Dynamic Spatial Reasoning Capability in a Graphical Interface Evaluation Tool

Michael Matessa (mmatessa@alionscience.com)
Rick Archer (rarcher@alionscience.com)
Rebecca Mui (rmui@alionscience.com)
Alion Science and Technology
Micro Analysis & Design Operation
1789 South Braddock Avenue, Suite 400
Pittsburgh, PA 15218 USA

Abstract

This paper describes dynamic and spatial reasoning enhancements to the Graph-Based Interface Language tool (GRBIL), which creates ACT-R models by demonstration. A new ability for users to create monitors enables procedures to be dynamically triggered. A new integration of ACT-R with a diagrammatic reasoning theory allows ACT-R to perform spatial reasoning. Capabilities of the tool are demonstrated in a robotic control task.

Introduction

Cognitive modeling can add value to interface evaluation, but it is currently not very practical in terms of amount of expertise and time. Recent efforts to allow easier construction of cognitive models have utilized high level languages (Howes, Lewis, Vera, & Richardson, 2005; Salvucci & Lee, 2003; St. Amant & Ritter, 2005) and modeling by demonstration (Archer, Lebiere, & Warwick, 2005; John, Prevas, Salvucci, & Koedinger, 2004). Modeling by demonstration is an easy way to create a sequence of procedural steps, but with dynamic interfaces, an additional method is needed to designate the condition in which the procedure should be applied. In addition, many real-world tasks require aspects of spatial reasoning as well as dynamic interaction. Examples of interfaces with these features include radar operator interfaces and interfaces for controlling robotic vehicles. This paper describes dynamic and spatial reasoning enhancements to the GRaph-Based Interface Language tool (GRBIL – Archer, Lebiere, & Warwick, 2005), which creates ACT-R models by demonstration. A new ability for users to create monitors enables procedures to be dynamically triggered. A new integration of ACT-R with a diagrammatic reasoning theory allows ACT-R to perform spatial reasoning.

ACT-R

The ACT-R cognitive architecture has shown an increasing ability to account for human visual information processing. Early ACT-R accounts of visual processing made a distinction between pre-attentive features available to vision and objects available after a shift of attention (Anderson, Matessa, & Lebiere, 1997). More recent work uses brain imaging as evidence for an Imaginal module where information can be visualized and manipulated (Anderson et al., 2004). However, current ACT-R theory is limited in the visual objects that can be recognized (text, lines, rectangles, and ovals) and does not provide primitive operators for getting attribute information such as length, relational information such as what objects are inside or next to other objects, inferred information such as projected intersections, and transformations on objects such as rotation. One solution to these limitations is to integrate a spatial reasoning theory into ACT-R. For interface evaluation, diagrams are a useful level of representation for reasoning. Larkin and Simon (1987) point out that diagrams automatically support a large number of perceptual inferences which are extremely easy for humans but that "...diagrams are useful only to those who know the appropriate computational processes for taking advantage of them."

DRS

Chandrasekaran et al. (2004) developed the diagrammatic reasoning theory DRS, which consists of a basic set of primitive objects, information gathering capabilities called Perceptual Routines, and creation/modification operations called Action Routines. A diagrammatic object can be one of three types: point, curve, and region. Figure 1 shows examples of object types and how they can be composed hierarchically. Perceptual Routines can be qualitative (e.g., LeftOf, On, InsideOf), quantitative (e.g., Distance, Angle, Length), or related to object recognition (e.g., ScanPath, Intersect). Action Routines can create or modify objects (e.g., Translate, Rotate, PathFinder).

