LADDER: A Sketch Recognition Language

Tracy Hammond and Randall Davis

What: We have created LADDER [5], a language to describe how sketched diagrams in a domain are drawn, displayed, and edited. The difficulty in creating such a language is choosing a set of predefined entities that is broad enough to support a wide range of domains, while remaining narrow enough to be comprehensible. The language consists of predefined shapes, constraints, editing behaviors, and display methods, as well as a syntax for specifying a domain description sketch grammar, ensuring that shapes and shape groups from many domains can be described. The language allows shapes to be built hierarchically (e.g., an arrow is built out of three lines), and includes the concept of "abstract shapes", analogous to abstract classes in an object oriented language. Shape groups describe how multiple domain shapes interact and can provide the sketch recognition system with information to be used in top-down recognition. Shape groups can also be used to describe "chain-reaction" editing commands that effect multiple shapes at once. To test that recognition is feasible using this language, we have built a simple domain-independent sketch recognition system that parses the domain descriptions and generates the code necessary to recognize the shapes.

Why: As pen-based input devices have become more common, sketch recognition systems are being developed for many domains such as mechanical engineering [1], UML class diagrams [4], webpage design [7], course of action diagrams [9], and many others. These sketch recognition systems allow users to more naturally sketch a design, as opposed to a traditional mouse and palette tool [6]. However, a sketch recognition system can be quite time consuming to build, with each system being built to handle the intricacies of each domain.

To date, sketch recognition systems have been domain-specific, with the recognition details of the domain hard-coded into the system. Developing such a sketch interface is a substantial effort. We propose instead that recognition be performed by a single domain-independent recognition system that uses a domain specific sketch grammar (an approach used with some success in speech recognition [11]). Programmers could then create new sketch interfaces simply by writing a sketch grammar describing the domain-specific information. LADDER is an improvement on existing shape languages which are not hierarchical [8] or don't provide for non-graphical information [10, 3, 2], such as editing behaviors or stroke order and direction.

How: The language consists of predefined shapes, constraints, editing behaviors, as well as a syntax for combining them. A domain description includes definitions of the shapes in the domain. A shape definition is consists of several sections. The *is-a* section (line 2) is an indication of any class of abstract shapes that it belongs to. The *components* (line 3) include the sub-shapes of which this shape is composed. (Shapes are defined hierarchically.) Note that the TriangleArrow is composed of a predefined shape Line as well a user defined shape Arrow. The *constraints* (line 4) specify the necessary relationships or hard constraints. They can also specify soft constraints that may not always occur in the ideally drawn shape and are thus may not be seen in the drawn object. However, these soft constraints typically occur often enough to be useful to the recognition process. For instance, a probable drawing order may be shaft, head1, l, head2. An optional numerical probability may follow the keyword word *soft*

The *aliases* (line 5) allows us to compute certain properties and name them for use later. The *display* section (line 6) defines how the shape should be displayed after it is recognized. A shape or its components may be displayed in any color in four different ways: 1) the original strokes of the shape, 2) the cleaned up version of the shapes where the best-fit primitives of the original strokes are displayed, 3) the ideal shape where primitive components with the constraints solved, or 4) another custom shape which specifies which shapes (line, circle, rectangle, etc.) to draw and where. *Editing behaviors* (line 7) define how a shape can be moved or deleted after it is recognized. The editing behavior below allows the user to move the entire arrow by clicking and dragging the shaft. The user can also click and drag the head or tail of the arrow while the opposite end remains fixed; the shaft stretches and rotates as appropriate.

Progress: Thus far, we have created a first version of the language. We have asked 35 people to describe shapes in natural language to help insure an intuitive vocabulary for the language. In order to test LADDER and our framework, we have developed a simple domain independent recognition system which takes a domain description as input.

```
(define shape TriangleArrow
  (comment "An arrow with a triangle head")                                    %1
  (is-a AbstractArrow)                                                         %2
  (components (Arrow a) (Line l))                                              %3
  (constraints                                                                 %4
    (meet l.p1 a.head1.p1) (meet l.p2 a.head2.p1)
    (angle a.shaft l 90) (angle l a.head1 45) (angle l a.head2 45)
    (soft draw-order a.shaft a.head1 l a.head2))
  (derived-properties                                                          %5
    (Point head a.shead) (Point tail a.tail)
    (Line shaft a.shaft) (Line head1 a.head1) (Line head2 a.head2))
  (display (original_strokes shaft)(ideal_strokes l head1 head2)              %6
  (editing                                                                     %7
   (edit (trigger (hold_drag head)) (action (rubberband head tail)))
   (edit (trigger (hold_drag tail)) (action (rubberband tail head)))
   (edit (trigger (hold_drag shaft)) (action (translate this)))
   (scribble shaft (delete this))))
```

Future: In the future, we will test human usability by asking users to develop domain descriptions using the proposed language. We are also working on a GUI to help users more easily describe shapes.

Research Support: This work is supported in part by the MIT/Microsoft iCampus initiative and in part by MIT's Project Oxygen.

References:

[1] Christine Alvarado. A natural sketching environmant: Bringing the computer into early stages of mechanical design. Master's thesis, MIT, 2000.

[2] Oliver Bimber, L. Miguel Encarnacao, and Andre Stork. A multi-layered architecture for sketch-based interaction within virtual environments. *Computer and Graphics: Special Issue on Calligraphic Interfaces: Towards a New Generation of Interactive Systems*, 24(6):851–867, 2000.

[3] Anabela Caetano, Neri Goulart, Manuel Fonseca, and Joaquim Jorge. Sketching user interfaces with visual patterns. *Proceedings of the 1st Ibero-American Symposium in Computer Graphics (SIACG02)*, pages 271–279, 2002.

[4] Tracy Hammond and Randall Davis. Tahuti: A geometrical sketch recognition system for uml class diagrams. *AAAI Spring Symposium on Sketch Understanding*, pages 59–68, March 25-27 2002.

[5] Tracy Hammond and Randall Davis. Ladder: A language to describe drawing, display, and editing in sketch recognition. *Proceedings of the 2003 Internaltional Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[6] Heloise Hse, Michael Shilman, A. Richard Newton, and James Landay. Sketch-based user interfaces for collaborative object-oriented modeling. Berkley CS260 Class Project, December 1999.

[7] James Lin, Mark W. Newman, Jason I. Hong, and James Landay. Denim: Finding a tighter fit between tools and practice for web site design. In *CHI Letters: Human Factors in Computing Systems, CHI 2000*, pages 510–517, 2000.

[8] James V. Mahoney and Markus P. J. Frommerz. Three main concerns in sketch recognition and an approach to addressing them. In *Sketch Understanding, Papers from the 2002 AAAI Spring Symposium*, pages 105–112, Stanford, California, March 25-27 2002. AAAI Press.

[9] J. Pittman, I. Smith, Phil Cohen, Sharon Oviatt, and T. Yang. Quickset: A multimodal interface for military simulations. *Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation*, pages 217–224, 1996.

[10] George Stiny and James Gips. Shape grammars and the generative specification of painting and sculpture. In C V Freiman, editor, *Information Processing 71*, pages 1460–1465. North-Holland, 1972.

[11] Zue and Glass. Conversational interfaces: Advances and challenges. *Proc IEEE*, pages 1166–1180, 2000.