Toward Intelligent Drawing Constraints

Ronald W. Ferguson, Neil Cutshaw, and Huzaifa Zafar

Visual Thinking Group, College of Computing, Georgia Institute of Technology
801 Atlantic Drive, Atlanta, GA 30332 USA
{rwf, amator, huzaifaz}@cc.gatech.edu

Abstract

Diagrammatic problem solving in domains such as architecture and design often involves continual redrawing of existing figures. Drawing programs, such as Visio and CorelDraw, accommodate redrawing using secondary correction techniques. These techniques act when a visual element is moved or changed, making additional changes to the drawing that maintain key preexisting visual relationships. These techniques include low-level constraints, such as snap and glue, and domain-specific glyphs with built-in constraints. As useful as these secondary correction techniques are, they sometimes operate in unintuitive ways, interfering with the flow of problem-solving. Here, we describe our plans to increase the effectiveness of secondary correction by making such constraints congruent with the place vocabulary of the domain in question. We introduce the idea of *place vocabulary constraints* (PVCs) that translate a particular place vocabulary into a set of geometric constraints. We argue that PVCs may provide a better mechanism for secondary correction during redrawing because multiple geometric constraints can be handled by a single place vocabulary constraint, because place vocabulary constraints are more congruent with users' expectations, and because they provide a better mechanism for explaining such constraints to the user.

Introduction

As part of our research on diagrammatic reasoning systems that dynamically interact with human problem solvers, we are studying how drawing systems may conserve particular visual relations during redrawing. Problem solving with diagrams often require continual redrawing, and so it is useful to think about the ways in which redrawing aids or inhibits problem solving. One clear problem is that while the changes the user is attempting to make to the diagram are often based on the problem domain, he or she must often make the changes at the purely geometric level, which can introduce problem-solving disfluencies.

One way to get around this difficulty may be to assume that all modifications are congruent with the domain's *place vocabulary* (Forbus, 1983). A place vocabulary is a set of related qualitative spatial relations for a particular problem-solving domain. For example, for a clock domain, the "meshing" and direction of gears is a crucial relationship (Forbus, Nielsen, & Faltings, 1991).The advantage of using place vocabularies to guide diagram redrawing is that each relationship in the place vocabulary may encapsulate any number of low-level geometric relations. For example, the meshing of single gear tooth and gap may involve a number of line segments and angles that must be of equivalent dimensions.

In the research described here, place vocabularies are integrated into a drawing system by reusing existing diagrammatic inferencing rules. We start with the rules used by our diagrammatic reasoning engine, GeoRep (Ferguson & Forbus, 2000), which infer place vocabulary relations from line drawings. We then capture critical low-level visual relations given as triggers in those rules, and enforce them in the current drawing. In the completed version of the project, the constraints will be integrated into GeoRep's truth-maintenance system, allowing place vocabulary constraints to be assumed and retracted based on the current diagram context. We believe that place vocabulary constraints will be easier for users to work with when they redraw diagrams. Because the constraints are derived from existing GeoRep rules, we also believe that they will be easy to develop and extend.

Diagram Redrawing in Problem Solving

Visual problem solving with diagrams often involves continual redrawing. For example, a software engineer may redraw a system diagram to take account of new inputs or outputs, or to break out the design of a particular module. Alternatively, an architect may expand and re-orient the shape of a family room in a floor plan, interactively reasoning about how expanding the room changes the character of the adjacent kitchen. Redrawing can occur for several reasons – it may involve changes to make the diagram more aesthetically pleasing, but also may involve changing the diagram to work though a set of alternatives, or to create a new diagram from a similar older diagram.

