CogSketch
Spatial Reasoning

Madeline Usher
Ken Forbus
Andrew Lovett
Matthew McLure

Version of 2/11/2016
# CONTENTS

1  **INTRODUCTION** 4
   1.1  **ACCESSING SPATIAL REASONING FACILITIES** 4
   1.2  **BRIEF OVERVIEW OF REPRESENTATION AND REASONING IN COGSKETCH** 4

2  **RCC-8 RELATIONS** 5

3  **HIGHER-LEVEL TOPOLOGICAL RELATIONS** 7

4  **VORONOI DIAGRAM RELATIONS** 8

5  **POSITIONAL RELATIONS** 9
   5.1  **ABOUT THE GYLPHS** 9
   5.2  **ABOUT THE OBJECTS REPRESENTED BY THE GYLPHS** 10

6  **GYLPH-GROUPS** 13
   6.1  **CONTAINMENT** 13
   6.2  **CONNECTNESS** 13

7  **MISCELLANEOUS GEOMETRY** 14

8  **VISUAL/CONCEPTUAL RELATIONS** 17

9  **RELATIONS USED IN SPATIAL ROUTINES** 18
   9.1  **OBJECT ATTRIBUTES** 18
   9.2  **SPATIAL RELATIONS** 19
   9.3  **PATTERN OF VARIANCE RELATIONS** 20
   9.4  **SPATIAL TRANSFORMATION ATTRIBUTES** 21

10  **DECOMPOSITIONS** 23
   10.1  **PREDICATES FOR GENERATING DECOMPOSITIONS** 23
   10.2  **EDGE-CYCLE ATTRIBUTES** 24
   10.3  **EDGE-CYCLE RELATIONS** 26
   10.4  **EDGE ATTRIBUTES** 27
1 INTRODUCTION

This is a very preliminary draft of documentation describing the spatial reasoning facilities built into CogSketch. We are making this draft documentation available to support those who wish to experiment with these facilities as they evolve.

1.1 ACCESSING SPATIAL REASONING FACILITIES

If you just want to try things out, a good way to get started is to use the Query window available through the knowledge inspector or by selecting the “View/Query Window” menu option. You’ll find the concordance of objects to names in the left-hand frame useful for formulating queries.

These facilities are also available through the KQML server, via the ASK operation.

1.2 BRIEF OVERVIEW OF REPRESENTATION AND REASONING IN COGSKETCH

Mostly yet to be written. Here are the absolutely key points.

1.2.1 STRUCTURE OF SKETCHES

Internally, each subsketch corresponds to a reasoning context (i.e. a microtheory). In our knowledge-representations we use:

\[(\text{ist-Information} \ <\text{subsketch-context} > \ <\text{fact}>)\]

1.2.2 TRUTH-MAINTENANCE CONSIDERATIONS

All results are cached in the LTRE that serves as FIRE’s working memory, so that they can be used in explanations (cf the drill-down in the knowledge inspector). This also enables results to be appropriately retracted when the sketch is changed. This is why spatial conclusions tend to include statements about the time the ink of the glyphs involved were last updated. Whenever a glyph is moved, the now obsolete last-ink assumption is retracted and a new assumption made. This updating is part of what is happening when ink processor (displayed as the left eye in the interface) is operating.
2 RCC-8 RELATIONS

The RCC-8 relations are a set of eight mutually exclusive relations that describe all possible topological relations between two 2D closed shapes.¹

![RCC-8 Relations Diagram]

(rcc8-DC glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
disconnected

(rcc8-EC glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
edge-connected

(rcc8-PO glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
partial overlap

(rcc8-EQ glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
equal

(rcc8-TPP glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
tangential proper-part

(rcc8-TPPi glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
Inverse tangential proper part

(rcc8-NTPP glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
Non-tangential proper-part

(rcc8-NTPPi glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
Inverse non-tangential proper-part

(hasRCC8Relation glyph1 glyph2 rel)
Automatically computed between pairs of glyphs on same layer.
Can be generated on demand by fire:ask.
Says that the given RCC8 relation holds for the two glyphs.
3 Higher-Level Topological Relations

CogSketch can compute several higher-level topological relations, based on the RCC-8 relations described above. The first three are computed automatically and are used during comparison (e.g., in worksheets and other applications). The final two can be computed on demand. Unlike RCC-8, these relations are not mutually exclusive. For example, if one glyph contains another, they may also intersect. If two glyphs are overlapping, they will necessarily intersect.

(containsObject glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Says that glyph1 contains glyph2. This is based on the RCC-8 relations rcc8-TPP and rcc8-NTPP. Note that transitive containment is avoided. That is, if (containsObject glyph1 glyph2) and (containsObject glyph2 glyph3) hold, then (containsObject glyph1 glyph3) will not hold.

(objectsIntersect glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Says that the glyph1 and glyph2's ink intersect. This is based on the RCC-8 relations rcc8-EC, rcc8-PO, and rcc8-TPP. Thus, if the relation rcc8-TPP holds, then both containsObject and objectsIntersect will hold.

(objectsOverlap glyph1 glyph2)
Automatically computed between pairs of glyphs on same layer.
Says that the interiors of glyph1 and glyph2 overlap. This is based on the RCC-8 relation rcc8-PO.

(symmetricBisection glyph1 glyph2)
Can be generated on demand by fire:ask.
Says that the two glyphs overlap one another to the extent that each bisects the other.

(objectContains-Transitive glyph1 glyph2)
Can be generated on demand by fire:ask.
Says the same thing as objectContains, except that transitive containment is preserved. I.e., if glyph1 contains glyph2, and glyph2 contains glyph3, then (objectContains-Transitive glyph1 glyph3) will hold, whereas (objectContains glyph1 glyph3) will not.
4 VORONOI DIAGRAM RELATIONS

The following facts are derived from the voronoi diagrams computed for a subsketch. Note that more than one voronoi may be computed for a given subsketch — rules in our knowledge-base which voronoi diagrams get computed. For example, in nuSketch Battlespace voronoi diagrams are made for the terrain information, pathways (avenues of approach, etc.), and the enemy and friendly military units. In CogSketch, we currently just compute one voronoi diagram for each subsketch, using all the glyphs on the subsketch as input.

