Developments in Business Simulation & Experiential Exercises, Volume 15, 1988

COMPUTER SIMULATION OF HUMAN INTERACTION

John D. Bigelow, Boise State University

ABSTRACT

A computer emulation of human interaction could be of great value in the process of learning interpersonal skills. However, the programs developed to this point are still far from capturing the complexity of actual interactions. This paper explores the nature of interpersonal interactions and the issues which need resolution in developing computer emulations of human interaction. A model of human interaction is presented. Following this, ways of managing the complexity of human interaction in a computer emulation are considered. Finally, Three approaches to developing computer emulations of human interactions are discussed.

INTRODUCTION

One kind of learning objective in many Organizational Behavior courses involves learning to effectively manage an ongoing interaction with another so as to accomplish something. For example, a person may want to communicate skillfully, resolve conflict, delegate a task, or influence another. The means by which this kind of learning goal is accomplished is usually person-intensive: after doing the requisite reading, individuals analyze cases and discuss them, watch modelling tapes, and may ultimately engage in live role plays.

Rather than moving directly to interpersonal dialogues as a way of learning interaction skills, it would be of great benefit if a person could first carry out one or more dialogues in which the other person is modeled by a computer (Kryuov and Kyukova, 1986) The computer could provide a patient tutor and consistent coaching. While a program would not replace actual role playing, it could help a person to practice using different approaches and to get a feel for how the different lines of action might go. Thus, an appropriately designed computer program could save instructor and class time in learning human interaction skills.

The number of programs which address human interaction is growing. However, to the author's knowledge, there are no programs yet available which are anywhere near providing a realistic ongoing interaction. The author has been developing computer software for management learning for several years, and recently spent a Summer attempting to create a program which emulated human interaction - - with very limited success. In doing so, the author gained some insight into the nature of the problems confronting the prospective author of such a program.

The purpose of this paper is to explore the nature of interpersonal interactions and the issues which need resolution in developing computer emulations of human interaction. This paper starts by presenting a model of human interaction. Following this, ways of managing the complexity of human interaction in a computer emulation are considered. Finally, Three approaches to developing computer emulations of human interactions are discussed.

MAPPING EPISODIC SPACE

The basic unit of human dialogue is the "episode" or a series of interchanges between two people. An episode begins when two people who have goals which are dependent on the other begin interacting. The dialogue proceeds as the two engage in a series of interchanges (alternations between speaking and listening) in which both attempt to meet their goals. The episode terminates when interchange ceases, for whatever reason (e.g., goal attainment, giving up, or interruption). Episodes may consist of 1-2 interchanges (as in greetings), or may involve a considerable number of interchanges -- perhaps 200-300. Typically a skill building episode will involve 30-80 interchanges.

An essential quality of an interaction episode is that it is progressive: i.e., it is likely that one or both parties will undergo a number of changes as a result of the ongoing interaction. Each person may try a series of tactics designed to accomplish his/her goals. Each may try a tactic, conclude it won't work, and try another. Each may revise his/her sense of what is wanted from the episode. Each may begin to view the situation differently, perceive that the episode is entering another stage, or change his/her emotional state.

The progressive quality of an episode means that the impact of a given interchange will probably vary, depending on the point during the episode that it is introduced. For example, if a person repeatedly resists a compromise and then gives in, the other person will probably feel more impatient and less satisfied than if the person had compromised at the outset. Again, a "hello" at the start of an interaction will probably elicit a response in kind. A "hello" at later stages will probably elicit puzzlement.

Operationally this means that it is not enough to simply have a repertoire of actions to be skilled in human interaction. It is also important to know "timing": to be able to identify the points in a dialogue where actions are most likely to have a desired effect, and how to move the dialogue to these points.

During each interchange of an episode, each party is likely to be making choices about what to do next. While people engaged in an episode will probably not consciously enumerate their choices, it's usually possible to think of about 3-10 fundamental alternatives at each point during an episode.