For many drawing systems, however, the process of redrawing introduces disfluencies. One problem in redrawing is that a single change at the conceptual level may require many low-level geometric modifications, each of which must be done separately. Widening a door in an architectural plan, for example, might involve lengthening the top and bottom of the door and moving the left and right sides. Although a linear stretching of the door might work in some cases, not all horizontal dimensions would be affected in the same way. For example, if the door contains a doorknob, the door could be widened, but the doorknob would not change dimensions, and might remain

a fixed distance from one side of the door, rather than a proportional distance from the side of the door. In the process of making these changes, however, the process of redrawing should avoid engaging the problem solver in unrelated subsidiary tasks.

Drawing programs attempt to reduce these disfluencies with mechanisms that make secondary corrections based on the user's direct modifications. Secondary corrections are geometric changes that are not directly manipulated by the user, but which make the rest of the diagram congruent with the changed elements. In general, these techniques are either low-level geometric constraints that are applied universally, or domain-dependent glyph-based constraints.

Low-level geometric constraints, such as "snap and glue," impose simple geometric constraints on new or modified visual elements in a global fashion. By imposing these constraints universally, it is more likely that the diagram as a whole will have a consistent style. For example, by restricting a line segment's orientation to one of several preset angles, or its endpoints to particular points on a grid, the eventual appearance of the whole diagram will be more consistent. Similarly, if one end of a line segment is "glued" to a polygon, the line segment can follow the polygon as it is moved around, eliminating the need for users to perform the correction themselves. Low-level constraints have been available since the time of the very first computer-based drawing systems (Sutherland, 1963).

Low-level geometric constraints have several advantages. First, they do not depend on the domain of the drawing, but can be applied universally. In addition, they are easily understood by the user, or can be made understandable with forms of passive feedback that do not interfere with the drawing process. For example, the grid used for snapping endpoints can be shown in the background, or the point of attachment can be briefly highlighted when a line segment is attached to a polygon. Additional audio cues may also be given. Finally, because they are relatively simple, when these constraints are inappropriate for the diagram domain, the user can usually turn them off.

The central problem with low-level constraints is their universal application across the figure. In many diagrams, different constraints may be needed for different parts of the same figure. For example, when block shapes represents a pair of software modules and a connecting arrow represents a communication link between them, it makes sense that the arrow should be stretched to follow the block shapes as it is moved. However, if a polygon represents a block mass and an attached arrow represents a force on that mass, stretching the arrow may imply a change in the amount of force, and so the arrow should be moved, but not stretched. For this reason, globally-enforced low-level constraints may increase the disfluencies in problem-solving when they act against the intent of the user.

One technique for handling the contextual nature of these constraints is to handle secondary corrections with custom glyphs, which are pre-built shapes that have domain-specific constraints on how they may be modified. These constraints are specifically tied to the domain. For example, the glyphs for logic circuits may allow gate glyphs to be rescaled, but not stretched along a single dimension.

In practice, custom glyphs can be difficult to create and extend. In systems such as Microsoft Visio, the constraints on such glyphs must be programmed in Visual Basic, although some general constraints (such as maintaining the same size aspect along vertical and horizontal dimensions) can be handled through dialogues. Because the constraints are often specified in terms of a set of low-level geometric constraints attached to a particular glyph, there is insufficient domain knowledge attached to the glyph to either extend the constraints to similar glyphs or to explain, in domain-specific terms, why particular constraints are being enforced for the glyph.

In this context, it is useful to consider an intermediate form of secondary correction, one that is more context sensitive than low-level geometric constraints, and is less brittle and more extensible than custom glyphs.

Place Vocabulary Constraints

We are currently building a drawing system that can perform secondary corrections using *place vocabulary constraints* (PVCs). These constraints perform secondary corrections on the diagram, but base these corrections on the place vocabulary of the diagram domain. For example, a line segment that is connected to the midpoint of second segment will not change if the second segment is moved. If, however, the first segment is identified as a wire, and the second as part of a logic gate, we extend the segment to preserve the *connected* relationship between the wire and the gate. PVCs attempt to preserve the visual relationships that support the current diagram interpretation.

