Report 82-39
Stanford -- KSL

Scientific DataLink

The Interviewer/Reasoner Model: An Approach to Improving System Responsiveness in Interactive AI Systems. Phillip E. Gerring, Edward H. Shortliffe, et al., Nov 1980

Heuristic Programming Project
Report No. HPP 82-39

November 1980

The Interviewer/Reasoner Model: An Approach to Improving System Responsiveness in Interactive AI Systems

Phillip E. Gerring, Edward H. Shortliffe,
and William van Melle

Published in the AI Magazine, Volume 3, Number 4. The AI Magazine is published by the American Association for Artificial Intelligence.

The Interviewer/Reasoner Model: An Approach to Improving System Responsiveness in Interactive AI Systems

Phillip E. Gerring
Teknowledge Inc.

Edward H. Shortliffe
Stanford University

William van Melle
Xerox Palo Alto Research Center

Abstract

Interactive intelligent systems often suffer from a basic conflict between their computationally intensive nature and the need for responsiveness to a user. This paper introduces the Interviewer/Reasoner model, which helps to reduce this conflict. This model partitions an intelligent system into two asynchronous components. The Interviewer's primary function is to gather data while providing an acceptable response time to the user. The Reasoner does most of the symbolic computation for the system. This paper describes the implementation of the model in both timesharing and personal workstation environments, and uses the ONCOCIN system as an example.

1. Introduction

Interactive intelligent systems often suffer from a basic conflict between their computationally intensive nature and the need for responsiveness to a user. An acceptable response time is needed both during system testing and to help insure end-user acceptability. During the normal course of development of an AI system there is substantial testing on real problems under the guidance of human experts whose time is usually valuable. Moreover, many end users (e.g., physicians) will simply refuse to use a system if they have to wait for a response. Both these considerations place a premium on responsiveness. Unfortunately, many AI systems run so slowly that waiting appears to be unavoidable.

This paper introduces the Interviewer/Reasoner model, a technique which can help to reduce this conflict under the conditions specified below. As is described in Section 2, the model partitions an AI system into two components which run asynchronously, one to handle the interactions with the user and the other to perform the symbolic computations. The Interviewer/Reasoner model has been implemented in ONCOCIN (Shortliffe, et al., 1981), an expert system that is designed to assist physicians in the treatment of cancer patients. ONCOCIN was developed at Stanford University on the SUMEX computer facility. Section 3 describes how the model has been implemented for ONCOCIN in a timesharing environment, and Section 4 discusses two possible implementations on personal workstations. Other application areas and further extensions to the model are considered in Section 5.

The work described in this paper was carried out at Stanford University and was partly supported by the National Library of Medicine under program project grant LM-00395.
The original idea for splitting the tasks of information gathering from reasoning in order to improve system response time was suggested by Ted Shortliffe and Chuck Clanton for the ONCOCIN project. Thanks are due to Eric Schoen and Bill van Melle for help with the implementation, to Mark Stefik and Harold Brown for help in writing this paper, and to the rest of the ONCOCIN project members, including Carli Scott, Miriam Bischoff, Charlotte Jacobs, and Craig Tovey. Further information, including implementation details, is available by writing Miriam Bischoff, The Heuristic Programming Project, Room TC-117, Stanford University School of Medicine, Stanford, California 94305.