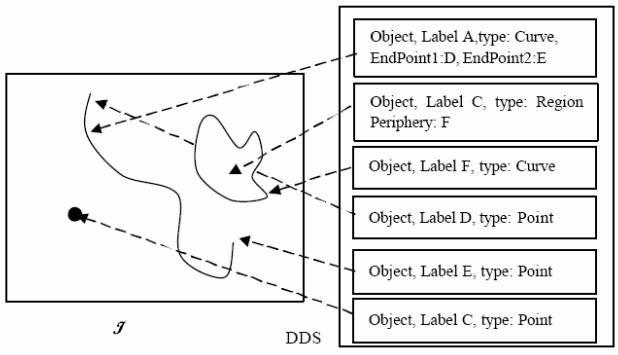

Figure 1: DRS object types

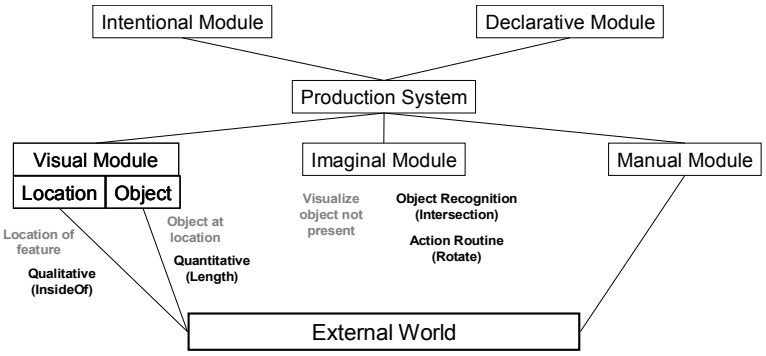

Figure 2: DRS routines in the ACT-R architecture

Chandrasekaran and Kurup (2007) incorporated DRS into Soar and used its chunking mechanism to learn efficient abstractions of observed visual material. The abstractions led to beneficial simplification in the case of memory of a specific path in a complex environment, and also led to incorrect but psychologically realistic reasoning in the case of determining whether Reno is east or west of San Diego (it is actually west). Lathrop and Laird (2006) incorporated a diagrammatic reasoning theory motivated by DRS into the Soar cognitive architecture and found an increase in functional capability and increase in efficiency of code as measured by number of decision cycles.

Combining DRS and ACT-R

Matessa and Brockett (2007) describe how the perceptual capabilities of ACT-R can be enhanced by the addition of DRS. Figure 2 shows where DRS routine types can naturally fit in the ACT-R architecture. One unnatural fit would be for qualitative routines to put all relational information (e.g., Region4 is above Region5) in the declarative module, which would cause a combinatorial explosion of facts. More naturally, qualitative routines can constrain locations returned by the visual module. For example, DRS could return the location of some object InsideOf a particular object. Quantitative routines can associate information with visual objects returned by the visual module. For example, DRS could associate the Length of a line with the returned visual object. Object recognition routines can return objects that are either literally in the display or implied by the display. For example, DRS could infer the projected line from a specified object to the nearest Intersecting object and return it to the imaginal module. Action routines can create or modify objects stored in the imaginal module. For example, DRS could Rotate an object in the imaginal module 45 degrees clockwise. To demonstrate these concepts, a model of maze navigation was created where the decision to move ahead or turn was based on the length of a projected line to the nearest intersecting wall. Without the integrated DRS, an ACT-R model would be forced to do maze navigation using mental arithmetic on coordinates instead of using more natural representations of intersection and line length.

In order to use DRS-enhanced ACT-R models to evaluate dynamic interfaces, the code from Matessa and Brockett (2007) was integrated into the GRBIL evaluation tool.

GRBIL

The goal of developing the GRaph-Based Interface Language tool (GRBIL) is to allow developers to easily design and evaluate system interfaces. The system allows the user to graphically define a system interface, demonstrate a set of procedures for using the interface, and automatically generate an ACT-R model of an interface operator, providing a time-stamped series of events and potential errors. In addition, the system allows the incorporation of dynamic models of the external world using a task network modeling environment named IMPRINT (IMProved Research INTegration Tool – Archer & Adkins, 1999). This enables the evaluation of interfaces that involve continually changing environments. Multiple IMPRINT tasks can run independently, and with multiple machines, multiple ACT-R agents can interact with IMPRINT in a common environment.

The first step in constructing an interface in GRBIL is to design the physical layout of the interface. This involves choosing the windows, subwindows, and interface controls that comprise the interface. This is done in a similar fashion to many modern interface layout applications using WYSIWYG drag and drop functionality. Once a control is added to a window, attributes such as size and background image may then be customized further from a menu. The second step in designing an interface is to provide a description of what actions each control will be capable of and what the desired effect of each action will be. This is done for each control in GRBIL via an "Event Actions" menu for each control. Using this process of adding interface windows, placing controls on those interfaces and then describing the effects of using those controls on the state of the interface, a GRBIL user can describe the functionality of an entire user interface.

