A Theory of Depiction for Sketches of Physical Systems

Kate Lockwood, Andrew Lovett, Ken Forbus, Morteza Dehghani and Jeff Usher

Northwestern University, Qualitative Reasoning Group
2145 Sheridan Road Room L359, Evanston Illinois
{kate, andrew-lovett, forbus, morteza, usher}@northwestern.edu

Abstract

Complex spatial and physical concepts are often communicated using diagrams. For many qualitative reasoning tasks, it is necessary that computers understand diagrams in much the same way as their human collaborators. Here we describe some preliminary work on basic diagram interpretation based on common depiction conventions. Using a combination of semantic and qualitative spatial information we are able to distinguish relevant regions and edges in sketched diagrams using the CogSketch sketch understanding system.

Introduction

Complex physical concepts are often communicated using a combination of text and diagrams. Often the diagram illustrates the physical arrangement of a system and accompanying text or captions describe a process that happens in that system. For example, consider Figure 1 below taken from *Sun Up Sun Down* (Buckly, 1979) an introductory solar energy text.

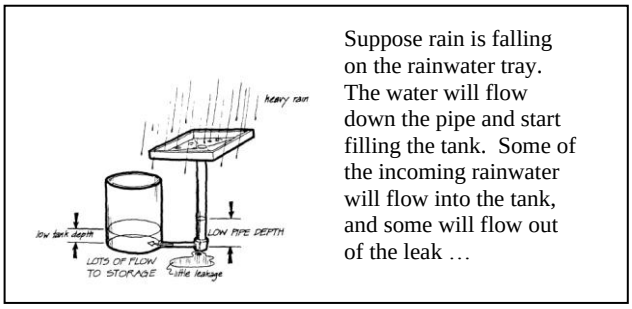

Figure 1. An example of a diagram and accompanying text from a solar energy textbook

The caption provides a description of a process, the filling of a tank with rainwater, while the diagram provides an illustration of the physical layout of the system (tank, pipe, etc).

Parsing the qualitative information in the diagram is easy for people, but quite complicated for software. For example people can both recognize the diagram as a full system and refer to its individual components like "the water in the tank". Part of this flexibility is due to our familiarity with diagrams and their depiction conventions. Another source of flexibility is our knowledge about how things like tanks and pipes and water work. We are able to leverage both types of knowledge when looking at a diagram.

To make intelligent systems that can reason and communicate using diagrams, they must understand diagrams in ways similar to their human users. Additionally, systems need to be able to understand not just polished textbook diagrams, but incomplete, messy user-sketched diagrams. For intelligent conceptual design aids and intelligent tutoring systems, being able to understand informally sketched concepts is paramount.

Traditional diagram and sketch understanding work is far from exhibiting the human-like flexibility needed in a diagram understanding system. Most sketch systems are focused on recognition. First, they segment images into lines and then combine the lines into larger objects. Objects are then matched against a library of known shapes, with the best match considered as the interpretation for the object. Such systems can work well in tightly constrained domains that have a small number of distinct symbols such as family trees (Alvarado & Davis, 2004), circuit and diagrams (Alvarado, Oltmans, & Davis, 2002), x,y plots (Futrelle, 1990) and maps (Reiter & Mackworth, 1989). Unfortunately, they do not translate well to diagrams like Figure 1.

In this paper we are proposing a more integrated approach to diagram interpretation using CogSketch, an open-domain sketch understanding system. In our model, we use a combination of semantic information about the objects and qualitative spatial relationships between them to infer relevant depiction conventions. Specifically, our goal is to correctly assign concepts to *edges* and *regions* in a diagram, consistent with depiction conventions.

The CogSketch approach is based on two insights: (1) In most human-to-human sketching, recognition is a catalyst, not a requirement. People use language to explain their sketches; we provide interface tools for providing functionally similar ways to *conceptually label* glyphs in a sketch. (2) Many of the conceptually relevant relationships in sketches are *qualitative*. For example, in the diagram in Figure 1 the specific details of the objects depicted does not matter, what matters is the qualitative relationships: how the objects are connected, what the level of the water is in the tank is relative to the placement of the leak, etc. Our approach is to model human visual and geometric processing of the ink in a sketch, combined with formal representations of conceptual knowledge drawn from a large-scale knowledge base, to provide *open-domain* sketch understanding abilities. This is very important for building intelligent systems for open-ended,

domains, such as engineering design, where the set of possible objects is extremely broad.