(vAdjacent glyph1 glyph2)
Can be generated on demand by fire:ask.
Says that the two glyphs are adjacent to each other on at least one of the voronoi diagrams containing both glyphs. The two glyphs are considered adjacent if at least one voronoi site (cell) of glyph1 is touching a site of glyph2.

(vNear glyph1 glyph2)
Can be generated on demand by fire:ask.
Says that the two glyphs are near each other on at least one of the voronoi diagrams containing both glyphs. The notion of 'near' is a bit looser than that of 'adjacent'. To compute a region of nearness around a glyph, we include all the immediately adjacent sites and expand outwards into sites whose area is mostly encompassed by the adjacent sites.

(vBetween glyph1 glyph2 glyph3)
Can be generated on demand by fire:ask.
Says that glyph2 is between glyph1 and glyph3 on at least one of the voronoi diagrams containing both glyphs.
5  POSITIONAL RELATIONS

5.1  ABOUT THE GLYPHS

(above glyph1 glyph2)
  Can be generated on demand by fire:ask.
Says that glyph1 is above glyph2 in the sketch coordinate system (Y-values increase as you go upwards; higher values are above lower values).

(below glyph1 glyph2)
  Can be generated on demand by fire:ask.
Says that glyph1 is below glyph2 in the sketch coordinate system (Y-values increase as you go upwards; higher values are above lower values). Note that above is our canonical representation -- Calling nusketch:compute-positional-relations where some glyph A is below of some other glyph B will not generate a below fact but will instead generate (above B A).

(rightOf glyph1 glyph2)
  Can be generated on demand by fire:ask.
Says that glyph1 is to the right of glyph2 in the sketch coordinate system (X-values increase as you go towards the right).

(leftOf glyph1 glyph2)
  Can be generated on demand by fire:ask.
Says that glyph1 is to the left of glyph2 in the sketch coordinate system (X-values increase as you go towards the right). Note that rightOf is our canonical representation -- Calling nusketch:compute-positional-relations where some glyph A is left of some other glyph B will not generate a leftOf fact but will instead generate (rightOf B A).

(positionalRelationFor glyph1 glyph2 rel)
  Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two glyphs. For glyphs, rel will be one of above, below, rightOf, or leftOf.

(enclosesVertically glyph1 glyph2)
  Can be generated on demand by fire:ask.
Says that glyph2 is completely within the vertical bounds determined by the minimum Y-coordinate and maximum Y-coordinate of glyph1.

(enclosesHorizontally glyph1 glyph2)
  Can be generated on demand by fire:ask.
Says that glyph2 is completely within the horizontal bounds determined by the minimum X-coordinate and maximum X-coordinate of glyph1.

(directionalSignature glyph1 glyph2 sig)
  Can be generated on demand by fire:ask.
Low-level presentation of the position relation between two glyphs. The signature is a set of one or more numbers indicating the region in which glyph2 is, relative to glyph1. 1 is directly right of glyph1, 2 is right and above, 3 is directly above, etc.
(occludes glyph1 glyph2 glyph3)
   Can be generated on demand by fire:ask.
Says that glyph1 is in a position between glyph2 and glyph3 such that it occludes each from the other.
In cases of partial occlusion, this relationship holds if any of the following three conditions is true: (1) More than 50% of the area of glyph1 lies in the “visibility region” between glyph2 and glyph3, i.e. more than 50% of glyph1 participates in the occlusion, (2) glyph1 occludes more than 50% of glyph2 from glyph3, or (3) glyph1 occludes more than 50% of glyph3 from glyph2.

5.2 ABOUT THE OBJECTS REPRESENTED BY THE GLYPHS

5.2.1 PHYSICAL-VIEW GENRE, LOOKING-FROM-SIDE POSE

(above obj1 obj2)
   Can be generated on demand by fire:ask.
Says that obj1 is above obj2.

(below obj1 obj2)
   Can be generated on demand by fire:ask.
Says that obj1 is above obj2. Note that above is our canonical representation -- Calling nusketch:compute-positional-relations where some object A is below of some other object B will not generate a below fact but will instead generate (above B A).

(rightOf obj1 obj2)
   Can be generated on demand by fire:ask.
Says that obj1 is to the right of obj2.

(leftOf obj1 obj2)
   Can be generated on demand by fire:ask.
Says that obj1 is to the left of obj2. Note that rightOf is our canonical representation -- Calling nusketch:compute-positional-relations where some object A is left of some other object B will not generate a leftOf fact but will instead generate (rightOf B A).

(positionalRelationFor obj1 obj2 rel)
   Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.

(enclosesVertically obj1 obj2)
   Can be generated on demand by fire:ask.
Says that obj2 is completely within the vertical bounds determined by the minimum Y-coordinate and maximum Y-coordinate of obj1.

(enclosesHorizontally obj1 obj2)
   Can be generated on demand by fire:ask.
Says that obj2 is completely within the horizontal bounds determined by the minimum X-coordinate and maximum X-coordinate of obj1.

5.2.2 PHYSICAL-VIEW GENRE, LOOKING-FROM-TOP POSE

(positionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.

5.2.3 PHYSICAL-VIEW GENRE, LOOKING-FROM-BOTTOM POSE

(positionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.

5.2.4 GEOSPATIAL-VIEW GENRE, LOOKING-FROM-TOP POSE

(eastOf obj1 obj2)
Can be generated on demand by fire:ask.

(northEastOf obj1 obj2)
Can be generated on demand by fire:ask.

(northOf obj1 obj2)
Can be generated on demand by fire:ask.

(northWestOf obj1 obj2)
Can be generated on demand by fire:ask.

(westOf obj1 obj2)
Can be generated on demand by fire:ask.

(southOf obj1 obj2)
Can be generated on demand by fire:ask.

(southWestOf obj1 obj2)
Can be generated on demand by fire:ask.

(southEastOf obj1 obj2)
Can be generated on demand by fire:ask.

(positionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.

(compassPositionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
rel will be one of eastOf, northEastOf, northOf, northWestOf, westOf, southWestOf, southOf, or southEastOf.

(compassCentroidRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
This is similar to compassPositionalRelationFor but only the centroids of the two glyphs are used to compute the compass positional relation, ignoring the blob boundaries.
5.2.5 GEOSPATIAL-VIEW GENRE, LOOKING-FROM-SIDE POSE
(positionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.