PVCs are implemented on top of the GeoRep diagrammatic reasoning engine (Ferguson & Forbus, 2000), a system that produces a diagram representation from a set of low-level visual elements. For example, given the SR flip-flop logic circuit in Figure 1, GeoRep produces the representation in Figure 2. GeoRep is designed to explore the relationship between early visual perception (which detects a set of qualitative visual relationships, such as parallel relations) and visually-driven reasoning tasks, such as circuit design. GeoRep has been applied to several domains, including logic circuits, military planning diagrams (Ferguson, Rasch, Turmel, & Forbus, 2000), abstract psychological stimuli (Ferguson, 2000), and simple juxtaposition-based physics diagrams (Ferguson & Forbus, 1998).

GeoRep has a two-level architecture (Figure 3).The first level represents domain-independent visual structure, white the second represents high-level visual relations.

The first level, the low-level relational describer (LLRD), creates a description of the low-level visual relations shown to be used in early qualitative perception. These relations include such relations as connectivity,

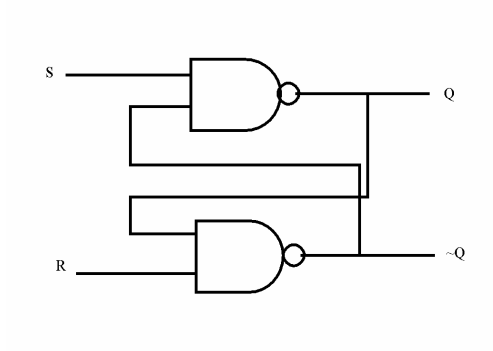

Figure 1: SRFF logic circuit diagram.

```
(SYSTEM-OUTPUT WIRE-S11)
(SYSTEM-OUTPUT WIRE-S15)
(INPUT-LABEL <POSITIONED-TEXT-3> WIRE-S1 "S")
(INPUT-LABEL <POSITIONED-TEXT-4> WIRE-S10 "R")
(INTERNAL-CONNECTION WIRE-12)
(INTERNAL-CONNECTION WIRE-11)
(SYSTEM-INPUT WIRE-S10)
(SYSTEM-INPUT WIRE-S1)
(NAND-GATE NAND-GATE-2)
(NAND-GATE NAND-GATE-1)
(OUTPUT-LABEL <POSITIONED-TEXT-1> WIRE-S11 "~Q")
(OUTPUT-LABEL <POSITIONED-TEXT-2> WIRE-S15 "Q")
```

Figure 2: Representation produced by GeoRep

parallelism, and boundary descriptions. These relations are based on those known to be found in early vision.

The second level, the high-level relational describer (HLRD), uses the low-level description as the basis for a high-level, domain dependent place vocabulary.

GeoRep uses a set of rules, called a visual domain theory, to represent the relationships between a set of low-level visual relations derived from human perception and the higher-level place vocabulary for the domain. A particular rule in the visual domain theory will collect a set of low-level relations, perhaps also making additional visual tests, and use them to infer a particular spatial relationship.

How PVCs work with GeoRep

The architecture of the GeoRep-PVC system is given in Figure 4. Here, the place vocabulary constraints are enforced by a new geometric constraint engine that interacts with the HLRD and the drawing interface.

As can be seen in the figure, the PVC system mediates between the low-level and high-level relational description in GeoRep, capturing visual relations as they are used by the HLRD to recognize place vocabulary relations, and then enforcing those relations using a constraint engine. For each rule, the PVC system determines which triggering visual relations can be enforced (such as *corner* or *parallel*), so that when a rule fires, constraints are set up

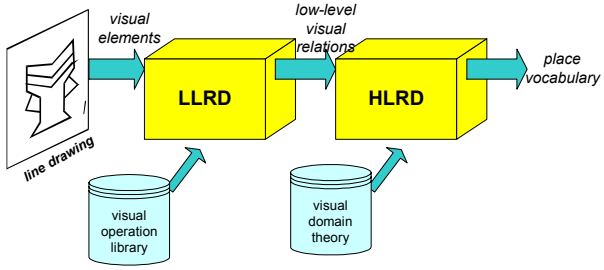

Figure 3: A simplified schematic of GeoRep's architecture.

based on those used to identify the glyph or the place vocabulary.