The rest of this paper describes our method for modeling this flexible interpretation of depiction conventions within CogSketch. First we review CogSketch. Next we describe the spatial extent problem. Then we describe how we combine semantic and geometric information to interpret a sketch, using a detailed example. We finish with related work and future work.

Sketching

All diagrams are sketched using CogSketch[1]. CogSketch is an open-domain sketch understanding system built on the nuSketch architecture (Forbus, Ferguson, & Usher, 2001). In CogSketch, each object drawn is represented by a *glyph*. A glyph contains both the actual ink drawn by the user and a *conceptual label.* The conceptual label is supplied by the user and is tied to a concept in the underlying knowledge base. Currently we are using a subset of the ResearchCyc[2] knowledge base (including over 30,000 concepts). Users can also supply a name with which to refer to the glyph. Names can be any natural language string. For example, Figure 2 shows a screenshot of a diagram drawn in CogSketch. In this diagram, the cylinder in the sketch is labeled as a `WaterTank` using the concept from ResearchCyc and is named "tank". This allows the user to refer to the tank simply as "tank". Likewise, if there were multiple tanks, they could each be given different identifying names. In CogSketch, users determine what ink belongs to a glyph by clicking a button at the beginning and end of drawing each glyph. All the ink drawn between button presses is part of the glyph.

Conceptual labeling allows CogSketch to truly be domain-independent and allows us to operate in domains without clear drawing conventions. All sketch understanding work must strike a balance between constraints on the user and the depth of interpretation that is possible. While labeling glyphs does require more work by the user, in return they gain freedom from recognition errors and the ability to be supported by more in-depth reasoning. Aside from manual segmentation, we place no other restrictions on how users draw each glyph. For example, they can use as many strokes as they like, connected or not, and can take as long as they like. This contrasts with a common practice in multimodal interfaces of using constraints such as time-outs and pen-up events to automatically infer segmentation. For our users, who are often thinking hard about what they are drawing, time-outs and pen-up constraints are poor segmentation signals and quite annoying to them.

CogSketch computes a variety of spatial relationships automatically, including the RCC-8 qualitative topology (Cohn, 1996) relationships and connected and contained groups of glyphs (see (Forbus, Tomai, & Usher, 2003) for details). The digital ink itself is also available in subsequent processing, re-sampled into constant-spaced intervals from the original time-stamped pen events.

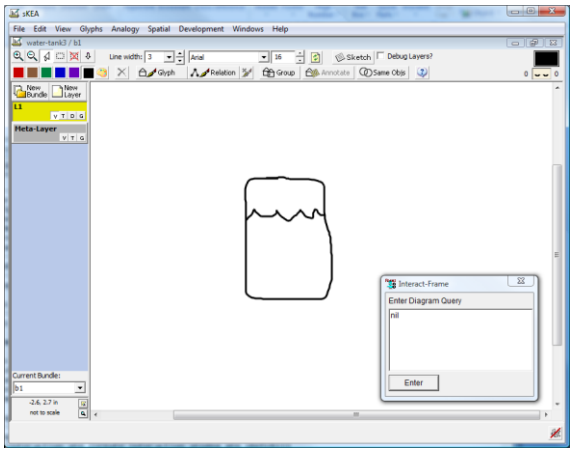

Figure 2. A screen shot of CogSketch showing a sketch of a tank of water.

Conceptual Segmentation

We define the task of *conceptual segmentation* to be the assignment of conceptual interpretations to regions and edges within the sketch. As noted above, conceptual labeling of ink is necessary, but not sufficient, for solving this problem. Consider the sketch in Figure 2 above showing a tank filled with water. We will use this example as an illustration throughout this paper. This sketch consists of two glyphs: one closed polygon representing the tank, and one line representing the water. Figure 3 illustrates.

