SKETCH-BASED SERVICES

Visual-Conceptual Relationship Suggestions in Natural Language

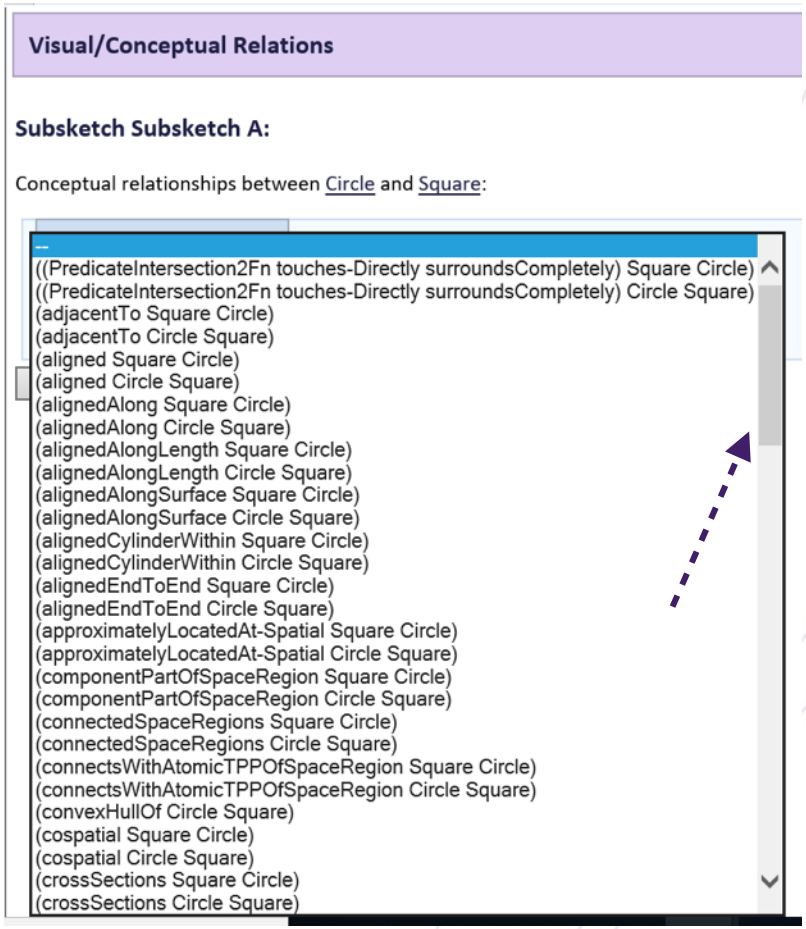
Victor S. Bursztyn

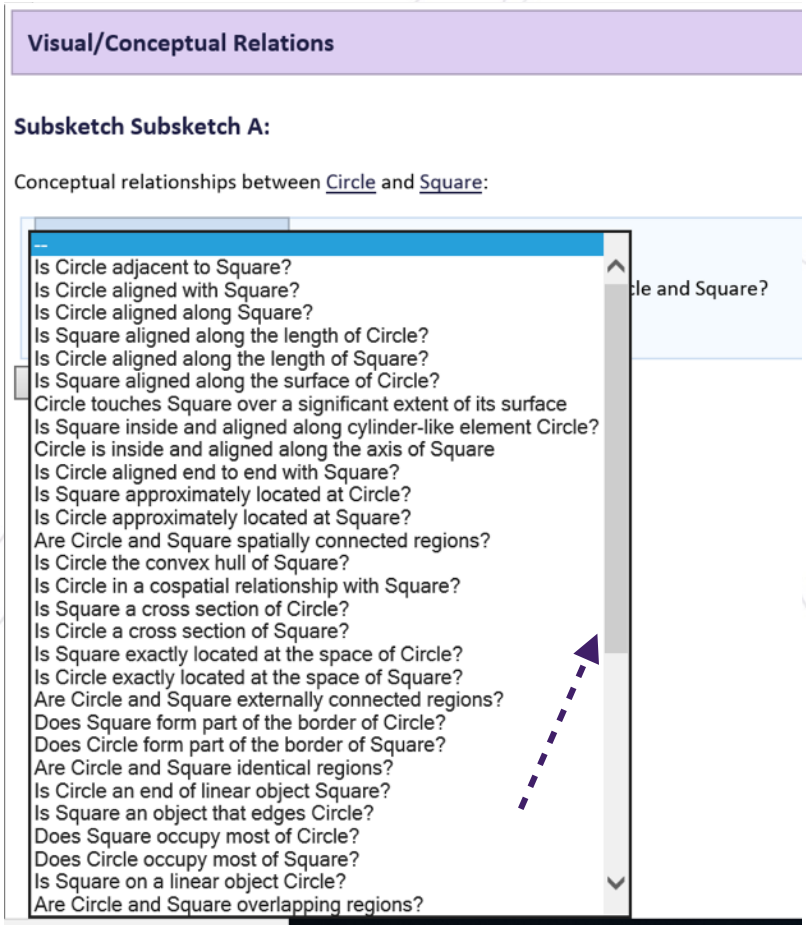
Northwestern | McCORMICK SCHOOL OF ENGINEERING