If we observe an actual episode, we only see the results of peoples' interchange choices. We do not see what would have happened if either had made different choices. A learning model needs to include all the choices students might plausibly make--even if those choices might not be considered effective--so as to allow them to learn from making those choices -

Let us call the set of all possible interchanges the "4-D space" of the episode, so as to distinguish it from the "3-D" path that an actual episode takes

through this space. Let us consider how we might portray this 4-D space.

Perhaps the best place to start is at the episode's starting point. Here no interchange has yet taken place, no choices have been made, and it is possible to describe the situation before it is changed by progress into the episode. The episode begins with the first interchange. If we assume that a person has more than one response to choose from during each interchange, we see that as we progress by one interchange, the number of possible interchanges increases by the product of the number of possible choices. Visually, the episode appears as a "tree", starting with a single interchange, and branching at each interchange. If we term the "size" of an episode the number of possible interchanges in the 4-D space, it is possible to calculate size using the following formula:

$$S = \sum_{i=1}^{n} i * c \qquad (1)$$

where S - size, c - the number of choices an individual has in choosing an interchange, and n - the number of interchanges in a 3-D episode.

This equation is based on two assumptions: 1) for each interchange an individual has the same number of choices and 2) all 3-D paths through the 4-D space involve the same number of interchanges. In actuality, the numbers of choices and interchanges will both probably vary widely for given interchanges and paths. Therefore, this equation is only used to provide a qualitative sense of the nature of the 4-D space.

If we consider as an example an episode involving five interchanges, in which a person can make five choices during each interchange, the number of possible interchanges in the 4-D space is 1 + 5 + 25 + 125 + 625, or 781 possible interchanges.

MANAGING COMPLEXITY IN EMULATING INTERACTION

The discussion above suggests that as the number of choices and interchanges in an episode increase, the total number of possible interchanges increases rapidly. In a normal episode (such as a conflict management role play) , the number of possible interchanges could plausibly be placed in the thousands or millions. Obviously, some means of managing the complexity of such a space must be used. Let us consider some of the ways this can be done.

Generalize the Episode to One Interchange

One way of reducing the complexity of the emulated episode is to simply identify one or more key choices which must be made at the start of an episode; e.g., the choice of a conflict resolution style, an influence approach. or a motivation technique. At the same time more detailed advice may be offered for possible use during the episode. There are a number of existing computer programs which take this approach:

Vroom-Yetton Decision Model (Computerized by Tim Schweitzer in compiled BASIC) The person answers a number of questions about the nature of a decision, and is provided advice about the appropriate level of group participation to elicit in making the decision.

Motivation (By Dave Whetten in the Ten-Core Authoring System) . It describes a situation and asks which motivational model should be applied. Feedback is given for correct and incorrect choices.

Management Readiness Tutor (By John Bigelow in Modula-2). It asks a series of questions to estimate a person's readiness to manage. it provides a report of results and suggestions for development.

Group Diagnostics (Roger Volkema in compiled BASIC). It asks a series of questions about a problematic group and provides a diagnosis and suggested remedies.

Decision-Making (By Tom Janz in compiled BASIC). It provides a hypothetical decision dialogue between two people. The user is asked to recognize the presence, absence, and appropriateness of five decision steps. Feedback as to correctness and incorrectness is given, along with a score.

Communication Edge (One of a series by the Human Edge Corporation). It elicits detailed information about the person and another, and provides specific advice about how to communicate well with the other.

The above examples range in the amount of specific advice offered, roughly in the order they are presented. For example, the Vroom-Yetton program simply identifies a participation approach and says nothing about how to implement it. On the other hand. the Human Edge program provides several pages of specific advice.

Include only Key or Archetypal interchanges

Another way to reduce the complexity of the emulated episode is to emulate only a few key interchanges. It may be possible to identify sets of interchanges which can be clustered together as predictable phases of an episode; e.g., opening, rounds of negotiation, and closing. One may then provide advice as to how to handle each of these phases, depending on choices made in earlier phases. The Executive Suite (By Armonk Corporation) provides an example of this strategy. This is a game in which a person enters a corporation and is confronted with a series of situational and job choices. The goal is to attain the executive suite, but much is learned about organizational power and politics in the process. This program provides only critical situations in which one's choices have career and political consequences. Further, it requires only one response (from a multiple choice set) at the outset of each situation. Thus, it makes manageable the enormous potential complexity of political interchanges in an organizational setting.