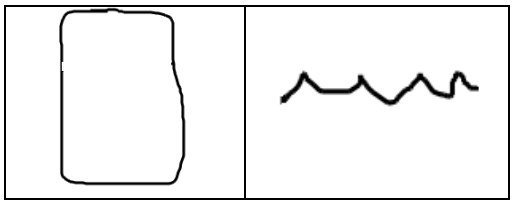

Figure 3. The two glyphs that make up the sketch in Figure 2. The glyph on the left is the tank and the glyph on the right is water.

If we simply use the conceptual labels, the system would think that the object water in the sketch was the edge created by the water glyph when in fact it is the area

1 CogSketch is publicly available at http://spatiallearning.org/projects/cogsketch_index.html
2 http://research.cyc.com/

inside the tank underneath the water glyph. The situation gets even more complex in a sketch like that in Figure 4 below. Here again, the water glyph is a single line, but this line is discontinuous and spans to different tank glyphs. The system would also need to infer that the pipe (another individual glyph) is also filled with water even though the pipe glyph does not touch the glyph representing the water.

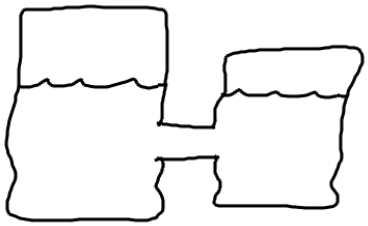

Figure 4. A two tank system as sketched in CogSketch

One way to address this would be to require users to draw following specific conventions – for example, have them trace around the inside of all of the tank/pipe glyphs so that the water glyph was one continuous closed shape. However, while we could institute that constraint, it only addresses this specific situation, and adding new constraints to address every new situation is untenable. Additionally, requiring users to trace the full outline of the water still leaves the situation ambiguous. The system still doesn't have a way to figure out if the user intended just the outline to represent water, or all of the space contained by the outline. For example, consider the two sketches in Figure 5 below.

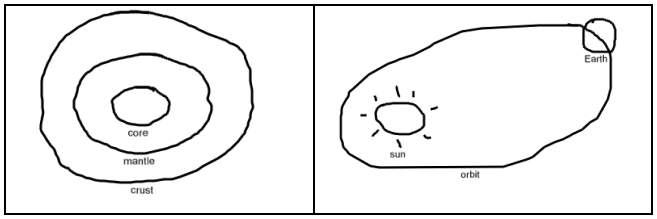

Figure 5. Two sketches, one of the layers of the Earth and another of a planet orbiting the sun.

Both sketches contain an outer ellipse. In the sketch on the left it represents the crust of the earth, and in the sketch on the right it represents the orbit of Earth around the sun. The interpretation for the two ellipses is different. In the sketch on the left the convention is that everything between the outer ellipse and the next ellipse is the stuff that makes up the crust. By contrast, in the sketch on the right, the orbit is actually just the edge represented by the glyph itself. This is why we need a combination of semantic and geometric information in order to make a correct interpretation. There are two parts to our interpretation process - the gathering of semantic information and the segmentation of the image.

We are interested in using the fact that we know what we are drawing and we know about how things are typically drawn – *depiction conventions* – to automatically derive the correct conceptual segmentation of the sketch. We test its segmentations by asking it to highlight the region or edge in a sketch representing a specific entity. If the correct area is highlighted, we conclude that the system has correctly interpreted that portion of the sketch.

Using semantic information for depiction reasoning

Once the appropriate glyph is identified, we access the conceptual label(s) provided by the user. In our example, the glyph being considered is labeled with the concept `Water` from the ResearchCyc KB. Knowing what the glyph represents helps us figure out how to interpret the diagram correctly. For example, ResearchCyc has 335 facts about water. This includes information about its role in the ResearchCyc ontology and, especially important for our purposes, some linguistic knowledge about the term.

Backchaining rules are used to ascertain whether a concept needs a region versus a polyline to depict it. For example, a concept might contain information that, linguistically, the word referring to it is a mass noun or a count noun. Mass nouns refer to entities that can be viewed as spatially flexible pieces of stuff, such as liquids and powders, whose boundaries are highly constrained by containment relationships. The concept `Water` is linguistically a mass noun, and consequently the system infers that it requires a volume to depict it.