Key Results

1. **VCR suggestions integrated with Verbalize for `insideInSketch` (17 relations) and `atOrOverlapsInSketch` (43 relations).**
2. **Additional integration with Multimodal Interpretation Group (12 relations).**
3. **Absence of `genFormat` returns `nil`, serving as a filter for suggestions that can't be clearly/concisely explained.**
4. **Symmetric relations are handled in order to filter redundant suggestions (e.g., "Is A adjacent to B?" appearing once).**

Original vs. Current Results





Micro-Theory `CogSketchNLGMt`

```
(in-microtheory CogSketchNLGMt)
(genFormat in-ImmersedFully "Is ~A fully immersed in ~A?" (TheList 1 2))
(genFormat inRegion "Is ~A in region ~A?" (TheList 1 2))
(genFormat constituents "Is ~A a constituent of ~A?" (TheList 1 2))
(genFormat cospatial "Is ~A in a cospatial relationship with ~A?" (TheList 1 2))
(genFormat externalParts "Is ~A an external part of ~A?" (TheList 2 1))
(genFormat hasStoredInside "Is ~A stored inside of ~A?" (TheList 1 2))
(genFormat in-Among "Is ~A among ~A?" (TheList 1 2))
(genFormat in-EmbeddedInFluidMatrix "Is ~A embedded in fluid matrix in ~A?" (TheList 1 2))
(genFormat in-PassesThrough "Does ~A pass through ~A?" (TheList 1 2))
(genFormat internalParts "Is ~A an internal part of ~A?" (TheList 1 2))
(genFormat internalSubRegions "Is ~A an internal subregion of ~A?" (TheList 1 2))
(genFormat mainConstituent "Is ~A the main constituent of ~A?" (TheList 1 2))
(genFormat mainFunctionalComponent "Is ~A the main functional constituent of ~A?" (TheList 1 2))
(genFormat objectFoundInLocation "Is ~A an object found in location of ~A?" (TheList 1 2))
(genFormat physicalParts "Is ~A a physical part of ~A?" (TheList 1 2))
(genFormat physicallyContains "Is ~A physically contained in ~A?" (TheList 1 2))
(genFormat primaryConstituent "Is ~A a primary constituent of ~A?" (TheList 1 2))
(genFormat cordEnd "Is ~A at the end of cordlike object ~A?" (TheList 1 2))
(genFormat connectedTo "Is ~A connected to ~A?" (TheList 1 2))
(genFormat connectedAtEnd "Is ~A connected at the end of ~A?" (TheList 1 2))
(genFormat connectedAtContact "Is ~A connected to ~A at a point of contact?" (TheList 1 2))
(genFormat connectedAtSpot "Is ~A connected to ~A at a localized spot?" (TheList 1 2))
(genFormat connectedAlongSurface "Is ~A connected along the surface of ~A?" (TheList 1 2))
(genFormat surfaceParts "Is ~A a surface part of ~A?" (TheList 1 2))
(genFormat surrounds-3D "Does ~A surround ~A along all the three dimensions?" (TheList 1 2))
(genFormat coversPartially "Does ~A partially cover ~A?" (TheList 1 2))