An example of how constraints are derived from a GeoRep rule is shown in Figure 5. This example shows a simple rule that is used to recognize a particular kind of NAND gate. This rule checks two attachment relations between an arc and a segment, one parallel relationship, one corner relationship, a polyline relationship, and one abuts relationship between a circle and arc. When the rule fires, GeoRep-PVC automatically creates new constraints based on those relations. The set of constrainable relations are defined in advance, and instances of constrainable relations are detected in each rule during rule compilation. A similar set of constraints (not shown) are created for the wire connected to the output of the NAND gate, enforcing the radial connection between the circle and the segment representing the output wire.

The effect of these constraints on drawing is demonstrated in Figure 6. When the bottom segment of the NAND gate is moved, the set of constraints modify the drawing to ensure that the result is still a NAND gate, and that the wire is still attached to the output of the gate.

The constraints themselves are handled via a constraint

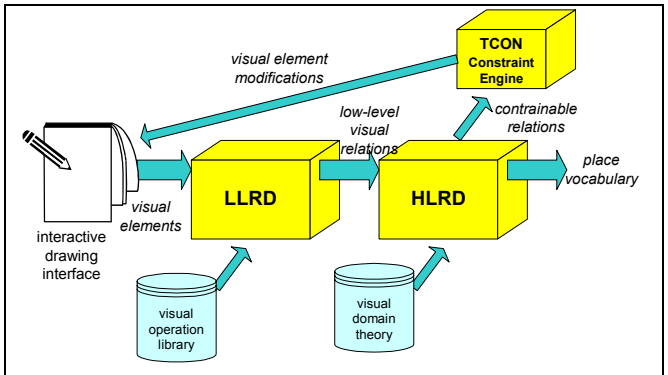

Figure 4: The PVC version of GeoRep adds an interactive drawing interface and a TCON-based constraint engine to the GeoRep architecture. The HLRD rules create the place vocabulary constraints, which are then dynamically enforced by the TCON constraint engine.

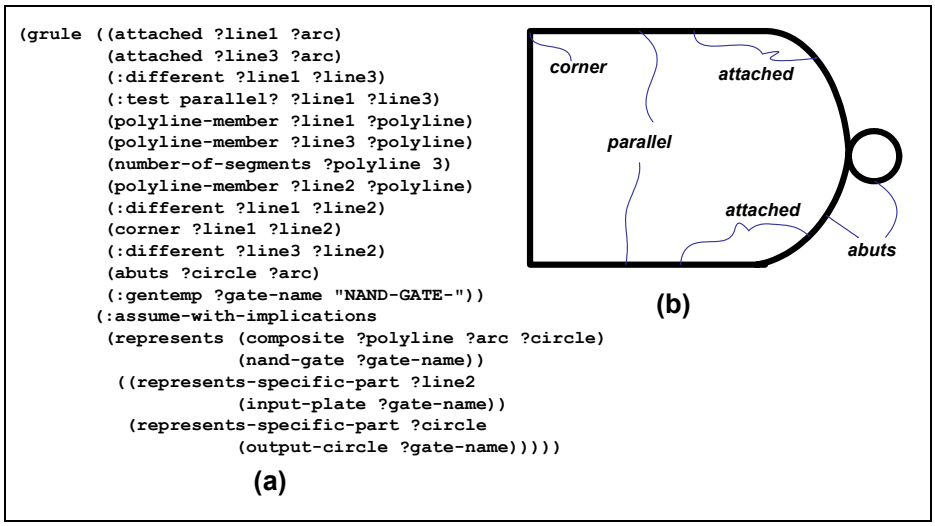

Figure 5: How PVC constraints are derived from existing GeoRep visual domain rules. Figure (a) shows an unmodified GeoRep rule for recognizing a NAND gate. When this rule is run with the PVC system, it automatically creates a set of constraints (b) based on those in the rule.

