(3) a. On the large table [**stands**$_{os1}$] a lighted lamp.

b. [**Lies**$_{os2}$] down on sofa.

c. MAURICE and HENRIETTE are in evening dress and [**sit**$_{os3}$] [**facing**$_{os4}$] each other ...

d. In the foreground a table is spread, with flowers in the centre, bowls full of fruit, wine in decanters, oysters on platters ...

Here, in Example (3a), the signal *stands* serves to adumbrate in what configuration the lamp is arranged individually. Because the lamp *stands* rather than, e.g., *lies* or *leans*, one immediately recognizes that the lamp is situated in such a way that its top-bottom axis is aligned vertically.

Likewise, in Example (3b), the signal *lies* indicates that some entity—in this case an implicit spatial entity, which would need to be created via a non-consuming SPATIAL_ENTITY tag—is configured in such a way that its top-bottom orientational axis is horizontally aligned.

2.2.1 Orientation Signal Extents

In general, whereas SPATIAL_ENTITY terms will be nominals, ORIENTATION_SIGNAL terms will belong to other parts of speech, e.g., verbs, prepositions, adjectives, and adverbs. The extent for orientation signals should consist of the head of the phrase, so for phrasal verbs, take only the head verb as the extent for the orientation signal, ignoring any particles. For prepositional phrases, only the head preposition would be tagged.

Attribute	Value
id	os1, os2, os3, ...
orientation_type	LATITUDINAL, LONGITUDINAL, LATERAL, VERTICAL, OTHER

Table 2: Attributes for ORIENTATION_SIGNAL

2.2.2 Orientation Signal Attributes

The orientation signal tag has one attribute that annotators need to consider, as shown in table 2:

orientation_type The orientation_type attribute specifies an axis of orientation that is accessed by the orientation signal. The LATITUDINAL value refers to the left-right axis, the LONGITUDINAL to the front-back axis, and the VERTICAL to the top-bottom axis. The LATERAL value is supplied to capture cases in which a horizontal axis is accessed, yet there is no distinction between the left-right and the front-back axes. The value of OTHER should be selected for such cases as specify a configuration not covered by the remaining values.

2.2.3 Orientation Signal Examples

Provided below, in Example (4), are sample annotations for extents to be tagged as ORIENTATION_SIGNAL:

(4) a. On the large table [**stands**$_{os1}$] a lighted lamp.

b. ... [**lies**$_{os2}$] down on sofa.

c. MAURICE and HENRIETTE are in evening dress and [**sit**$_{os3}$] [**facing**$_{os4}$] each other ...

d. In the foreground a table is spread, with flowers [**in**$_{os5}$] the centre, bowls [**full**$_{os6}$] of fruit, wine [**in**$_{os7}$] decanters, oysters [**on**$_{os8}$] platters ...

3 Link Tags

3.1 Configuration Link Tag

The CONFIGURATION_LINK relates two SPATIAL_ENTITY tags. For each CONFIGURATION_LINK, one SPATIAL_ENTITY will be referred to as the figure in the relation, and the other will be referred to as the ground. The figure is the spacial entity that moves or is located in relation to the ground. Similarly, the ground is the spacial entity against which the figure moves, or in relation to which the figure is located. In Example (5a), *cat* is the figure and *table* is the ground, since the *cat* is located in relation to the *table*. In Example (5b), *man* is the figure and *street* is the ground, since the *man* is moving in relation to the *street*. The figure is typically the smaller, more movable object, but this is not always the case, as can be seen in Example (5c). In English, the figure tends to come before the ground in the sentence. While this is a helpful heuristic, it should not be the sole basis of your annotation decisions, as is evident from Example (5d).

(5) a. The [cat$_{figure}$] is sitting on the [table$_{ground}$].

b. The [man$_{figure}$] walked down the [street$_{ground}$].

c. The [house$_{figure}$] landed on the [witch$_{ground}$].

d. ...sitting at the [desk$_{ground}$] was a beautiful [woman$_{figure}$].

3.1.1 Configuration Link Attributes

The attributes for the CONFIGURATION_LINK tag are enumerated in table 3.

Attribute	Value
`id`	`cl1`, `cl2`, `cl3`, ...
`figure_config`	LEFT, RIGHT, FRONT, BACK, TOP, BOTTOM, SIDE, TOP_OR_BOTTOM, ANY, OTHER
`ground_config`	LEFT, RIGHT, FRONT, BACK, TOP, BOTTOM, SIDE, TOP_OR_BOTTOM, ANY, OTHER
`coercion`	FIGURE, GROUND, or NONE
`trigger`	An ID of a `orientation_signal` tag that triggers the link
`direction`	FIGURE_GROUND, GROUND_FIGURE

Table 3: Attributes for CONFIGURATION_LINK

`figure_config` & `ground_config` The combination of the `figure_config` and `ground_config` attributes are used to specify the configuration between the participating figure and ground entities.

(6) a. The [cat$_{figure}$] is sitting on the [table$_{ground}$].

b. The [cat$_{figure}$] is sitting alone [$_{ground}$].

`dim_coercion` This attribute can be used to indicate that the dimensionality of the figure entity is coerced to that of the ground or vice versa. Consider Example (7a). While we know that the sun is a 3-dimensional sphere, in this example the *sun* is being coerced into 2-dimensions by the ground, namely the *sky*. This type of coercion does not take place in Example (7b), since both the *earth* and the *sun* would be considered 3-dimensional objects moving in 3-dimensional space.

(7) a. The [sun$_{figure}$] travels across the [sky$_{ground}$].

b. The [earth$_{figure}$] rotates around the [sun$_{ground}$].

`trigger` This attribute takes the ID value of the `orientation_signal` that triggered the `configuration_link`. Recall that creating an `orientation_signal` means committing to creating a CONFIGURATION_LINK, however, there may be situations where a CONFIGURATION_LINK is created without any triggering signal.

`direction` This attribute is used to indicate the asymmetrical directionality of the configuration relation in terms of figure-to-ground or ground-to-figure. When creating a CONFIGURATION_LINK, if the ID of the SPATIAL_ENTITY tag that represents the figure is selected first, then the `direction` attribute should take a value of FIGURE_GROUND. If the link were created in the reverse order, such that the ID of the SPATIAL_ENTITY tag that represents the ground is selected first, then the `direction` attribute should take a value of GROUND_FIGURE.

4 Fully Annotated Examples

5 Phased Annotation Approach

For this task we will be dividing the annotation into three sections as enumerated in the outline below. This phased approach is intended to establish a consistent set of entity and signal extent tags early in the process. For the first annotation phase, annotators will simply be identifying SPATIAL_ENTITY and SPATIAL_SIGNAL tag extents. After the first annotation phase, a cursory round of adjudication will be performed in order to lock in a consistent set of extent tags. Once this consistent set of extents has been frozen, annotators will fill the attributes for those extent tags. Finally, during the third phase of annotation, annotators will create the links between the locked set of extent tags. After the third annotation phase the final adjudication will occur.

Annotation Phase 1 Extents for SPATIAL_ENTITY and ORIENTATION_SIGNAL tags.

Extents Adjudication Extent tags will be adjudicated and locked.

Annotation Phase 2 Attributes for locked SPATIAL_ENTITY and ORIENTATION_SIGNAL tags.

Annotation Phase 3 Creation of CONFIGURATION_LINKS annotation of link attributes.

Final Adjudication Link tags and all tag attributes will be adjudicated.