Reduce the Number of Choices

It is also possible to reduce the complexity of the emulated episode by reducing the number of choices allowed. The interactive cases developed by Dennis Moberg and David Caldwell (in compiled BASIC) provide an example of this strategy. Their programs consist of a set of cases designed to facilitate teaching of diagnostic arid application skills. Students are provided case information, make decisions, and learn the effects of their decisions as the case unfolds. If a student makes what is considered to be an unwise choice, he/she will simply be advised as to why it was inadvisable, and be returned to a mainline path. In this way the inadvisable paths are truncated, reducing the size of the emulated episodic space.

Create an Interchange Generator

The approaches discussed to this point all map parts of the 4-D space, and allow users to create 3-D paths through it. An alternative to prior identification and creation of all possible interchanges is to create an interchange generator, which creates the situations based on the interchanges to that point. For example, one might emulate a conflict episode by creating a computerized model of an other. The user's inputs would impact on this model, leading to responses by an emulated person.

This approach has some distinct advantages. By creating interchanges in situation can create an interactive program which is capable of an indefinite number of interchanges in a very large 4-D space. In addition, one could change the starting situation or the "personality" of the other without having to modify the various interchanges of the 4-D space one by one.

Eliza (By Steve Grumette in interpreted BASIC) is an example, albeit a rudimentary one, of a program which interacts with the user in natural language. Eliza mimics a therapist and begins by printing to the screen, "HOW DO YOU DO, PLEASE TELL ME YOUR PROBLEM." Eliza interacts with the user by using simple algorithms to reflect back the user's statements in a quasi-theraputical format.

Two difficult issues must be confronted before such an emulation could be developed. The first concerns the creation of the model. One would have to think of the parameters and variables of a person and how they are affected by interchanges. Undoubtedly a simplified model of the other would be necessary, which is geared toward a specific type of interchange. Even so, the model may turn out to be fairly complex.

The second issue concerns how to create an interface between the model and the person interacting with the program - - both interpreting the user's inputs, and formulating outputs to the user. With reference to inputs, It's quite difficult to create a program which can interpret natural language statements. It might be possible to make the input side more manageable by requiring that the user use a simplified and more structured language. This input would then be analyzed by a 'parser" subroutine which would break down the inputs and assign them values which could be used by the program. Such an approach is commonly used in "text adventures" such as those which can be created by world Builder, where the user is confronted with a setting and some puzzle to be solved. The user moves and acts in the setting created by the program through 2-3 word sentences, such as "go north", or pick up book", or "read sign".

with reference to language outputs from the program to the person, this may be a somewhat more manageable task. A library of partial statements and sentence frames could be created and drawn on in formulating responses. For example. the Human Edge programs above contain extensive text files which are drawn on in formulating advice.

DEVELOPMENT APPROACHES

A person wishing to develop a computer emulation of an episode for use in interpersonal skill development may use a number of approaches. Three appear to be in widespread use: authoring systems, expert systems, and simulations. Each is discussed and critiqued in this section.

Authoring Systems

A number of authoring systems are available, all of which work on much the same principle: The developer designs a number of computer screens which present text and pictures, and may be accompanied with sound (music and speech), video clips, or video stills. The user is presented with a series of choices and indicates a choice by typing it via keyboard, or possibly by indicating a region of the screen (using touch or a mouse). The choice leads to another screen, etc. , until the person or the program terminates the session.

