AAAI-86 Workshop on Intelligence in Interfaces

August 14, 1986

Participant List and Abstracts

Workshop Co-Chairmen

Bob Neches
USC / Information Sciences Institute

Tom Kaczmarek
Inference Corporation

Additional Organizers

Larry Miller and Norm Sondheimer
USC / Information Sciences Institute

List of Participants

- Brad Allen, Inference Corp., 5300 West Century Boulevard, #700, L.A., CA 90045. Telephone: (213)417-7997 *(Abstract, Pg. 1)*

- Yigal Arens, USC, Computer Science Dept., Los Angeles, CA 90089-0782. Telephone: (213)743-7848 *(Abstract, Pg. 4)*

- Ruven Brooks, Schlumberger-Doll, Old Quarry Road, Ridgefield, CT. 06877-4108. Telephone: (203)431-5515 *(Abstract, Pg. 7)*

- Carol Broverman, Karen Huff, & Victor Lesser, Computer and Information Science Dept., University of Massachusetts, Graduate Research Center, Amherst, MA 01003. Telephone: (413)545-1322 *(Abstract, Pg. 8)*

- John Seely Brown, Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304. Telephone: (415) 494-4000

- Dick Burton & Walter Johnson, Xerox PARC, Intelligent Systems Lab, 3333 Coyote Hill Rd., Palo Alto, CA 94304. Telehone: (415)494-4343 *(Abstract, Pg. 13)*

- Jack Carroll, IBM Watson Research Center, Yorktown Heights, NY 10598. Telephone: (914)945-2947 *(Abstract, Pg. 15)*

- W.B. Croft, Dept. of Computer & Info. Science, University of Massachusetts, Amherst, MA 01003. Telephone: (413)545-1322

- Joseph Deken, NSF, Div. of Computer Research, Washington, D.C. 20550. Telephone: (202) 357-1185

- Kate Ehrlich, Symbolics, Inc., User Interface Group, 11 Cambridge Center, Cambridge, MA 02142. *(Abstract, Pg. 21)*

- Ross Fanuef, Digital Equipment Corporation, 77 Reed Road, HL02-3/C10, Hudson, MA 01749. Telephone: (617)568-5480.

- Gerhard Fischer, University of Colorado at Boulder, Computer Science Dept., Campus Box 430, Boulder, CO. 80309. Telephone: (303) 492-7514 *(Abstract, Pg. 24)*

- Peter Friedland, NASA (Until January 1: Knowledge Systems Laboratory, Stanford University, 701 Welch Road, Palo Alto, CA 94304. Telephone: (415) 723-3728)

- James Geller & Stuart Shapiro, SUNY Buffalo, Department of Computer Science, 226 Bell Hall, Amherst Campus, State University of New York at Buffalo, 14260. Telephone: (212) 460-7209 *(Abstract, Pg. 31)*

- Phil Hayes, Carnegie-Mellon University and Carnegie Group, Computer Science Department, Pittsburgh, PA 15213. Telephone: (412) 268-2631

- Ellen Hays, University of Pennsylvania, Philadelphia, PA. *(Abstract, Pg. 37)*

- Haym Hirsh, Knowledge Systems Laboratory, Stanford University, 701 Welch Road, Building C, Palo Alto, CA 94304. Telephone: (415) 723-2273 *(Abstract, Pg. 40)*

Science, 562 McBryde Hall, Blacksburgh, VA 24061. *(Abstract, Pg.81)*

- Allen Sears, DARPA, 1400 Wilson Blvd, Arlington, VA 22209. Telephone: (206) 692-7844

- Norm Sondheimer, USC / Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292. Telephone: (213) 822-1511 *(Abstract, Pg. 84)*

- Marilyn Stelzner & Michael Williams, Intellicorp, 1975 El Camino Real West, Mountain View, CA 94040. Telephone: (415) 965-5500 *(Abstract, Pg. 91)*

- Dick Waters, MIT, Artificial Intelligence Lab, Cambridge, MA 02139-1986. Telephone: (617)- 253-6037

- Bob Wilensky, U.C. Berkeley, Computer Science Division, Department of Electrical Engineering & Computer Science, Berkeley, CA 94720. Telephone: (415) 642-7034

- Ursula Wolz, Columbia University, 463 Computer Science Building, New York, NY 10027. Telephone: (212) 280-8124 *(Abstract, Pg. 93)*

Knowledge Based Interfaces

James Geller & Stuart Shapiro, Department of Computer Science, 226 Bell Hall, Amherst Campus, SUNY Buffalo, Buffalo, NY 14260

In this abstract we will first explain how our work reflects our viewpoint on the problem of intelligent interfaces and then discuss our specific research project.