Figure 6 below shows an outline of the process we use to determine the correct depiction for a glyph using both the conceptual label and the ink. This figure shows the algorithm as it is currently implemented, as we expand the number and type of diagrams that we interpret, the algorithm will be further refined. In the first step, the conceptual label is accessed and the knowledge base is queried to determine which category the entity belongs to: (1) a mass noun or entity that subclasses from the Cyc concept `TangibleStuffCompositionType` (2) an entity that subclasses from `Path-Spatial` (3) or a physical object.

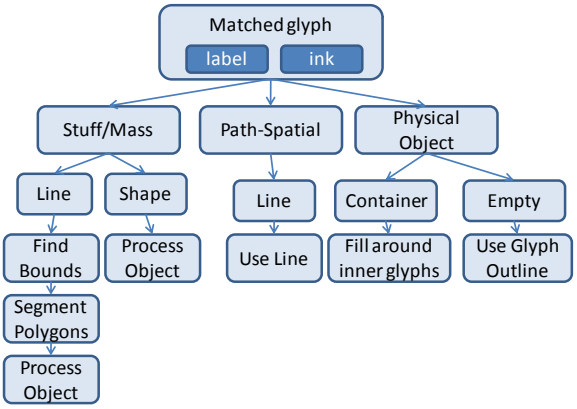

Figure 6. Outline of the spatial extent identification algorithm as it is currently implemented

Inferring the geometry of depiction

Once the system has inferred the conceptual category for a glyph, it attempts to find or construct the appropriate geometric entity. For the water/tank example (an instance of the stuff/mass path through Figure 6) it starts by classifying the geometric properties of the ink for the glyph, determining if it is a line or a region. For example, the glyph representing water in Figure 5 is a polyline, not a region. Since the depiction of water requires a region, the system has more work to do. If the user had drawn the water by tracing out a region inside the tank, then the system would be satisfied with the glyph itself as the geometric entity.

The next step is to determine if there are other glyphs which can help constrain the extent of the object. In this example, the tank glyph constrains the extent. We find such glyphs by looking for RCC8 relationships, i.e., glyphs for which the water is either TPP or NTPP (i.e., Tangential Proper Part or Non-Tangential Proper Part). When these relationships hold, between the tank glyph and the water glyph, we then do a follow-up check to see if the water intersects (within a threshold) both sides of the tank.

Once we have both glyphs (the water and the tank) we need to find the region representing the part of the tank where the water is found. This is accomplished by combining the ink from the two glyphs and segmenting the ink into edges and *edge cycles*. Edges are identified by segmenting the ink at places where one line intersects another, or where there is a clear corner along a line. Edge cycles are identified by finding minimal closed cycles among the edges. In the current example, CogSketch identifies two edge cycles, one representing the area in the tank above the water and the other representing the area in the tank below the water.

For stuff/mass nouns, the system assumes the user has drawn the uppermost edge of the object, and that the object descends from there to fill the container below it. Thus, in the current example, the system looks for a cycle such that glyph for water overlaps with the top of the cycle, while the rest of the cycle is made up of points from the tank glyph. If an appropriate cycle is found, it is identified as the region that the user is looking for, and it is then converted to a polygon and processed like a physical object.

Physical objects (the third path in Figure 6) are checked to see if they contain other glyphs (containment is one of the spatial relationships computed automatically by CogSketch). If the glyph has other objects inside of it, the algorithm as currently implemented assumes that the correct segmentation for the glyph is the space around the inner objects. This is the correct interpretation for situations like the layers of the earth, or bubbles in soda. Figure 7 shows the results of the query "mantle" in a sketch of the layers of the Earth.

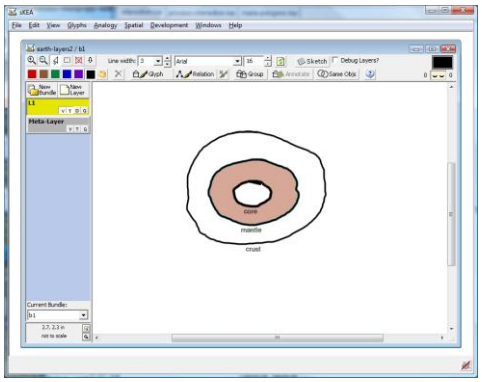

Figure 7. The results from the user query "mantle" in a sketch of the layers of the Earth.