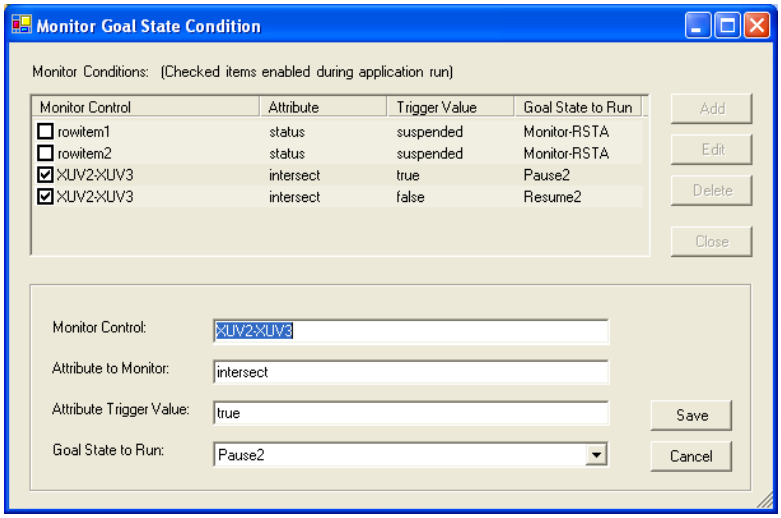

Figure 3: GRBIL monitor used to define a condition of procedure execution

The next step in setting up an interface for evaluation is to define the procedures that a user of the system might wish to perform. This is accomplished by demonstrating a series of actions (e.g., button clicks, object movement). The recorded series of actions is referred to as a "goal state" procedure in the GRBIL terminology. Any number of goal state procedures can be demonstrated and saved for later execution by an ACT-R model. In the model, action steps are represented as a list that is retrieved from declarative memory, and so can be used to predict errors that are dependent on sequence length and positional confusion (cf. Anderson & Matessa, 1997; Matessa & Polson, 2006). Model tracing is used so that potential errors are only noted in the output while the correct retrievals are actually made so the model runs correctly.

Once goal state procedures are defined, monitors can be created that designate the condition in which the procedure should apply. There are two types of monitors: object monitors that check the status of an interface object and spatial monitors that use DRS to check the status of graphical objects. Figure 3 shows the interface used to create monitors. A monitor control is either the name of an interface object, name of a graphical object or names o multiple graphical-objects. An attribute to monitor is an attribute of an interface object defined in GRBIL or a DRS routine for a graphical object. An attribute trigger value must match the value of an attribute before the given goal state procedure can be executed. Any number of monitors can be demonstrated and saved for later execution by an ACT-R model

At run time, goal state procedures and monitors are chosen and ordered. An ACT-R model is created that performs executes the procedures in the given order. The action steps of a goal state procedure are performed without interruption. Between goal state procedures, the model evaluates the next unmatched monitor, using necessary shifts of attention. If the monitor matches, the goal state procedure for that monitor is executed.

Output from the model run includes a time-stamped series of events and possible memory errors (failures to retrieve or the retrieval of similar but incorrect chunks).

Robotic Interface Test Case

In order to test the capabilities of ACT-R models generated with GRBIL, an interface for an unmanned vehicle Operator Control Unit (OCU) was selected (Figure 4). The OCU is used by operators to control unmanned vehicles in the field. Operators set up plans for the vehicles which are then executed independently. Operators are also responsible for monitoring the status of the vehicles. The interface for the OCU is quite complex, with several modes of operation and control menus. In our implementation, ACT-R performed the actions of operator agents and IMPRINT performed the actions of unmanned vehicle agents. Multiple IMPRINT vehicle agents can run independently, and with multiple machines, multiple ACT-R operator agents can interact with the IMPRINT vehicle agents in a common environment. The ACT-R operators are able to monitor specific attribute values in the environment (such as text describing the status of a vehicle) and use DRS spatial reasoning to detect more general conditions such as projected vehicle intersection (Figure 5). For a particular set of procedures, interacting models of multiple operators (one setting up and initiating vehicles, one monitoring) predict improved performance over a model of a single operator. This is a result of the ability of the purely monitoring operator to react at the same time the single operator would be busy with a procedure.

In order to validate the model's predictions of errors, latencies, and other performance measures, the project team has begun to collect performance data from human operators performing interface tasks such as mission planning and execution on the actual Robotic OCU. Fleetwood et al. (2006) report that timing predictions of tasks not involving spatial reasoning generally match preliminary data.