Almost any serious human communication makes use of different modalities. People transmit more than words. They have, for example, anger or pity in their voice, and most people underline their conversations with gestures. The flow of conversation is controlled by looking into each other's eyes, or by looking away.

Lately AI has been trying to get away from simple printed text, and many AI programs include a graphics component. Researchers in multi media communication are working on voice mail and the integration of pictures into textual documents. This is a very valuable development, but we see the danger that some application oriented research might find itself in the same situation as early machine translation: the knowledge representation basis is too thin to be built upon.

There is a distinct danger that the graphics interface does not understand the language that the language interface uses. It seems more important to design a knowledge representation that is able to express broad knowledge of

whatever is being represented and then attach different front ends to it, than to use a complex front end that has only little access to internal states of the rest of the program. The necessity for a good underlying representation is probably the most important message that AI can give to user interface design.

In the spectrum between apprentice system and tool package, one would probably try to place us on the side of the tools. But in fact we are starting one step earlier. Tools have a "procedural flavor" and we try to define a powerful representation before we talk about tools at all.

Our key philosophy is that information should be displayed graphically whenever possible. Objects can be displayed if the system knows what they look like or can create a drawing from its knowledge of their parts, etc. Properties can be displayed graphically if they are physical and the system knows how they affect the relevant object, or if the system knows how to symbolize the property graphically.

It is possible to display propositions or objects, both denoted by semantic network nodes. When given an object, it is displayed having all its displayable properties. When

given a proposition that an object has a displayable property, the object is displayed as having that property. All graphical information is stored in a propositional representation, except for a small number of primitive forms. The network database contains assertions that link objects to forms and specify absolute positions and relative positions. Other information deals with part hierarchies, assemblies of objects, and attributes.

Attribute dimensions are implemented as functionals, applied to form functions. We view "inheritability" as an assertable meta-attribute. As opposed to most inheritance schemes we are using inheritance along a class and along a part hierarchy. These two ideas together permit properties to be inherited along a part hierarchy if this is deemed appropriate, as would be the case for size, but not for faultiness.

Concerning our implementation, we have been working on an expert maintenance system for circuit board diagnosis. This system reasons through a faulty circuit board and identifies a single component or a set of components which are responsible for the problem. The knowledge base used is implemented in the SNePS semantic network system. During the reasoning process a graphical tracing system informs the

user what the system is currently "thinking" about.

A graphics terminal displays a wire plan of the device. All parts are by default shown in blue. A set of colors is used to indicate state changes symbolically. When the user supplies information that a certain output of the device does not show the expected value, the blue output port becomes magenta, our color symbolizing a violated expectation.

The reasoning process first generates a set of suspects. This results in changing suspect components to green. While the reasoning proceeds the system focuses on different components, and the current focus object is always displayed blinking.

The final result of the reasoning process is the display of the faulty parts in red, with all "acquited" parts turned back to blue, and all other parts unchanged in green or magenta.

Our analysis and implementation of graphical knowledge is totally independent from the reasoning and maintenance system, sharing only a common knowledge base with it. In fact display can be used in a totally different setting. In order to explain this we have to introduce two more of the

tools used in our system.

The first one permits a user to create a piece of display code by drawing the object, without any programming. We refer to this utility as "readform" and we use it to create the "primitive forms" referred to earlier.

The second utility is an ATN interpreter/compiler that is part of the SNePS system. These two parts, in cooperation with display, permit a user to create a device (from a limited class) by having a dialogue with the system. For instance a user can first draw a multiplier (using readform) and give it the name "xmult". Then he can tell the system in natural language: D1M1 is a multiplier. The form of a multiplier is xmult. D1M1 is at 200 200. D1M2 is 100 right and 50 above D1M1. D1M2 is a multiplier. If he then asks the system to "show D1M1", then it will correctly draw D1M1. A large number of more complicated natural language requests is possible, dealing with all aspects of part hierarchies, etc. that the system understands.

We would now like to make short summarizing statements about the topics to be discussed. We take it as obvious that any non-trivial interface can only gain if it behaves "intelligently". We hold that the need to find a good

representation is one of the most fundamental and fundamentally applicable lessons from AI. If an interface uses a different representation from the "main program", then an additional translation step comes in immediately which can only complicate matters. So the domain program has to use a representation which is amenable to different modalities of user interfaces. With this claim we require more than a symbiotic relationship, we require an integrated system. On the other hand it seems difficult to add an intelligent interface on to a "dumb" system.

Our display function is a graphical "generation" function, in full analogy to a language generation program. This enables us to talk about a much broader class of abstraction than any type of "language only" interface could.