If a physical object has no interior glyphs, the whole area of the glyph is considered the correct depiction and it is highlighted in the diagram. Figure 8 shows the results of our system on the water and tank example when queried for "water".

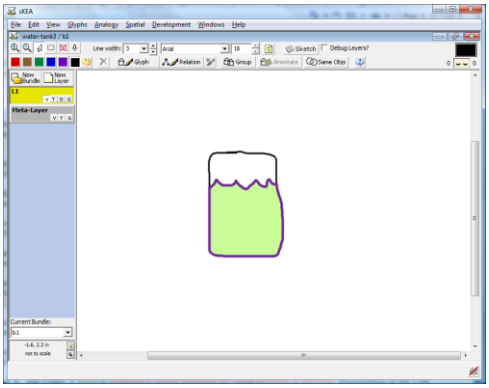

Figure 8. A screen shot of CogSketch showing the results from the user query "water". The shaded are represents the region that the system infers is water.

This approach easily extends to other, more complex situations. In Figure 9 the sketch is composed of four glyphs: tank1 (the tank on the left), a pipe, tank2 (the tank on the right), and one glyph representing the water. Since our algorithm for locating cycles of edges is flexible enough to find cycles over multiple glyphs, the two tank problem is easily handled.

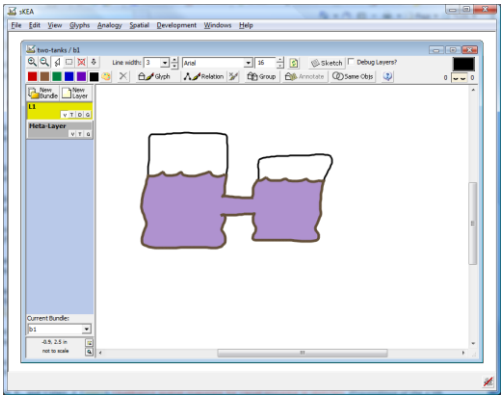

Figure 9. Another result for the query "water". In this case the sketch contains four separate glyphs {tank1, tank2, pipe, water}.

We are also able to handle situations where there are several glyphs that are conceptually labeled as mass nouns, even if they are drawn similarly. In Figure 10 the sketch is a tank with both oil and water in it. When queried for "oil" our system is able to easily identify the extent of the area representing oil. Situations like this would be particularly tricky for template based systems since both oil and water are drawn with similar glyphs. Also, while the wavy line is typical of a convention used to indicate liquid in a sketch, it is by no means a standardized symbol.

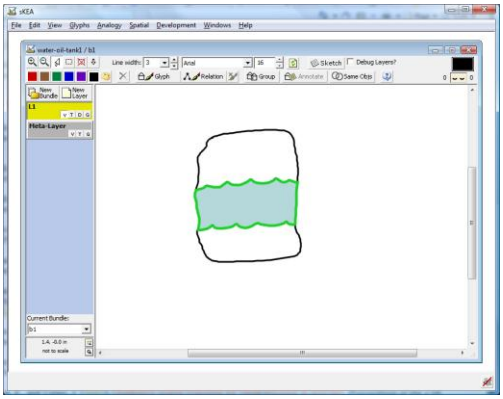

Figure 10. In this example, the system is able to easily discriminate between the region representing "oil" and that representing "water" using the same techniques.

The current algorithm for physical objects has been sufficient for all of the diagrams that we have considered in this paper, however, when a glyph is a container it isn't always the case that you want just the space around the interior glyphs. For example, consider a glass of water with a straw in it. When you are determining the spatial extent of the water, it actually also covers the area occupied by the straw. We are extending the spatial extent algorithm to account for situations like this by further examining the objects in container/contained groups. This is another example where we will need to combine conceptual information from the KB with spatial information from the ink to identify the correct spatial extent.