5.2.6 GEOSPATIAL-VIEW GENRE, LOOKING-FROM-BOTTOM POSE
(positionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.

5.2.7 ABSTRACT-VIEW GENRE, UNSPECIFIED-VIEWPOINT POSE
(positionalRelationFor obj1 obj2 rel)
Can be generated on demand by fire:ask.
Says that the given positional relation holds between the two objects.
6 Glyph-Groups

6.1 Containment

(isa (ContainedGlyphGroupFn outside-glyph (TheList inside-glyph1 inside-glyph2 ...)) ContainedGlyphGroup)
Automatically computed between glyphs on a subsketch.
Reifies the glyphs contained inside another glyph, thus allowing us to reason about the group as an entity.

(containedGlyphGroupInsider inside-glyph glyph-group)
Automatically computed between glyphs on a subsketch.
Says that inside-glyph is one of the glyphs contained inside in the specified ContainedGlyphGroup.

(containedGlyphGroupTangentialInsider inside-glyph glyph-group)
Automatically computed between glyphs on a subsketch.
Says that inside-glyph is one of the glyphs contained inside in the specified ContainedGlyphGroup and is touching the interior wall of the containing glyph.

(containedGlyphGroupContainer outside-glyph glyph-group)
Automatically computed between glyphs on a subsketch.
Says that outside-glyph is the glyph containing the inside glyphs in the specified ContainedGlyphGroup.

6.2 Connectness

(isa (ConnectedGlyphGroupFn (TheList glyph1 glyph2 ...)) ConnectedGlyphGroup)
Automatically computed between glyphs on a subsketch.
Reifies a group of connected glyphs.

(connectedGlyphGroupMember glyph glyph-group)
Automatically computed between glyphs on a subsketch.
Says that glyph is one of the glyphs contained inside in the specified ConnectedGlyphGroup.

(connectedGlyphGroupTangentialConnection glyph1 glyph2 glyph-group)
Automatically computed between glyphs on a subsketch.
Says that glyph1 and glyph2 are connected because their outsides are touching and that they belong to the specified ConnectedGlyphGroup.

(glyphGraphEdgesFor glyph-graph glyph (TheList connected-glyph1 connected-glyph2 ...))
Automatically computed between glyphs on a subsketch.
Says that glyph is a node in glyph-graph whose arcs consist of links to the glyphs in the given list.
7 MISCELLANEOUS GEOMETRY

(areaOfObject object area)
Can be generated on demand by fire:ask.
Gives the area of the object in "real-world" units (i.e. if the glyph is a square that is one unit wide by one
unit tall, and each unit represents 10 Km, then the area would be (Kilometer 100) ). The area is
computed using the blob boundary of the glyph that represents the specified object.

(calculateReferencePoint object point)
Can be generated on demand by fire:ask.
Specifies the centroid of the object, in "real-world" units. The centroid is computed from blob boundary
of the glyph that represents the specified object.

(distanceBetween obj1 obj2 distance)
Can be generated on demand by fire:ask.
Returns the distance between the glyphs representing the two objects.

(entityCentroidOnPath obj path-glyph)
Can be generated on demand by fire:ask.
(entityCentroidOnPath ?probe ?path) is true exactly when ?probe is close enough to some point on
?path, as determined by ?path's width.

(entityLocatedAt obj location-obj)
Can be generated on demand by fire:ask.
Object is located at location, meaning inside or overlapping or touching.

(entityLocatedInside obj location-obj)
Can be generated on demand by fire:ask.
Object is located inside location.

(facingRelation obj1 obj2 facingRelationSpecifier)
Can be generated on demand by fire:ask.
Says that the specified relation holds between the facing directions of Obj1 and Obj2. Acceptable
relations here are onLeftFlank-Facing, onRightFlank-Facing, onRear-Facing, and onFront-Facing.

(fixedAxisRotation situation1 situation2 obj1 origin)
Can be generated on demand by fire:ask.
Indicates that, in situation2, obj1 has been rotated around origin without translation from its orientation
in situation1.

(blobIntersectsConvexHull glyph1 glyph2)
Can be generated on demand by fire:ask.
Says that the blob boundary of glyph1 intersects the convex hull of glyph2.

(lengthAlongMajorAxis obj distance)
Can be generated on demand by fire:ask.
Gives the length of an object along its major axis in "real-world" units.
(widthAcrossMajorAxis SpatialThing Distance)  
*Can be generated on demand by fire:ask.*  
Gives the width of an object across its major axis in "real-world" units.

(majorAxisAngle glyph angle)  
*Can be generated on demand by fire:ask.*  
Represents the major axis angle, relative to the current 2D orientation of the glyph.

(outsideNormal glyph surface qualitative-vector)  
*Can be generated on demand by fire:ask.*  
Says that the vector is the outside normal for the specified surface of a glyph. The surface is an outside edge of a glyph computed via glyph decomposition.

(overlappingParts glyph1 segment1 glyph2 segment2)  
*Can be generated on demand by fire:ask.*  
Says that glyph1 and glyph2 overlap and specifies the line segments where the overlap occurs.

(qualitativeVectorBetween NuSketchPoint NuSketchPoint QualitativeVector)  
*Can be generated on demand by fire:ask.*  
qualitativeDirection is true iff the third argument is the qualitative direction from the first argument to the second argument.

(relativePositionAlong Path-Spatial SpatialThing-Localized SpatialThing-Localized PathRelativePositionRelation)  
*Can be generated on demand by fire:ask.*  

(relativePositionChange situation1 situation2 obj1 obj2 qualitative-vector)  
*Can be generated on demand by fire:ask.*  
Gives the qualitative change in the vector difference between obj2 and obj1 that happened between situation1 and situation2.

(shortestRotationDir situation1 situation2 obj rotation-dir)  
*Can be generated on demand by fire:ask.*  
Indicates the direction in which obj rotated between situation1 and situation2.

(surfaceOverlapDirToPoint QMSurface QMSurface QMPoint QualitativeVector)  
*Can be generated on demand by fire:ask.*  
surfaceOverlapDirToPoint is true iff the qvector is the direction from the midpoint of the overlap of the two surfaces to the qpoint.

(ConvexHullFn glyph)  
A functional wrapper that can be wrapped around any glyph when querying for a topological relationship. The query will effectively replace the glyph in the query with its convex hull.

(surfaceContactDirection glyph1 glyph2 direction)  
*Can be generated on demand by fire:ask.*
Says that glyph1 has a surface contact with glyph2, where the direction of contact is in the specified qualitative direction.