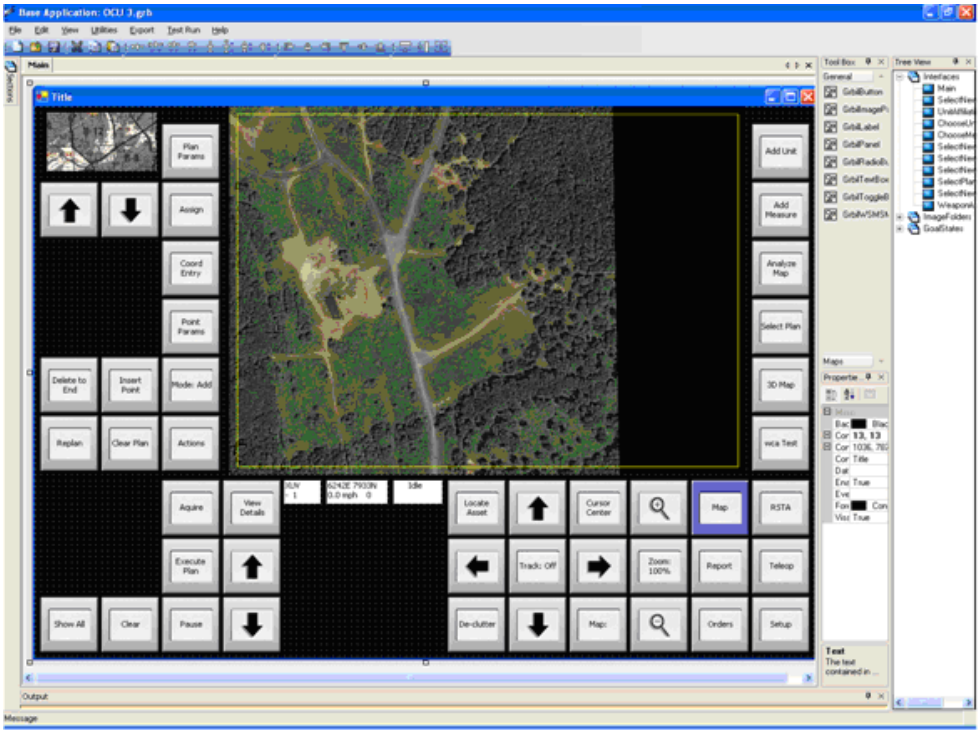

Figure 4: GRBIL tool viewing the Operator Control Unit interface

Related Work

Tools that enable modeling by demonstration, such as CogTool (John et al., 2004), solve some affordability problems. However, they offer no standard solution for spatial reasoning or integration with dynamic environments. As of this writing, visual processing in CogTool is limited to scripted attention movement to interface objects. The environment can not be monitored to detect a change. CogTool simulates the environment with a storyboard of static frames and transitions between these frames triggered by user actions, not a dynamic environment. Under some circumstances, CogTool storyboards could be combined with dynamic simulations (e.g., driving) with some hand-coded modifications.

In addition, CogTool currently produces Keystroke-Level Models (Card, Moran, & Newell, 1983) implemented in ACT-R, while GRBL produces more direct ACT-R code. An example of this difference is that CogTool represents cognitive activity associated with memory retrieval with a 1.2 second mental operator, while GRBIL uses the retrieval mechanism of ACT-R to determine the timing of retrievals.

Conclusion

The project has so far demonstrated that even a complex interface such as an OCU for robotic vehicles can be duplicated by the GRBIL system without writing a line of code. The creation of this interface requires only drag and drop placement of controls and a menu-driven description of functionality. The project has also shown the ability to automatically generate cognitive models that perform dynamic spatial reasoning. The creation of these models requires only the demonstration of procedures and the creation of monitors with a simple interface.

Future Work

The next step in our research is to validate the enhanced GRBIL tool by collecting data on OCU tasks involving spatial reasoning such as projected vehicle intersection. The DRS theory implemented in ACT-R currently uses the default ACT-R timing for movement of attention and object recognition. The validation studies should indicate if further visual processing time is needed to accomplish the DRS routines.

Also, since GRBIL models retrieve procedure steps from memory, it will be possible to use ACT-R's theory of production compilation to predict the trajectory of learning from error-prone retrieval to efficiently compressed performance.

Acknowledgments

We would like to thank the Army Research Laboratory for their funding of this effort and Bonnie John for her comments.