An authoring system can be used to emulate an episode. Its main advantage is that it is easy to interface with the user, since the computer's outputs are fixed for each situation, and users' inputs are defined by the program. There are least four disadvantages to an authoring system approach:

1. Large numbers of screens must be generated. Since a screen must be created for each interchange, the number of screens which must be generated can escalate to unmanageable numbers before a reasonable emulation is accomplished.
2. Choices are predefined. The user's choices are predefined by the program. This means that the developer must be careful to consider likely preferences by users when developing the program, or users will feel that their choice is not represented. A greater problem is the "priming effect of defining the choices the user may makes By doing so, the program limits the students' learning to identify these choices by him/herself, and encourages a "multiple choice orientation to moving through the episode.
3. They are inflexible. Authoring systems are much like a novel, in that initial premises are defined, and all else follows from them. If one wants to change the situation or the personality of the other, most or all of the screens must be modified - - one by one.
4. They are not (yet) developer-oriented. The authoring systems with which the author is familiar with are not well geared to efficient development. They tend to be clumsy to use and difficult to debug. Since the most important resource drawn on in developing an interchange emulation is the limited time of the developer, this is a very serious shortcoming. What is needed is an authoring system which is easy to learn, moves smoothly between frame generation and debugging, allows for modular programs (i.e., combining separately developed units together), and provides some way of easily moving through the large tree structure characteristic of interpersonal episodes.

Expert Systems

Another tool for the prospective developer is an expert system shell. These are programs which allow one to emulate an expert decision maker. Rules used for making the decision are entered, and the program is started. It then asks the user for information needed to make the expert decision. When the program has obtained enough information, it provides the decision.

An expert system can be used to provide a "one interchange" emulation of an episode. For example, the author has programmed the Vroom-Yetton decision

tree into an expert system. Any of the "one interchange" emulations discussed earlier could be implemented using an expert system shell.

The expert system approach provides a lot of potential in developing interchange emulations. It is possible to get expert system shells which are easy to program and easy to debug. However, current expert system shells are oriented to providing a single, though possibly complex, decision, They are less well adapted to providing much supporting information or to providing a series of decisions in an evolving situation. Since expert system technology is simply a programming approach, there is no reason why it couldn't be adapted to these ends. In addition, an expert system shell could contribute to developing the model for an interchange generator.

Simulations

Finally, one can use a simulation approach in emulating human interaction. Essentially one would develop a computer model of another and a user interface, through which the model interacts with the learner. Although a number of organizational simulations have been developed and are being used in university courses, no interpersonal simulations are yet correspondingly in use. Probably the reason is that organizations are easier to model (being both simpler and more quantitative), and that interfaces with organizational models are easier (human language is not needed)

To the author's knowledge there are no software programs available which would ease the task of emulating human interaction. The developer using this approach will probably use a computer language, such as BASIC, Modula-2, or LISP. Each of these (and other) languages has distinct advantages and disadvantages, and there is no easy way to determine which is best.

A simulation of human interaction overcomes a number of disadvantages found in other approaches. It could provide a wide variety of choices to the user and episodes of indefinite length. It would be relatively easy to modify so as to create other situations and different kinds of personality profiles.

However, a simulation would require developing a model of a person - - not an easy thing to do. In addition it would require developing some way of interacting with the model - - presumably through natural language or a language subset,

CONCLUSION

In this paper the nature of human interaction and issues in developing a computer emulation of human interaction have been explored. It is seen that actual interaction episodes of any length are potentially quite complex. The emulation developer is faced with two choices. First, he/she may choose to use an authoring system and/or expert system to implement the emulation. However, ft will then be inevitably necessary to simplify the emulated episode by restricting the number of interactions and/or the number of choices the user may make.

Second, the developer may choose to use a simulation approach. While this may eventually result in an emulation which overcomes some of the limitations of the first approach, the developer must identify a suitable programming language, and develop both a model of the other and an interface with the model.

Which of these approaches is most promising? The first approach requires significant simplification of situations. Still, it can be used to provide valuable learning in a broad interactional format, and is more do-able. The second approach requires a lot of development, and we won't see any implementations soon. Still, a simulation of human interaction, if done well, will be something to get excited about.

REFERENCES

BUCKLEY, Walter (1967). Sociology and Modem Systems Theory. Englewood Cliffs, New Jersey: Prentice-Hall.

KRYUOV and KYUKOVA (1986). "Towards a Simulation Games Classification and Dialogue Types," Simulations and Games 17(3) September: 393-402.