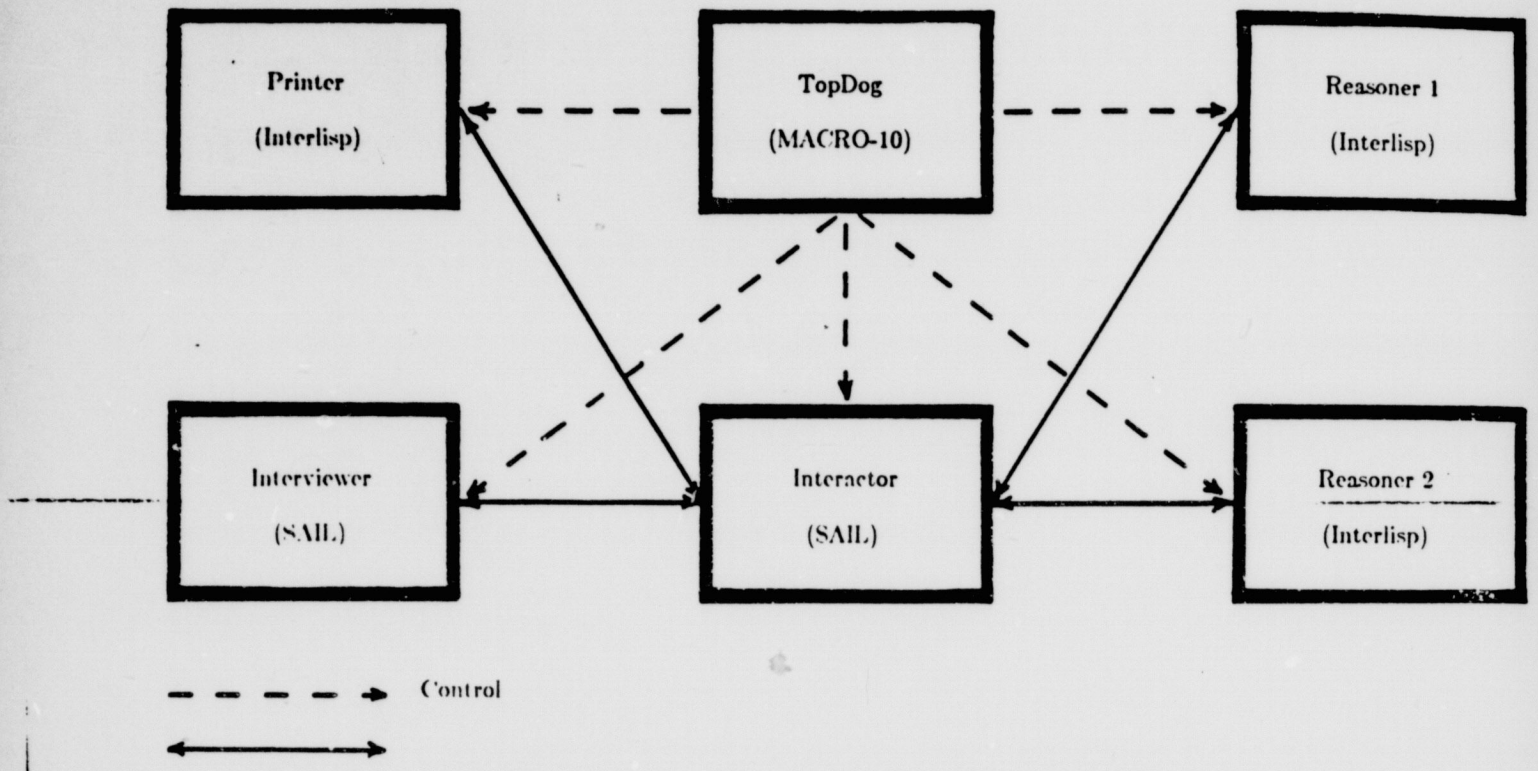

Figure 1. The Interviewer/Reasoner model. The Interviewer interacts with the user, buffering and ordering data. The Reasoner performs the intelligent system's symbolic computation.

2. The Interviewer/Reasoner Model

The Interviewer/Reasoner model partitions an intelligent system into two components: an Interviewer, which mediates interactions between the user and the system, and a Reasoner, which does most of the symbolic computation for the system (Fig. 1). The Interviewer's primary function is to gather data. It should ideally never cause the user to wait, and must operate independently of the reasoning process. The Reasoner must run in the background, essentially invisible to the user. If it needs additional data not normally gathered by the Interviewer, it can interrupt the user, via the Interviewer, to ask for those data. To minimize delay before its results are available to the user, the Reasoner should start using data as soon as they are available.

These characteristics constrain both the nature of the Reasoner and the kind of domain for which the model is appropriate. In order to work effectively as a background process, the Reasoner should be primarily data-driven. A goal-directed system usually needs specific pieces of information at unpredictable times. A goal-directed Reasoner would therefore need to direct the information-gathering process, and the Interviewer/Reasoner model would gain nothing. For the model to be useful, the Interviewer must be able to gather information independently of the reasoning process. For example, this is possible for consulting systems which form conclusions from relatively standard data sets as they are entered.