\(\text{glyphFromGlyphIntersection } \text{glyph1 } \text{glyph2 layer new-glyph}\)  
*Can be generated on demand by fire:ask.*
Says that the new-glyph, to be placed on layer, is made from the intersection of the regions of glyph1 and glyph2

\(\text{glyphFromGlyphUnion } \text{glyph1 } \text{glyph2 layer new-glyph}\)  
*Can be generated on demand by fire:ask.*
Says that the new-glyph, to be placed on layer, is made from the union of the regions of glyph1 and glyph2

\(\text{glyphFromGlyphDifference } \text{glyph1 } \text{glyph2 layer new-glyph}\)  
*Can be generated on demand by fire:ask.*
Says that the new-glyph, to be placed on layer, is made from the region of glyph1 that isn’t overlapping glyph2
8 **Visual/Conceptual Relations**

For many reasoning tasks it is helpful to clarify for CogSketch what relationship holds between the objects depicted based on the visual relationships that hold between the glyphs that depict them. We call these visual/conceptual relationships, since they are derived from both kinds of information. For example, two glyphs that touch could indicate that they are rigidly connected, hinged, or any of a number of other possibilities, depending on the specific kinds of entities involved. To specify these relationships for a sketch, click on the icon. A browser window will open, providing a form which enables you to choose a specific relationship for each pair of entities that have a form of visual relationship. When you are finished selecting relationships, click the submit button. If you need to change any of the VCR relationships, check the retract box underneath that relationship and click the submit button. You should get the drop-down menu again and you can re-assign the relationship between the involved glyphs.

Some things to know about this computation:

- The set of possible candidates can be quite large or quite small, depending on the visual and conceptual properties of the entities involved. Having over one hundred possibilities is not uncommon. Sorting through them can be complex, and we recommend having a KB Browser window open on the side while you are familiarizing yourself with the options.
- Only relations whose argument type constraints are consistent with the declared types of the entities are presented as possibilities. No other information is currently used to filter the set, e.g., it does not try to derive each of the possibilities and its negation using the information in the sketch plus the KB knowledge.
- The only visual relationships between glyphs that currently trigger this computation are touching and insideness (i.e., RCC8-EC, RCC8-PO, RCC8-TPP, RCC8-NTPP). If you do not get possible relationships for a pair of glyphs, check to see if one of these visual relationships holds, and if not, redraw or move one of them.
- If the set of relationship options does not include something that you think should be there, check to see if that relationship is consistent with the types of entities involved, using the KB Browser.
9  RELATIONS USED IN SPATIAL ROUTINES

The following four tables describe the attributes and relations used by Spatial Routines during cognitive modeling simulations (e.g., see Perceptual Sketchpad and Geometric Analogy in the User Manual). The first two tables describe the attributes and relations used to represent an image, i.e., a sketch drawn in CogSketch or a psychological stimulus imported into CogSketch. The latter two tables describe how routines represent a pattern-of-variance. A pattern-of-variance describes the differences found between images. It is used by Spatial Routines in some problem-solving tasks, such as geometric analogy and Raven’s Progressive Matrices.

9.1  OBJECT ATTRIBUTES

Each object in an image (i.e., each glyph in CogSketch) is assigned a set of attributes from the following list.

<table>
<thead>
<tr>
<th>Type</th>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Shape Type</td>
<td>2D-Shape-Generic&lt;sup&gt;A&lt;/sup&gt;</td>
<td>Any visible 2D shape</td>
</tr>
<tr>
<td></td>
<td>Implicit-Shape&lt;sup&gt;A&lt;/sup&gt;</td>
<td>A shape implied by negative space</td>
</tr>
<tr>
<td></td>
<td>Proximal-Similar-Shape-Group&lt;sup&gt;A&lt;/sup&gt;</td>
<td>A group formed of several similar closed shapes that are about equally distant from each other</td>
</tr>
<tr>
<td>Specific Shape Type</td>
<td>Dot-Shape&lt;sup&gt;A&lt;/sup&gt;</td>
<td>Special shape: dot (any very small glyph)</td>
</tr>
<tr>
<td></td>
<td>VerticalEdge/HorizontalEdge&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape consisting of a single straight edge, horizontally or vertically aligned</td>
</tr>
<tr>
<td>Closedness</td>
<td>2D-Shape-Closed&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Any closed shape</td>
</tr>
<tr>
<td></td>
<td>2D-Shape-Open&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Any open shape</td>
</tr>
<tr>
<td>Convexity</td>
<td>2D-Shape-Convex&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Closed shape without any concavities</td>
</tr>
<tr>
<td>Curvedness</td>
<td>2D-Shape-Curved&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape with only curved edges</td>
</tr>
<tr>
<td></td>
<td>2D-Shape-Straight&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape with only straight edges</td>
</tr>
<tr>
<td>Junctions</td>
<td>2D-Shape-Forked&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape with junctions where 3+ edges meet</td>
</tr>
<tr>
<td>Symmetry</td>
<td>Symmetric-Shape&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape with one axis of symmetry</td>
</tr>
<tr>
<td></td>
<td>Perpendicular-Symmetric-Shape&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape with two perpendicular axes of symmetry</td>
</tr>
<tr>
<td></td>
<td>Fully-Symmetric-Shape&lt;sup&gt;A,s&lt;/sup&gt;</td>
<td>Shape with two perpendicular axes of symmetry, plus at least one more axis, and with a fairly even aspect ratio</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Multiply-Symmetric-Shape (^A, S)</td>
<td>Other shape with multiple axes of symmetry</td>
<td></td>
</tr>
<tr>
<td>Symmetric-Blob</td>
<td>Shape with no detected symmetry axes, but with no axes of elongation, suggesting an overall regular shape</td>
<td></td>
</tr>
<tr>
<td>Alignment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D-Shape-AxisAligned (^A)</td>
<td>Shape whose edges are aligned with x- and y-axes</td>
<td></td>
</tr>
<tr>
<td>2D-Shape-Oblique (^A)</td>
<td>Shape whose edges are diagonal (non-axis-aligned)</td>
<td></td>
</tr>
<tr>
<td>Relative Size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TinySizeGlyph (^A)</td>
<td>Tiny closed shape (based on distribution of sizes in a context)</td>
<td></td>
</tr>
<tr>
<td>Small/Medium/LargeSizeGlyph (^A)</td>
<td>&quot;</td>
<td></td>
</tr>
<tr>
<td>Relative Location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Centered-Element (^A)</td>
<td>Element located at the center of an image</td>
<td></td>
</tr>
<tr>
<td>OnLeft-Element/OnTop-Element (^A)</td>
<td>Element located at the left/top side of an image</td>
<td></td>
</tr>
<tr>
<td>Relative Edge Width</td>
<td></td>
<td></td>
</tr>
<tr>
<td>narrowEdged/wideEdged</td>
<td>Describes shapes whose edges are narrow or wide, relative to other shapes in the context</td>
<td></td>
</tr>
<tr>
<td>Fill Color</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ObjectsColoredFn X)</td>
<td>Shape with fill color X</td>
<td></td>
</tr>
<tr>
<td>Edge Color</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ObjectsBorderColoredFn X)</td>
<td>Shape with edges of color X</td>
<td></td>
</tr>
<tr>
<td>Texture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TexturedObject</td>
<td>Object with some fill texture (a set of parallel straight lines inside it)</td>
<td></td>
</tr>
</tbody>
</table>