engine based on CONLAN (Forbus, 1981), called TCON (Forbus & de Kleer, 1993). TCON has a number of useful characteristics that it inherits from CONLAN. First, individual constraint templates can be reused as needed and new constraint templates can be created by aggregating previously-defined templates. TCON, like many constraint languages, is based on the insights of an earlier constraint system which was also used for geometric constraint processing and simulation, ThingLab (Borning, 1981).

We have extended TCON by interfacing it with the HLRD in GeoRep, and by enhancing its conflict-handling mechanism. In the original TCON, conflicts between constraints are handled by an individual handler. In the TCON version used in GeoRep-PVC, contradiction handlers depend on the set of constraints being used, and there are "handler stacks" for each constraint that allow multiple resolution methods to be tried for any particular set of conflicting constraints. For example, in the correction methods we currently use, we prefer to retain corner constraints over perpendicular constraints.

In the full version of GeoRep-PVC, we expect there will be two more critical additions. First, the rules used by the HLRD and the constraint system will be richer than those in the original GeoRep. Second, the constraints will be attached to the dependency network created by the HLRD.

The reason for the richer rules is that most of the rules used for glyph recognition by GeoRep are not specific enough. While they are sufficient for recognizing particular glyphs in situations where near-misses are not available, they do not fully test the set of relations that a human would in recognizing a particular glyph. For example, in recognizing a NAND gate, the output circle should be much smaller than the arc it abuts. However, that is not currently checked. This results in potential false positives for recognition, and also means that the NAND gate in underconstrained for some types of redrawing.

GeoRep used sparse rules, in part, because glyph recognition in GeoRep was brittle. However, recent work in our lab by Bokor & Ferguson (2004) has shown that relatively rich rules can be used if combined with a probabilistic weighting mechanism. When this system is integrated into GeoRep-PVC, it should be possible to have rich rules that are neither brittle, nor are underconstrained in the drawing interface.

The second addition, attaching PVC constraints to dependency network nodes, will allow the system to assume or retract PVCs based on the diagram context. If changes to the diagram change the glyph interpretations, constraints attached to those interpretations will no longer be enforced. For example, if a segment in a NAND gate is removed, the system will retract the NAND gate interpretation for the figure. When the NAND gate interpretation is retracted, all constraints based on that interpretation will also be retracted. The result of this will be that when a figure is changed in a way that eliminates its previous interpretation, it will be possible to manipulate it in a way that would have violated the previous constraints.

Finally, attaching PVC constraints to the dependency network will allow the system to explain the constraints it is enforcing in terms of the domain, not simply in terms of the geometric constraints themselves. For example, if the user asks why a wire's position was changed, the geometric explanation might be "because this circle was moved, and because this segment must stay connected to this circle, we stretched the segment." With a link to the knowledge-rich dependency network, it should be possible to say instead something like "because the NAND gate