In order for the model to work most effectively—i.e., to maximally overlap interviewing and reasoning—neither the Interviewer nor the Reasoner should have to wait long for the other. The Reasoner must be able to use the data in whatever order they arrive, or it must be possible for the Interviewer to obtain early any data that are critical to the Reasoner's processing. For consulting problems, the critical data questions can often be asked at the beginning of the session.

ONCOCIN follows the latter strategy. After performing a history and physical examination, the physician uses a video display terminal to fill out a patient's flow sheet. The flow sheet is a list of patient "parameters" (blood counts, lab tests, etc.) for which the physician supplies values. If the Reasoner needs information not on the flow sheet, it can interrupt and ask the physician for that information. Because an actual interruption can be annoying, ONCOCIN handles this situation by simply inserting the new data queries at a convenient place on the flowsheet that the physician is filling out. After the physician supplies all the

Figure 2. The ONCOCIN system. The TopDog process monitor controls the other system components. The Interactor handles all interprocess communication. The Printer controls a lineprinter used for generating patient flow sheets, summary sheets, therapy recommendations, etc. The Interviewer interacts with physicians via a display terminal with a special keyboard. The Reasoners interpret patient data and make therapy recommendations; one finishes up one case while the scond starts another.

values, the Reasoner makes therapy recommendations. The most useful parameters, from the Reasoner's point of view (i.e., those which will allow it to begin significant processing immediately), are positioned at the top of the form. If they were scattered throughout the form or near the end, the physician might have to wait an unacceptably long time before the recommendations were ready. This is a problem with goal-directed systems such as MYCIN (Shortliffe, 1976), where the physician must provide an answer, wait, provide another, wait again, etc., until the program reaches a conclusion.

3. Implementation for a Timesharing Environment

On a large timesharing machine, a straightforward implementation of the Interviewer/Reasoner split is to run each as a separate process and provide some means of communication between them. This is how the current prototype version of ONCOCIN is implemented.

ONCOCIN uses two support programs (Fig. 2). A multiple process monitor, familiarly called TopDog, creates the processes for the Interviewer and the Reasoner and sets them in motion. The Interviewer and the Reasoner communicate by passing messages to each other via an additional process, called the Interactor. The Interactor provides a "mail service;" each process has an "in-box" and an "out-box." The Interactor picks up messages from the out-boxes, queues them if necessary, and delivers them one at a time to the appropriate in-boxes.

Messages have a priority level. Those with higher priority will be delivered first; otherwise any queueing is simply first-in-first-out. Priority is often important. For example, if ONCOCIN's Reasoner needs to interrupt with a question, that datum requires a higher priority than other flow sheet values (of which there may already be a queue). Alternatively, an unimplemented but useful feature would be to allow the physician to ask questions of the form "What was the patient's white blood cell count the last time the toxicity was this severe?" Here the user would expect to get an immediate answer; giving the more important messages higher priority allows this to be done.

An advantage of this multiple process approach is the ablility to implement the Interviewer and the Reasoner in different languages, each suited to its purpose. ONCOCIN's Interviewer is coded in SAIL (Reiser, 1976), which provides high-speed interactive I/O capabilities, while the Reasoner is coded in Interlisp (Teitelman, 1978), which provides symbol manipulation capabilities.

The prototype ONCOCIN system was installed for experimental use in the oncology clinic at the Stanford Medical Center in May 1981. During clinical use, the system runs on a dedicated timeshared DECSystem-2020. Physicians interact with the Interviewer via a video display terminal with a special keyboard. System response time is acceptable, with

a typical interaction lasting approximately five minutes. Because only one terminal is available on the current system, only one physician may use the system at a time; the ONCOCIN project hopes to provide more readily available access to the consultation programs by transferring the system to a network of personal workstations.