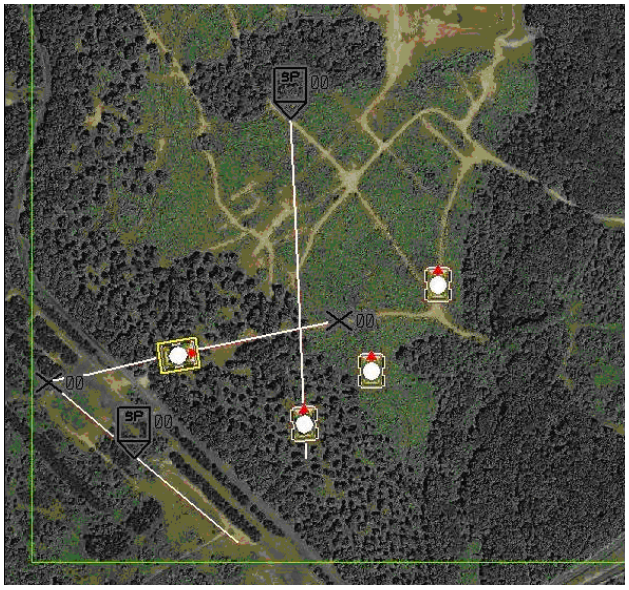

Figure 5. Vehicle path intersection detectable by DRS-enhanced models

References

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. Psychological Review 111, (4). 1036-1060.

Anderson, J. R. & Matessa, M. (1997). A production system theory of serial memory. Psychological Review, 104 (4), 728-748.

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. Human Computer Interaction, 12(4), 439-462.

Archer, S. G. & Adkins, R. "IMPRINT User's Guide" prepared for US Army Research Laboratory, Human Research and Engineering Directorate, April 1999.

Archer, R. D., Lebiere, C., Warwick, W. (2005). Design and Evaluation of Interfaces Using the GRaph-Based Interface Language (GRBIL) Tool. ANSE Human Systems Integration Symposium on "Enhancing Combat Effectiveness Through Warfighter Performance", June 20-22, 2005, Arlington VA.

Byrne, M. D., (2001). ACT-R/PM and menu selection: Applying a cognitive architecture to HCI. International Journal of Human-Computer Studies, 55, 41-84.

Card, S. K., Moran, T. P., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum Associates.

Chandrasekaran, B., & Kurup, U. (2007). Bimodal Cognitive Architectures: Learning and Memory in Spatial Reasoning. ARL Advanced Decision Architectures RMB Meeting, February 21-23. Westminster, CO.

Chandrasekaran, B., Kurup, Banerjee, Josephson, & Winkler (2004). An Architecture for Problem Solving with Diagrams. In Diagrammatic Representation and Inference, A. Blackwell, K. Marriott and A. Shomojima, Eds., Lecture Notes in Artificial Intelligence 2980, Berlin: Springer-Verlag, pp. 151-165.

Fleetwood, M., Lebiere, C., Archer, R., Mui, R., & Gosakan, M. (2006). Putting the Brain in the Box for Human-System Interface Evaluation. Proceedings of the 50th Annual Human Factors and Ergonomics Society Meeting. Santa Monica, CA

Howes, A., Lewis, R. L., Vera, A., & Richardson, J. (2005). Information-Requirements Grammar: A theory of the structure of competence for interaction. In Proceedings of the 27th Annual Meeting of the Cognitive Science Society, 977-983. Hillsdale, NJ: Lawrence Erlbaum.

John, B. E. & Salvucci, D. D. (2005) Multi-Purpose Prototypes for Assessing User Interfaces in Pervasive Computing Systems. IEEE Pervasive Computing 4(4), 27-34.

John, B., Prevas, K., Salvucci, D., & Koedinger, K. (2004) Predictive Human Performance Modeling Made Easy. Proceedings of CHI, 2004 (Vienna, Austria, April 24-29, 2004) ACM, New York.

Larkin, J.H. & Simon, H.A. (1987). Why a diagram is (sometimes) worth ten thousand words. Cognitive Science, 11, 65-99.

Lathrop, S., and Laird, J.E. (2006). Incorporating Visual Imagery into a Cognitive Architecture: An Initial Theory, Design and Implementation. University of Michigan Technical Report CCA-TR-2006-01.

Matessa, M. & Brockett, A. (2007). Using a Diagram Reasoning System with ACT-R. 16th Conference on Behavior Representation in Modeling and Simulation.

Matessa, M., & Polson, P. (2006). List Models of Procedure Learning. In Proceedings of the International Conference on Human-Computer Interaction in Aeronautics (HCI-Aero), San Francisco, CA.

Salvucci, D.D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In Human Factors in Computing Systems: CHI 2003 Conference Proceedings (pp. 265-272). New York: ACM Press.

St. Amant, R., & Ritter, F. E. (2005). Specifying ACT-R models of user interaction with a GOMS language. Cognitive Systems Research. 6(1) 71-88.