The processing for an entity that has been determined to be an instance of a Path-Spatial proceeds much like the processing of a mass noun, by first checking to see how the object is drawn in the sketch. Here we will refer back to the solar system/orbit example from Figure 5. In this case, the system checks to see if the path is represented by a single line, like the orbit in the sketch. This suggests that the points on the line make up the conceptual entity. The other option, of course, is that a path is depicted by multiple lines or polygons such as a drawing of a railroad track or road. This condition is not currently being handled by our system, but is in the process of being added.

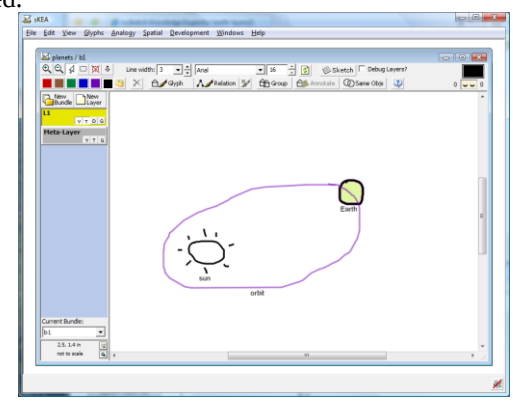

Figure 11. A screenshot showing the spatial extent identified for "orbit" and "Earth" in a simplified drawing of the solar system. Even though both objects are drawn similarly, conceptual information provides clues as to their different interpretations.

Compound Queries

Often the parts of a diagram that need to be referred to are more complex than just "water". For example, when doing problems in physics or chemistry, it may be useful to be able to refer to the water in one part of the apparatus only. Our system also handles queries of the form *<object> <relation> <object>*. Information about

relations from ResearchCyc is used to understand the semantics of such queries. Figure 12 illustrates the result for the query "water in tank1". The analysis is essentially that of Figure 9 with the additional specification of "in tank1" leading to the intersection of the water polygon and the tank1 polygon.

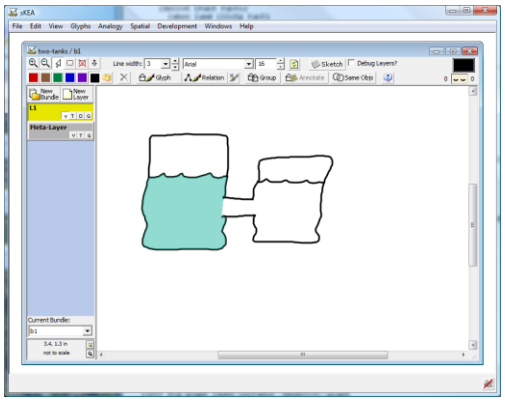

Figure 12. Screenshot showing the result of the query "water in tank1"

Related Work

The division of scene elements into edges and regions in sketches was explored in the Mapsee program of Reiter and Mackworth (Reiter & Mackworth, 1989). They proposed a logical framework for depiction that formalized the mapping between images and scenes of simple maps containing roads, rivers, shores (represented as edges in the images) and water and land (represented by regions in the images). They identified a set of six visual relations ({tee, chi, bounds, closed, interior, and exterior}) and provided axioms and constraints which combined these visual primitives and mapped them to the scene elements (roads, rivers, etc). Like Mapsee, we are concerned with modeling how conceptual entities are depicted. However, Mapsee was designed for one domain, i.e., maps, and its axioms map visual elements directly to interpretations in that domain. By contrast, our model works through an intermediate distinction – regions versus edges – and performs reasoning over a large-scale, off-the-shelf knowledge base to identify depiction constraints. Their task was fundamentally one of image interpretation, recognizing unlabelled lines as map elements, whereas our task starts with conceptually labeled ink.

Alvarado and colleagues (Alvarado & Davis, 2004; Alvarado, Oltmans, & Davis, 2002) describe a multi-domain sketch recognition engine. Their systems use a hierarchical shape description language where low level shape description (circles, arrows, etc) are defined once in a domain-independent fashion. Then a separate set of rules ties a given shape to a domain specific interpretation (e.g. an arrow represents a child link in a family tree diagram). This approach can work well in a very tightly constrained domain with a small number of differentiated symbols (family trees, circuit diagrams, etc.) Unfortunately, it does scale to the more open-domain, unconstrained types of sketches that we are concerned with.