4. Implementations for a Personal Workstation Environment

Several manufacturers have recently introduced high-performance personal Lisp workstations. These machines have high-resolution graphics and a larger address space than is available on the timesharing systems most commonly used for AI systems. They permit a scaling of ambitions with respect to both user interfaces and complexity of computation. This emphasizes, rather than alleviates, the need for the Interviewer/Reasoner model when implementing interactive intelligent systems: fancier interfaces, especially using graphics, require more resources, whereas less stringent address space limitations permit larger systems. This scaling of ambitions can very well lead to ever-slower system response.

The implementation for a personal workstation environment that is most analogous to the timesharing one, if the workstation supports multiple processes, is to run the Interviewer and the Reasoner as separate processes on a single machine. The ONCOCIN project is exploring this approach as multiple process software becomes available. An advantage of this over a timesharing environment is that communication is considerably simplified if the processes run in the same address space (as is the case with most of the available workstations). However, since some workstations have less computational power than large machines, this approach has the possible danger of exceeding the capabilities of a single machine; this may make it unacceptable for some problems.

An alternative implementation is to run the Interviewer on one workstation, the Reasoner on another, and arrange for communication via a network (some workstations have multiple processors which can function separately; these are included under this category). The primary advantage of this approach is that the two processes do not contend for a single machine's resources, thus increasing the computational power available to the system. Another advantage for system developers is that the Reasoner can maintain its own display of its inner workings; a workstation's graphics capabilities permit more transparent high-level monitoring of complex computational systems (Model, 1979). The knowledge engineer can use this separate display to monitor and debug the system unobtrusively (i.e., without having to preempt the expert's terminal) in a real-time manner. Its major disadvantage is economic, in that it requires more than one workstation. This is not necessarily a disadvantage, since two workstations are expected to cost much less than even a part of a large timesharing system.

5. Other Applications and Extensions

Another kind of domain where the Interviewer/Reasoner model would be appropriate is for real-time signal processing problems. The Interviewer would not interact with a human user but would monitor signal sources where the real-time considerations demand greater responsiveness to the input data than the Reasoner could supply. The Interviewer could also order the data, passing critical information to the Reasoner first.

There are several intriguing possible extensions to the Interviewer/Reasoner model. First, one can imagine having multiple Interviewers communicating with a single Reasoner. This would be useful for systems which have several informants. Second, there could be a single Interviewer and multiple Reasoners. This can be useful in several settings:

1. Common interface for multiple systems: Each Reasoner can be a different intelligent system. The Interviewer is then a common interface between the user and the several Reasoners, thus freeing the user from learning several different protocols.
2. Concealing asynchronicity: The Reasoners can be pieces of the same system (either a system broken into pieces by functionality or size, or a distributed system). The Interviewer then presents a unified interface, hiding the fact that the user is interacting with several asynchronous programs at once.
3. Multiple problems at once: Two or more Reasoners can perform the same intelligent task, but can be working on separate problems (cases). For example, one Reasoner can be finishing up one problem while another starts a new case. This is done in ONCOCIN to avoid waiting between patients; a new Reasoner can begin on a new case immediately while the previous Reasoner performs file cleanup functions necessary at the end of each consultation session.

Third, there could be multiple Interviewers and multiple Reasoners. This brings to mind powerful combinations of the first two extensions; for example, multiple informants to a distributed intelligent system.

References

Model, Mitchell L., Monitoring System Behavior In a Complex Computational Environment, Report No. CSL–79–1, XEROX Palo Alto Research Center, Palo Alto, Calif., January, 1979.

Reiser, John F., ed., SAIL, Stanford Artificial Intelligence Laboratory Memo AIM–289, Dept. of Computer Science, Stanford University, Stanford, Calif., August, 1976.

Shortliffe, E. H., *Computer-Based Medical Consultations: MYCIN*, Elsevier/North Holland, New York, 1976.

Shortliffe, E. H., Scott, A. C., Bischoff, M. B., Campbell, A. B., van Melle, W., and Jacobs, C. D., ONCOCIN: An expert system for oncology protocol management, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vol. II, pp. 876–881, 1981.

Teitelman, Warren, Interlisp Reference Manual, XEROX Palo Alto Research Center, Palo Alto, Calif., 1978.