A: These attributes describing shape or location are important in guiding the comparison between images. However, when Spatial Routines represent the pattern-of-variance between images, they are abstracted out. They are replaced with changes in spatial relations and with shape transformations (see below).

S: These attributes describe rotation-invariant features of an object’s shape.

### 9.2 Spatial Relations

These are the relations used by spatial routines.

<table>
<thead>
<tr>
<th>Type</th>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative position</td>
<td>(above X Y)</td>
<td>Object X is above object Y</td>
</tr>
<tr>
<td></td>
<td>(rightOf X Y)</td>
<td>Object X is right of object Y</td>
</tr>
<tr>
<td>Topology</td>
<td>(elementContains X Y)</td>
<td>Closed shape X contains object Y</td>
</tr>
<tr>
<td></td>
<td>(elementsIntersect X Y)( ^S )</td>
<td>The edges of objects X and Y intersect</td>
</tr>
<tr>
<td></td>
<td>(elementsOverlap X Y) ( ^S )</td>
<td>The interiors of closed shapes X and Y overlap</td>
</tr>
</tbody>
</table>
Frame of reference relations

- *(centeredOn X Y)* Object X contains object Y. Object X creates a frame of reference in which object Y’s relative location can be encoded. In this case, object Y is located at the center of object X.

- *(onLeft/Right/Top/BottomHalfOf X Y)* ""

Alignment relations

- *(parallelElements X Y)* X and Y are single edges (are rows of objects). They are parallel.

- *(perpendicularElements X Y)* ""

- *(collinearElements X Y)* X and Y are single edges and are collinear, or X is a single edge and Y is a small object that lies along its extension.

- *(centeredOn X Y)* This relation can also indicate X is a single edge (or set of edges) and Y lies at its center.

Transformation relations

- *(reflectedShapes-XAxis X Y)/ (reflectedShapes-YAxis X Y)* Indicates that there is a reflection between the two objects’ shapes.

- *(reflectedShapes X Y)* Indicates that there is some other reflection.

- *(rotatedShapes-90 X Y)/ (rotatedShapes-180 X Y)* Indicates there is a rotation between the two objects’ shapes.

- *(rotatedShapes X Y)* Indicates that there is some other rotation.

S: Symmetric relations in which the order of the arguments is irrelevant. E.g., *(elementsOverlap X Y) => (elementsOverlap Y X)*

### 9.3 Pattern of Variance Relations

When a spatial routine computes a pattern of variance between images, it compares the images’ visual representations, containing the attributes and relations described above. It produces a set of relations describing changes between the images. These relations are given below.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(changeBetweenImages fact Image-X Image-Y)</em></td>
<td>Indicates that a fact holds in Image-Y but not in Image-X.</td>
</tr>
<tr>
<td><em>(reversalBetweenImages fact-X fact-Y Image-X Image-Y)</em></td>
<td>Indicates that a relation reverses between Images X and Y. For example, Object-A may be above Object-B in Image-X, but below Object-B in Image-Y.</td>
</tr>
<tr>
<td><em>(changeBetweenImagesFromTo fact-X fact-Y Image-X Image-Y)</em></td>
<td>Indicates an object transformation between Images X and Y. For example, an object might rotate, or it might change shape entirely. See the next table for a list of possible transformations.</td>
</tr>
</tbody>
</table>
(holdsInImage fact Image-X) Indicates a that fact holds in Image-X. This relation may be used instead of changeBetweenImages in special cases.

### 9.4 Spatial Transformation Attributes

When a spatial routine computes a pattern of variance between images, it represents changes to an object's shape with the relation:

\[ \text{(changeBetweenImagesFromTo (isa Element-1 Spatial-trans-attribute) (isa Element-1 Spatial-trans-attribute) Image-X Image-Y)} \]

The attributes may describe a transformation such as a rotation, a deformation such as becoming longer, or a total change in shape. A generalized form allows the Structure-Mapping Engine (our model of comparison) to generalize when comparing representations. For example, it may be possible to align (isa Object-A ScaledShape-Bigger) with (isa Object-1 DeformedShape-Longer) because the two attributes share the generalized form TransformedShape-Larger.