(genFormat alignedCylinderWithin "Is ~A inside and aligned along cylinder-like element ~A?" (TheList 1 2))
(genFormat spiralsAround "Does ~A spiral around ~A?" (TheList 1 2))
(genFormat alignedAlongLength "Is ~A aligned along the length of ~A?" (TheList 1 2))
(genFormat adjacentTo "Is ~A adjacent to ~A?" (TheList 1 2))
(genFormat aligned "Is ~A aligned with ~A?" (TheList 1 2))
(genFormat alignedAlong "Is ~A aligned along ~A?" (TheList 1 2))
(genFormat alignedAlongSurface "Is ~A aligned along the surface of ~A?" (TheList 1 2))
(genFormat alignedEndToEnd "Is ~A aligned end to end with ~A?" (TheList 1 2))
(genFormat approximatelyLocatedAt-Spatial "Is ~A approximately located at ~A?" (TheList 1 2))
(genFormat whollyLocatedAt-Spatial "Is ~A entirely located in the space of ~A?" (TheList 1 2))
(genFormat exactlyLocatedAt-Spatial "Is ~A exactly located at the space of ~A?" (TheList 1 2))
(genFormat partiallyLocatedAt-Spatial "Is ~A partially located in the space of ~A?" (TheList 1 2))
(genFormat connectedSpaceRegions "Are ~A and ~A spatially connected regions?" (TheList 1 2))
(genFormat convexHullOf "Is ~A the convex hull of ~A?" (TheList 1 2))
(genFormat crossSections "Is ~A a cross section of ~A?" (TheList 1 2))
(genFormat externallyConnectedSpaceRegions "Are ~A and ~A externally connected regions?" (TheList 1 2))
(genFormat formsBorderPart "Does ~A form part of the border of ~A?" (TheList 1 2))
(genFormat identicalSpaceRegions "Are ~A and ~A identical regions?" (TheList 1 2))
(genFormat linearObjectEnds "Is ~A an end of linear object ~A?" (TheList 1 2))
(genFormat objectEdges "Is ~A an object that edges ~A?" (TheList 1 2))
(genFormat occupiesMostOf "Does ~A occupy most of ~A?" (TheList 1 2))
(genFormat onLine "Is ~A on a linear object ~A?" (TheList 1 2))
(genFormat overlappingSpaceRegions "Are ~A and ~A overlapping regions?" (TheList 1 2))
(genFormat partOfSpaceRegion "Is ~A part of region ~A?" (TheList 1 2))
(genFormat partiallyOverlappingSpaceRegions "Does ~A partially overlap region ~A?" (TheList 1 2))
(genFormat partiallySpatiallyIntersects "Does ~A partially intersect space of ~A?" (TheList 1 2))
(genFormat properPartOfSpaceRegion "Is ~A a proper part of region ~A?" (TheList 1 2))
(genFormat spatiallyIntersects "Does ~A spatially intersect ~A?" (TheList 1 2))
(genFormat spatiallySubsumes "Does ~A spatially subsume ~A?" (TheList 1 2))
(genFormat touches "Does ~A touch ~A?" (TheList 1 2))
(genFormat touches-Externally "Does ~A externally touch ~A?" (TheList 1 2))
(genFormat stronglyConnectedSpaceRegions "Are ~A and ~A strongly connected regions?" (TheList 1 2))
```

```
(genFormat adjacentTo
"Is ~A adjacent to ~A?"
(TheList 1 2))
```