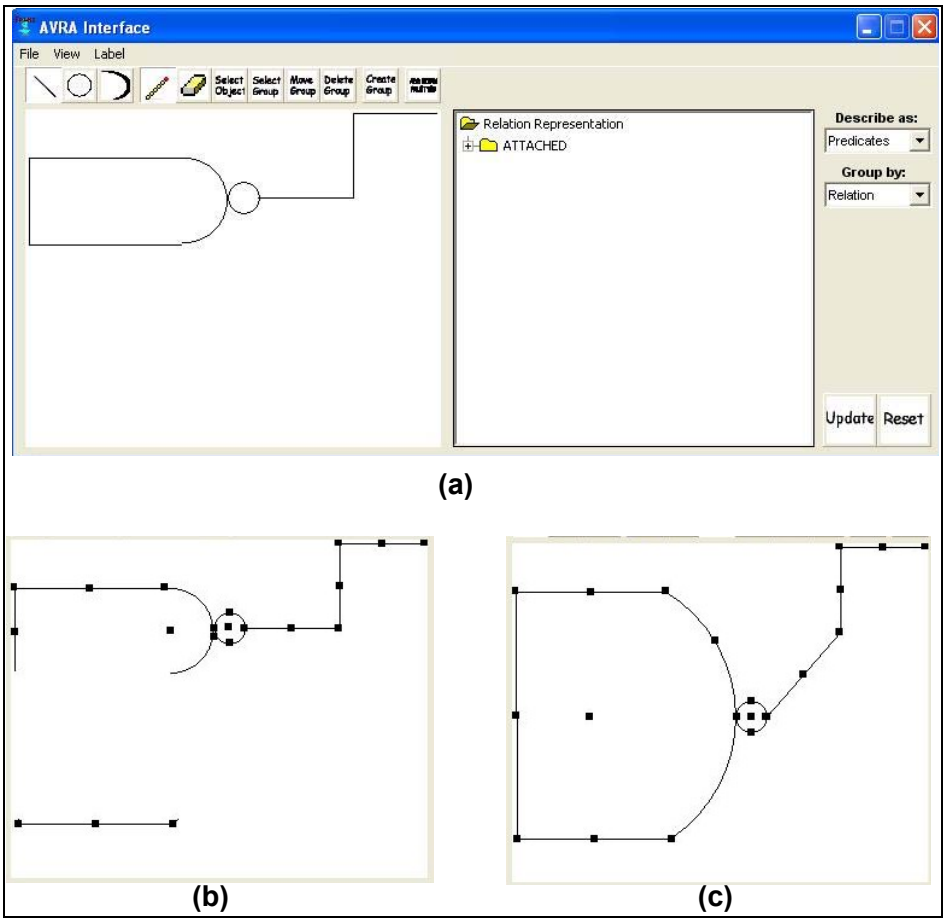

Figure 6: Example of PVCs in action. In (a), the user draws a NAND gate and an output wire using the built-in drawing tools. The system recognizes the NAND gates and automatically creates a set of constraints based on the relations that contributed to the recognized glyph. In (b), the user grabs and drags a single line segment to the bottom of the screen. In (c), the system uses constraints derived from the original recognition rule to change the other visual elements to retain the NAND gate and output wire interpretation.

moved, we stretched the wire to ensure that it was still connected the GATE's output."

Related Work and Discussion

GeoRep is only one in a series of recent diagrammatic reasoning systems that integrate knowledge-based reasoning with geometric or spatial reasoning. These systems include the Electronic Cocktail Napkin (Gross, 1996), Quickset (Cohen et al., 1997), several systems by Davis (2002), and sKEA (Forbus & Usher, 2002).

GeoRep-PVC owes a great debt to many constraint-based systems that have gone before, starting with ThingLab (Borning, 1981). Systems such as Sketchpad (Sutherland, 1963) demonstrated the general utility of low-level constraints in drawing systems, while others such as Geometer's Sketchpad (Scher, 2000) demonstrated the use of knowledge-based constraints in a particular domain. The difference between GeoRep-PVC and previous systems are 1) that it is part of an attempt to link place vocabularies, as defined in qualitative spatial reasoning, with user interfaces; and 2) that the system integrates the constraint system with a symbolic reasoning system, in part to provide better explanations about the constraints to users.

This work is part of a much larger effect to create an *advanced diagrammatic reasoning architecture* (ADRA). As part of this new architecture, we have developed an incremental spatial reasoning mechanism that allows dynamically changing input (Ferguson, Bokor, Mappus, & Feldman, 2003), and a mechanism for probabilistic recognition of glyphs (Bokor & Ferguson, 2004). Additional work is underway on an attention model for the system, and on integration of the spatial reasoner with our regularity-detection module, MAGI (Ferguson, 1994).