<table>
<thead>
<tr>
<th>Type</th>
<th>Generalized Form</th>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shape Type</strong></td>
<td></td>
<td>(ShapeTypeFn id)</td>
<td>A shape of type ( \text{id} ) (id’s are arbitrary numbers assigned to the shape types encountered in a given context).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ShapeTransFn id trans scaling)</td>
<td>A shape of type ( \text{id} ) that has been transformed and/or scaled from the prototype for that type (e.g., a circle that’s bigger than the circles we’ve seen before).</td>
</tr>
<tr>
<td><strong>Rotation</strong></td>
<td></td>
<td>RotatedShape-90</td>
<td>A shape that’s been rotated 90 degrees.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RotatedShape-X</td>
<td>Some other rotation (rounds to the nearest 45 degrees).</td>
</tr>
<tr>
<td><strong>Reflection</strong></td>
<td></td>
<td>ReflectedShape-XAxis</td>
<td>A shape that’s been reflected over the ( \text{x} )-axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ReflectedShape-YAxis</td>
<td>A shape that’s been reflected over the ( \text{y} )-axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ReflectedShape-DiagAxis</td>
<td>A shape that’s been reflected over one of the two diagonal, 45-degree axes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ReflectedShape-Other</td>
<td>A shape that’s been reflected over some other axis.</td>
</tr>
<tr>
<td><strong>Scaling</strong></td>
<td></td>
<td>ScaledShape-Bigger</td>
<td>A shape that’s become bigger.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ScaledShape-Smaller</td>
<td>A shape that’s become smaller.</td>
</tr>
<tr>
<td>Deformation</td>
<td>Transformed Shape-Larger</td>
<td>DeformedShape-Longer</td>
<td>A shape that’s become longer along one dimension.</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
<td>----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Transformed Shape-Larger</td>
<td>DeformedShape-LongerPart</td>
<td></td>
<td>One part of a shape has become longer.</td>
</tr>
<tr>
<td>Transformed Shape-Larger</td>
<td>DeformedShape-AddedPart</td>
<td></td>
<td>A shape that’s had a part added to it.</td>
</tr>
<tr>
<td>Transformed Shape-Smaller</td>
<td>DeformedShape-Shorter</td>
<td></td>
<td>A shape that’s become shorter along one dimension.</td>
</tr>
<tr>
<td>Transformed Shape-Smaller</td>
<td>DeformedShape-ShorterPart</td>
<td></td>
<td>One part of a shape has become shorter</td>
</tr>
<tr>
<td>Transformed Shape-Smaller</td>
<td>DeformedShape-RemovedPart</td>
<td></td>
<td>A shape that’s had a part removed from it.</td>
</tr>
<tr>
<td>Addition</td>
<td>-----</td>
<td>AddedShape</td>
<td>A shape that’s been added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RemovedShape</td>
<td>A shape that’s been removed.</td>
</tr>
<tr>
<td>Other</td>
<td>-----</td>
<td>ConstantShape</td>
<td>A shape that’s remained unchanged.</td>
</tr>
</tbody>
</table>

NOTE: If a shape remains constant throughout the images being compared, it is not represented at all in the pattern of variance. On the other hand, if a shape remains constant between two images and then transforms in a third image, then its remaining constant between the first two images will be represented using ConstantShape.
10 Decompositions

CogSketch can visually decompose a glyph into networks of edges/junctions and edge-cycles. Edges are segments of ink with relatively constant curvature that meet at junctions, and edge-cycles are closed sequences of edges that surround regions of whitespace and perimeters of edge-connected objects - islands of connected ink within the glyph. Viewing an edge/junction decomposition as a graph embedding, the edge-cycles correspond to its faces and the edge-connected objects correspond to its connected components.

A set of edge-cycle representations, which include some of the shape attributes from Section 9.1, can be generated for a glyph directly. A separate predicate is available for computing edge-cycle representations in which perceptually similar nearby cycles are grouped into texture regions, which are also described in terms of their edge-cycles. Edge representations can be generated for an edge-cycle, either by describing the relationships between, and attributes of, edges as they trace around the cycle, or by constructing a skeleton using the medial axis transform of the edge-cycle polygon. The next section outlines the predicates that generate these representations and Section 10.2 details the attributes and relations that show up in the generated representations.

10.1 Predicates for Generating Decompositions

The following predicates can be queried to generate decomposition representations. In general, they take six arguments. To cause the decomposition to be constructed, the first three arguments should be ground and the latter three should be variables. The first argument is the object to be decomposed into parts. The second and third are parameters that control the flexibility with which CogSketch will consider (a) edges to be perpendicular or parallel, and (b) objects to be the same size\(^2\). They should be integers ranging from [-1,2]. The fourth, fifth, and sixth arguments will be bound respectively to the microtheory (i.e. context) where the representations were stored, the set of names for perceptual entities (edge-cycles or edges) mentioned in the generated representations, and the number of facts generated.

(edgeCycleRepresentationsFor glyph angle-mod size-mod microtheory cycle-names num-facts)  
Can be generated on demand by fire:ask.

Says that the structured representations of the edge-cycles in glyph, as generated with angle-threshold modifier angle-mod and size-threshold modifier size-mod (described above), are stored in the context microtheory, describe the edge-cycles cycle-names, and span the number of facts num-facts.

(edgeCycleRepresentationsFor-Textured glyph angle-mod size-mod microtheory cycle-names num-facts)  
Can be generated on demand by fire:ask.

Says that the structured representations of the edge-cycles in glyph, as generated with angle-threshold modifier angle-mod and size-threshold modifier size-mod (described above) and grouped into textures based on local perceptual similarities (e.g. orientation, area, complexity, edge-continuity), are stored in the context microtheory, describe the edge-cycles cycle-names, and span the number of facts num-facts.

\(^2\) CogSketch uses a simple linear thresholding scheme to put relative sizes into four buckets (e.g. lengthTiny, lengthShort, lengthMedium, lengthLong). However, if it considers two objects to be close enough to the same size and they straddle a threshold, it will avoid splitting them into two separate buckets by allowing the threshold to shift temporarily.
Can be generated on demand by fire:ask. Says that the structured representations of the edges in an edge-cycle, as generated with angle-threshold modifier angle-mod and size-threshold modifier size-mod (described above), are stored in the context microtheory, describe the edges edge-names, and span the number of facts num-facts.

Can be generated on demand by fire:ask. Says that the structured representations of the edges along the skeleton of the closed regions in edge-cycle, as generated with angle-threshold modifier angle-mod and size-threshold modifier size-mod (described above), are stored in the context microtheory, describe the edges edge-names, and span the number of facts num-facts. The skeleton is generated using a version of the medial axis transform that is filtered using the exterior segmentation (the edges in edge-cycle) and carved at qualitative transitions in the radius function (the distance from each point in the skeleton to the closest point along the exterior).

### 10.2 Edge-cycle Attributes

Edge-cycles are assigned attributes from the following catalog. There is significant overlap with the object attributes from Section 9.1.