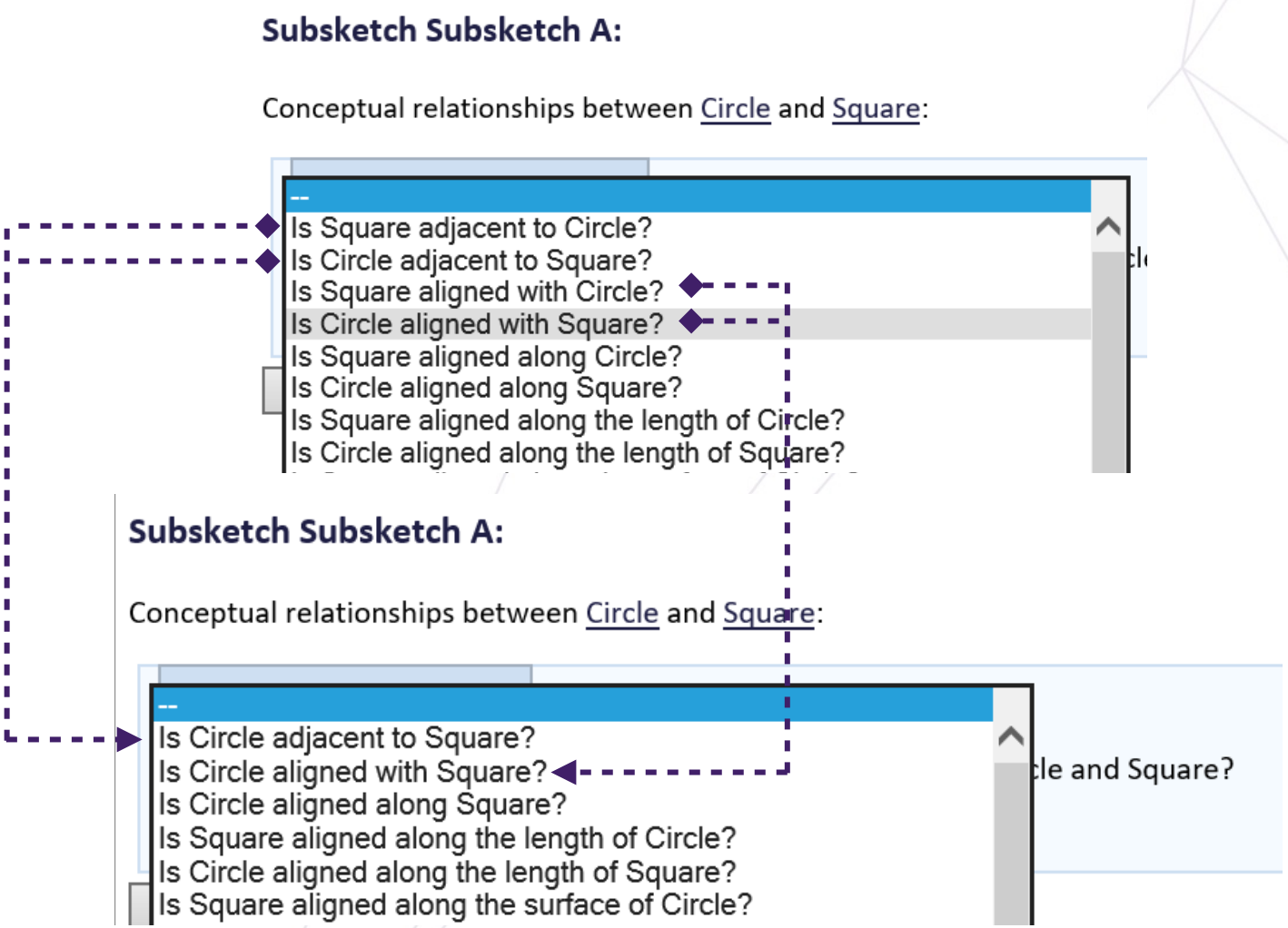
Function `print-vcr-reln-select`

```
(defun print-vcr-reln-select (names relns q-idx)

  …

 (dolist (reln relns)
  (let ((pretty-exp (fire:verbalize (sublis names (first reln))
                                    :context 'd::CogSketchNLG2Mt
                                    :fallback nil)))
        (when pretty-exp
          (net.html.generator:html
           (:newline)
           ((:option value
            (format-relation-predicate (car reln) names))
            (:princ pretty-exp)))))))))
```

Before & After Filtering



Function `filter-symmetric-relns`

```
(defun filter-symmetric-relns (possible-relns)
  (remove-duplicates
   possible-relns
   :test #'(lambda (x y)
             (let ((exp1 (first x))
                   (exp2 (first y)))
               ;; expn = (relation object1 object2)
               (and (equal (first exp1) (first exp2))
                    (equal (second exp1) (third exp2))
                    (equal (second exp2) (third exp1))
                    (kb:instance-of? (first exp1)
                                     'd::SymmetricRelation
                                     'd::EverythingPSC))))))
```

Questions & Comments:

victorbursztyn2022@u.northwestern.edu