Continued development should lead to an effective system that can be used in a variety of domains. Our initial prototype of the system (as demonstrated here) is complete, and we are expanding the set of figures it handles. Once this is done, we will test it with users to refine its capabilities.

References

Bokor, J. L., & Ferguson, R. W. (2004). *Integrating probabilistic reasoning into a symbolic diagrammatic reasoner.* Paper presented at the Qualitative Reasoning Workshop, Evanston, IL.

Borning, A. (1981). The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems, 3*, 355-387.

Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., et al. (1997). QuickSet: Multimodal interaction for distributed applications. In *Proceedings of the Fifth Annual International Multimodal Conference* (pp. 31-40). Seattle.

Davis, R. (2002). *Position statement and overview: Sketch recognition at MIT.* Paper presented at the AAAI Spring Symposium on Sketch Understanding, Palo Alto, CA.

Ferguson, R. W. (1994). MAGI: Analogy-based encoding using symmetry and regularity. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 283-288). Atlanta, GA: Lawrence Erlbaum Associates.

Ferguson, R. W. (2000). Modeling orientation effects in symmetry detection: The role of visual structure. In *Proceedings of the 22nd Conference of the Cognitive Science Society* (pp. 143). Hillsdale, New Jersey: Erlbaum.

Ferguson, R. W., Bokor, J. L., Mappus, R. L., IV, & Feldman, A. (2003). Maintaining spatial relations in an incremental diagrammatic reasoner. In W. Kuhn, M. Worboys & S. Timpf (Eds.), *COSIT 2003, Spatial Information Theory*: Springer-Verlag.

Ferguson, R. W., & Forbus, K. D. (1998). Telling juxtapositions: Using repetition and alignable difference in diagram understanding. In K. Holyoak, D. Gentner & B. Kokinov (Eds.), *Advances in Analogy Research* (pp. 109-117). Sofia, Bulgaria: New Bulgarian University.

Ferguson, R. W., & Forbus, K. D. (2000). GeoRep: A flexible tool for spatial representation of line drawings. In *Proceedings of the 18th National Conference on Artificial Intelligence* (pp. 510-516). Austin, Texas: AAAI Press.

Ferguson, R. W., Rasch, R. A. J., Turmel, W., & Forbus, K. D. (2000). *Qualitative spatial interpretation of Course-of-Action diagrams.* Paper presented at the Intelligent Systems Demonstrations, 18th National Conference on Artificial Intelligence, Austin, Texas.

Forbus, K. D. (1981). *A CONLAN Primer* (No. 4491): Bolt Beranek and Newman Inc.

Forbus, K. D. (1983). Qualitative reasoning about space and motion. In D. Gentner & A. Stevens (Eds.), *Mental Models* (pp. 53-73). Hillsdale, NJ: Lawrence Erlbaum Associates.

Forbus, K. D., & de Kleer, J. (1993). *Building Problem Solvers*. Cambridge, MA: The MIT Press.

Forbus, K. D., Nielsen, P., & Faltings, B. (1991). Qualitative spatial reasoning: The CLOCK project. *Artificial Intelligence, 51*(1-3).

Forbus, K. D., & Usher, J. (2002). Sketching for knowledge capture: A progress report. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (pp. 71-77). San Francisco.

Gross, M. D. (1996). The Electronic Cocktail Napkin: A computational environment for working with design diagrams. *Design Studies, 17*(1), 53-69.

Scher, D. (2000). Lifting the curtain: The evolution of the Geometer's Sketchpad. *The Mathematics Educator, 10*(2), 42-48.

Sutherland, I. E. (1963). *Sketchpad, a Man - Machine Graphical Communication System.* Unpublished PH.D., Massachusetts Institute of Technology, Cambridge, MA.

Acknowledgements

This research was supported by the National Science Foundation under the Artificial Intelligence and Cognitive Science program, as well as by Vulcan, Inc. through the HALO II initiative.