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Shape Type</strong></td>
<td>PerceptualEdgeCycle</td>
<td>Any visible edge-cycle</td>
</tr>
<tr>
<td></td>
<td>PerceptualEdgeCycle-Perimeter</td>
<td>An edge-cycle that traces the perimeter of an edge-connected object or a texture</td>
</tr>
<tr>
<td></td>
<td>PerceptualEdgeCycle-Atomic</td>
<td>An edge-cycle that does not contain any other edge-cycles from the same edge-connected object (it may contain other edge-connected objects, and thus, their edge-cycles)</td>
</tr>
<tr>
<td><strong>Specific Shape Type</strong></td>
<td>Dot-Shape</td>
<td>Special shape: dot (any very small glyph)</td>
</tr>
<tr>
<td><strong>Closedness</strong></td>
<td>2D-Shape-Closed</td>
<td>Any edge-cycle that forms a closed shape</td>
</tr>
<tr>
<td></td>
<td>2D-Shape-Open</td>
<td>Any edge-cycle that forms an open shape</td>
</tr>
<tr>
<td></td>
<td>2D-Shape-PartiallyClosed</td>
<td>Any edge-cycle that forms an open shape with some closed parts</td>
</tr>
<tr>
<td><strong>Convexity</strong></td>
<td>2D-Shape-Convex</td>
<td>Closed shape without any concavities</td>
</tr>
<tr>
<td>Curvedness</td>
<td>2D-Shape-Curved</td>
<td>Shape with only curved edges</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>2D-Shape-Straight</td>
<td></td>
<td>Shape with only straight edges</td>
</tr>
<tr>
<td>Junctions</td>
<td>2D-Shape-Forked</td>
<td>Shape with junctions where 3+ edges meet</td>
</tr>
<tr>
<td>Symmetry</td>
<td>Symmetric-Shape</td>
<td>Shape with one axis of symmetry</td>
</tr>
<tr>
<td></td>
<td>Perpendicular-Symmetric-Shape</td>
<td>Shape with two perpendicular axes of symmetry</td>
</tr>
<tr>
<td></td>
<td>Fully-Symmetric-Shape</td>
<td>Shape with two perpendicular axes of symmetry, plus at least one more axis, and with a fairly even aspect ratio</td>
</tr>
<tr>
<td></td>
<td>Multiply-Symmetric-Shape</td>
<td>Other shape with multiple axes of symmetry</td>
</tr>
<tr>
<td></td>
<td>Symmetric-Blob</td>
<td>Shape with no detected symmetry axes, but with no axes of elongation, suggesting an overall regular shape</td>
</tr>
<tr>
<td>Alignment</td>
<td>2D-Shape-AxisAligned</td>
<td>Shape whose edges are aligned with x- and y-axes</td>
</tr>
<tr>
<td></td>
<td>2D-Shape-Oblique</td>
<td>Shape whose edges are diagonal (non-axis-aligned)</td>
</tr>
<tr>
<td>Relative Size</td>
<td>areaTiny</td>
<td>Tiny closed shape (based on distribution of sizes in a context)</td>
</tr>
<tr>
<td></td>
<td>areaSmall/areaMedium/areaLarge</td>
<td>&quot;</td>
</tr>
<tr>
<td>Relative</td>
<td>edgeComplexityVeryLow</td>
<td>Edge-cycle with very few edges (based on distribution of complexities in a context)</td>
</tr>
<tr>
<td>Complexity</td>
<td>edgeComplexityLow/edgeComplexityMedium/edgeComplexityHigh</td>
<td>&quot;</td>
</tr>
<tr>
<td>Major Axis</td>
<td>2D-Shape-VerticalMajorAxis</td>
<td>Shape whose major axis is vertical</td>
</tr>
<tr>
<td>Orientation</td>
<td>2D-Shape-HorizontalMajorAxis</td>
<td>Shape whose major axis is horizontal</td>
</tr>
<tr>
<td></td>
<td>2D-Shape-ObliqueMajorAxis</td>
<td>Shape whose major axis is oblique</td>
</tr>
</tbody>
</table>
### 10.3 Edge-cycle Relations

Edge-cycles are related to one another using relations from the following catalog.

<table>
<thead>
<tr>
<th>Type</th>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connectivity</strong></td>
<td>(cyclesShareEdge ?c1 ?c2)</td>
<td>The cycles share an edge</td>
</tr>
<tr>
<td></td>
<td>(cyclesShareJunction ?c1 ?c2)</td>
<td>The cycles share a junction</td>
</tr>
<tr>
<td></td>
<td>(cornerAdjacent-Dominates-TJunction ?c1 ?c2)</td>
<td>There exists a T-junction where ?c1 is the dominant cycle (along the crossbar of the T) and ?c2 is one of the smaller cycles in the T.</td>
</tr>
<tr>
<td></td>
<td>(cornerAdjacent-Codominated-TJunction ?c1 ?c2)</td>
<td>There exists a T-junction where ?c1 and ?c2 are the smaller corners in the T.</td>
</tr>
<tr>
<td></td>
<td>(cornerAdjacent-Dominates-ArrowJunction ?c1 ?c2)</td>
<td>There exists an arrow-junction where ?c1 supplies the dominant cycle (along the front of the arrow) and ?c2 supplies one of the smaller corners in the arrow.</td>
</tr>
<tr>
<td></td>
<td>(cornerAdjacent-Codominated-ArrowJunction ?c1 ?c2)</td>
<td>There exists an arrow-junction where ?c1 and ?c2 are the smaller corners in the arrow.</td>
</tr>
<tr>
<td></td>
<td>(cornerAdjacent-YJunction ?c1 ?c2)</td>
<td>There exists a Y-junction that contains corners from ?c1 and ?c2</td>
</tr>
<tr>
<td></td>
<td>(cornerAdjacent-Dominates-LJunction ?c1 ?c2)</td>
<td>There exists an L-junction where the larger corner comes from ?c1 and the smaller comes from ?c2</td>
</tr>
<tr>
<td><strong>Positional</strong></td>
<td>(above ?c1 ?c2)</td>
<td>?c1 is above ?c2</td>
</tr>
<tr>
<td></td>
<td>(rightOf ?c1 ?c2)</td>
<td>?c1 is to the right of ?c2</td>
</tr>
<tr>
<td><strong>Relations to parent objects</strong></td>
<td>(objectContains ?c ?o)</td>
<td>The edge-cycle ?c contains the object ?o. ?o is either an edge-connected object or a hole in a texture region</td>
</tr>
<tr>
<td></td>
<td>(hasEdgeCycle ?eco ?c)</td>
<td>Edge-cycle ?c is an edge-cycle in the edge-connected object ?eco</td>
</tr>
<tr>
<td></td>
<td>(hasPerimeterEdgeCycle ?o ?c)</td>
<td>Edge-cycle ?c is the perimeter edge-cycle for ?o, which is an edge-connected object, or a hole in a texture region</td>
</tr>
<tr>
<td></td>
<td>(hasHole ?r ?h)</td>
<td>?h is a hole in the texture region ?r</td>
</tr>
</tbody>
</table>
## 10.4 Edge Attributes

Edges are assigned attributes from the following catalog.