There could be advantages to incorporating some carefully restricted low-level shape recognition to our depiction reasoning, to identify common elements (e.g., arrows). For example, in a physics system, it might be useful to automatically recognize arrows and interpret them as forces while leaving the types of objects that those forces can act on unconstrained given the wide variety of physical objects in the world.

We believe that recognition is not very important for the sketch understanding tasks we are focused on. Unlike sketches in engineering design, where later versions will need to be imported to a formal CAD system, the sketches produced for student assessments are meant to be short lived. Also, while the amount of detail can vary greatly, much of it is superfluous to the pedagogical goals of the assignment and is not important for the overall interpretation of student understanding.

Conclusions and Future Work

We have described how to use a combination of semantic and geometric information to identify one type of depiction convention in sketched diagrams. Our interpretation process closely couples semantic and geometric information to reason about depiction conventions and to use those conventions to segment the sketch into meaningful regions and edges.

Our work on depiction conventions is motivated by several projects. Creating a platform for sketch-enabled educational software is one of our long-range goals. Another is the use of sketches in multimodal knowledge capture. For example, diagrams in educational materials are accompanied by explanatory text. We are creating a system that learns from sketched diagrams plus accompanying simplified English text. Being able to correctly interpret how entities in the diagram are depicted is essential for integrating knowledge across modalities.

We are also interested in studying depiction conventions which are widely used, but not domain or situation dependent. For example, call-outs and cut-aways are two conventions that are used across disciplines which have important implications for how diagrams (and the spatial relations in them) should be interpreted.

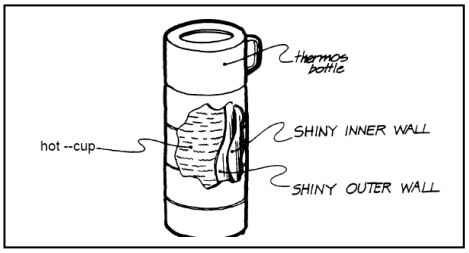

Figure 13. Example of a cut-away in a diagram

CogSketch is free and available online (the online version comes bundled with OpenCyc, as opposed to ResearchCyc which was used for this work). As more people download and use CogSketch, we are hoping to amass a large library of sketches. This library will enable us to more thoroughly survey the conventions used in sketched diagrams. It will also provide a corpus of labeled sketches that we hope will be useful to us and to others in the sketch understanding community.

Acknowledgements

This work was supported by a grant from the Office of Naval Research and by the National Science Foundation under Grant No. SBE-0541957, The Spatial Intelligence and Learning Center.

References

Alvarado, C., Davis, R. (2004). Sketchread: a multi-domain sketch recognition engine. *Proceedings of the 17th annual acm symposium on user interface software and technology*.

Alvarado, C., Oltmans, M., Davis, R. (2002). A framework for multi-domain sketch recognition. *Proceedings of aaai spring symposium on sketch understanding*.

Buckley, S. (1979). *Sun Up to Sun Down*. New York: McGraw Hill.

Cohn, A. (1996) Calculi for Qualitative Spatial Reasoning. In *Artificial Intelligence and Symbolic Mathematical Computation*, LNCS 1138, eds: J Calmet, J A Campbell, J Pfalzgraph, Springer Verlag, 124-143.

Forbus, K., Ferguson, R., & Usher, J. (2001). Towards a computational model of sketching. *IUI'01*. January 14-17, 2001. Santa Fe, New Mexico.

Forbus, K., Tomai, E., and Usher, J. (2003). Qualitative spatial reasoning for visual grouping in sketches. *Proceedings of the 17th International Workshop on Qualitative Reasoning*, Brasilia, Brazil, August.

Futrelle, R. P. (1990). Strategies for Diagram Understanding: Object/Spatial Data Structures, Animate Vision and Generalized Equivalence. In *10th ICPR* (pp. 403-408): IEEE Press.

Reiter, R. and Mackworth, A.K. (1989). A Logical Framework for Depiction and Image Interpretation. *Artificial Intelligence*, 41, 125-155.