<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Curvature</strong></td>
<td>StraightEdge</td>
<td>Any straight edge</td>
</tr>
<tr>
<td></td>
<td>CurvedEdge</td>
<td>Any curved edge</td>
</tr>
<tr>
<td></td>
<td>EllipseEdge</td>
<td>Any curved edge whose endpoints connect</td>
</tr>
<tr>
<td></td>
<td>MajorArcEdge/MinorArcEdge</td>
<td>A major/minor arc</td>
</tr>
<tr>
<td></td>
<td>SemicircleEdge</td>
<td>A semicircle</td>
</tr>
<tr>
<td><strong>Convexity</strong></td>
<td>ConvexEdge</td>
<td>A curved edge that forms a convexity in the parent cycle</td>
</tr>
<tr>
<td></td>
<td>ConcaveEdge</td>
<td>A curved edge that forms a concavity in the parent cycle</td>
</tr>
<tr>
<td><strong>Orientation</strong></td>
<td>VerticalEdge/HorizontalEdge/ObliqueEdge</td>
<td>A straight edge that is vertical/horizontal/diagonal</td>
</tr>
<tr>
<td></td>
<td>CurvedEdge-UpBumped, CurvedEdge-DownBumped, CurvedEdge-LeftBumped, CurvedEdge-RightBumped</td>
<td>A curved edge whose apex points upward/downward/left/right</td>
</tr>
<tr>
<td><strong>Relative Size</strong></td>
<td>lengthTiny</td>
<td>Tiny edge (based on distribution of sizes in a context)</td>
</tr>
<tr>
<td></td>
<td>lengthShort/lengthMedium/lengthLong</td>
<td>&quot;</td>
</tr>
<tr>
<td><strong>Radius function behavior (Medial Axis edges)</strong></td>
<td>Directed/Undirected</td>
<td>A skeleton edge along which the radius function is relatively stable / narrowing</td>
</tr>
<tr>
<td></td>
<td>Concave/Linear/Convex</td>
<td>A directed skeleton edge along which the second derivative of the radius function is ( &gt; /=/ &lt; 0 ).</td>
</tr>
<tr>
<td></td>
<td>Obtuse/Right/Acute</td>
<td>A directed skeleton edge along which the first derivative of the radius function is ( &gt; /=/ &lt; -1 ).</td>
</tr>
<tr>
<td><strong>Source/Sink</strong></td>
<td>Source/Sink</td>
<td>An undirected skeleton edge whose directed neighbors are exclusively directed away from / toward it</td>
</tr>
<tr>
<td></td>
<td>matRadiusVeryNarrow</td>
<td>A skeleton edge with a very narrow radius function relative to the distribution of edges in the medial axis</td>
</tr>
<tr>
<td></td>
<td>matRadiusNarrow, matRadiusMedium, matRadiusThick</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
## 10.5 Edge Relations

Edges are related to one another using relations from the following catalog.

<table>
<thead>
<tr>
<th>Type</th>
<th>Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Relative Orientation/Position</strong></td>
<td>(parallelElements ?e1 ?e2)</td>
<td>The edges are parallel</td>
</tr>
<tr>
<td></td>
<td>(perpendicularElements ?e1 ?e2)</td>
<td>The edges are perpendicular</td>
</tr>
<tr>
<td></td>
<td>(collinearElements ?e1 ?e2)</td>
<td>The edges are collinear</td>
</tr>
<tr>
<td><strong>Corners</strong></td>
<td>(convexAngleBetweenEdges ?e1 ?e2)</td>
<td>The corner going from ?e1 to ?e2 in the cycle is convex wrt the cycle</td>
</tr>
<tr>
<td></td>
<td>(concaveAngleBetweenEdges ?e1 ?e2)</td>
<td>The corner going from ?e1 to ?e2 in the cycle is concave wrt the cycle</td>
</tr>
<tr>
<td></td>
<td>(acuteCorner (convexAngleBetweenEdges ?e1 ?e2))</td>
<td>The corner going from ?e1 to ?e2 in the cycle forms an acute angle</td>
</tr>
<tr>
<td></td>
<td>(obtuseCorner (convexAngleBetweenEdges ?e1 ?e2))</td>
<td>The corner going from ?e1 to ?e2 in the cycle forms an obtuse angle</td>
</tr>
<tr>
<td></td>
<td>(perpendicularCorner (convexAngleBetweenEdges ?e1 ?e2))</td>
<td>The corner going from ?e1 to ?e2 in the cycle forms a right angle</td>
</tr>
<tr>
<td><strong>Corner Relations</strong></td>
<td>(cycleAdjacentAngles (convexAngleBetweenEdges ?e1 ?e2) (concaveAngleBetweenEdges ?e3 ?e4))</td>
<td>The corner going from ?e3 to ?e4 follows the corner going from ?e1 to ?e2 in the cycle</td>
</tr>
<tr>
<td><strong>Radius function behavior (Medial Axis edges)</strong></td>
<td>(weaklyDirectedConnection (directedConnection ?e1 ?e2))</td>
<td>Either ?e1 is directed toward ?e2, which is undirected, or ?e2 is directed away from ?e1, which is undirected. ?e1 and ?e2 must be connected</td>
</tr>
<tr>
<td></td>
<td>(weaklyDirectedConnection (directedConnection ?e1 ?e2))</td>
<td>?e1 is directed towards ?e2, which is directed in the same direction. The two edges are connected.</td>
</tr>
<tr>
<td></td>
<td>(sourceConnection (elementsConnected ?e1 ?e2))</td>
<td>?e1 and ?e2 are connected, and directed away from one another</td>
</tr>
<tr>
<td></td>
<td>(sinkConnection (elementsConnected ?e1 ?e2))</td>
<td>?e1 and ?e2 are connected, and directed toward one another</td>
</tr>
<tr>
<td></td>
<td>(clockwiseNeighborAtJunction ?e1 ?e2)</td>
<td>There exists a junction in which, going around it in the clockwise direction, ?e2 follows ?e1</td>
</tr>
</tbody>
</table>