

A co-operative computer based on the principles of human co-operation

A. A. CLARKE AND M. G. G. SMYTH

LUTCHI Research Centre, Loughborough University of Technology, Loughborough LE11 3TU, UK

Co-operation is presented as a technique for radically improving human-computer interaction with complex knowledge bases during problem-identifying and problem-solving tasks. A study of human-human co-operation literature indicated the importance of creating an environment where the refinement of solutions can be based on argument and the resolution of differing viewpoints, as it is through this interaction that the nature of the problem is revealed. To bring about such an environment, the work identified and created three mechanisms now considered to be central to human-computer co-operation; goal-oriented working (GOW), an agreed definition knowledge base (ADKB), and a model which, using problem-domain rules, stimulates the interaction between the user and the machine: the partner model (PM). To identify the requirements of the co-operative machine more completely, a software exemplar was constructed, using the task metaphor of spatial design. The result of the work is the implementation of a machine software architecture which demonstrates the functioning of co-operation. This co-operative computer, its evaluators believe, supports a user-machine interaction having a totally new and different quality. The machine architecture and software tools and techniques developed in the work can form the foundation for building future co-operative systems.

Introduction

Co-operation is a group behaviour that can have distinct, even critical, advantages in problem identification and problem solving. It was therefore considered important to explore the potential of co-operation in human-computer operations. The modern computer, with its present architectures and ways of interfacing with the user, is intended to be a helpful tool, but a tool that is merely helpful cannot be called co-operative. A new architecture and interfacing was required before human-computer co-operation could be realized. In principle, co-operative techniques could be applied to many functionalities of the computer; for example, to the human-computer interface, resource management, knowledge acquisition or knowledge representation. This paper starts with a discussion of co-operation and related issues, then moves to a description of the (necessarily limited) use of the principles of co-operation in the successful design and implementation of an exemplar co-operative computer, and its application in the domain of spatial design.

Co-operation

Animal studies suggest that the behavior called "co-operation" is primarily associated with problem solving and the external environment. Young (1979)

proposed that human co-operative behaviour extends this association to include the internal, psychosocial environment. He also observed that co-operation is a product of the process of evolution: "the [physical and mental] equipment Man receives from his past history is especially designed to ensure communication and co-operation."

Arguably, the most important advantage of co-operation in general is the synergy arising in the performance of certain tasks. This effect most noticeably occurs when the combined action of two or more individuals realises net benefits that are more than twice the benefits available to a single individual (Rothstein & Pierotti, 1988). The authors of this present paper hypothesized that it would be of great benefit if the potential of the synergy of human co-operation could be achieved in human-computer working.

DEFINITION OF CO-OPERATION

The term "co-operation" is used by people operating different universes of discourse, and because of this, has acquired a variety of meanings. The definition "co-operation is the situation where the movement of one member towards the goal will to some extent facilitate the movement of other members towards the goal" (Deutsch, 1962) is widely referred to, and was used in shaping the philosophy of the work in this paper.

THE ELEMENTS OF CO-OPERATION

Co-operation is a set of relations among behaviours and their consequences, i.e. it is neither a simple behaviour nor a specific pattern of behaviours. Five elements defining the content of co-operative relations have been identified by Marwell and Schmitt (1975), namely: goal directed behaviour; a reward system, with rewards for each party; distributed responses; co-ordination; and social co-ordination. From Deutsch (1962) it can be seen that there must be *one agreed, common goal* towards which the parties are working. *Reward* may be found in the fact of having completed the task, or in the final nature of the physical subject of the task, or in the satisfaction of the striving for achievement of the goal. *Responses* must be distributed among the parties for co-operation to occur; the responses can be conjunctive (where all parties must make correct responses) or disjunctive (where the necessary response could come from any one party). *Norms* are legitimate, socially shared standards or guidelines for the accepted and expected pattern of conduct by the parties in co-operation (Birenbaum & Sagarin, 1976). They may be formalized, and external to the task (e.g. as in written laws) or they may arise out of the task itself. If the norms or standards of any particular situation or set of conditions are breached, then co-operation may cease. *Co-ordination*, in general, requires the presence of cues to synchronize activities, and they can be mechanical or social cues. The latter are important enough to be counted as a separate element because they hinge on the need for all parties to be present, and be able to see and be seen by all, and on the existence of norms. Co-ordination can be task- or time-related. Two other aspects of *time* must also be taken into account, although they are not specifically itemized by Marwell and Schmitt (1975). These are (i) the length of the period of time during which the onset of responses by parties must

occur for the responses to be effective (i.e. the latency of the co-operative response): if any party does not respond in time, then co-operation will be jeopardized, and (ii) the length of time required for each component response to be completed; the parties must spend enough time on the responses.

Elements may be prominent in different ways in different types of co-operation. For example, all five elements plus timing are clearly seen in the tightly-knit co-operative working of a small group such as a surgical theatre team. "Intermediate" co-operation may be taking place when there is no social co-ordination, as may be seen in some manufacturing activities. Finally, co-operation (albeit of a very meagre sort) may be taking place where only the first two elements (goal seeking and rewards) are prominent; for instance, where participating shareholders in some enterprise share the common goal of wanting more money, and take the rewards of more money as and when it occurs.

TWO FORMS OF NON-CO-OPERATIVE POWER

It has been stated above that the term "co-operation" is used in many ways. Two well-studied forms of group behaviour which represent this are Tit-For-Tat (TFT) and the Prisoners' Dilemma Game (PDG). TFT is described as "a strategy that can be employed in a game theory to elicit stable co-operation" (Axelrod & Hamilton, 1981). It is a strategy based on the other person's action; if one person acts co-operatively, then the other co-operates. If he or she acts competitively, then the other also competes. TFT is different from the situation where two or more people are co-operating so as to reach a mutual goal, and so cannot be called co-operation. PDG is useful for studying group behaviour and decision theories in an abstract sense, but it does not accord well with co-operation. The existence of a common goal is an ill-defined motivator for the two prisoners; rewards can be lacking for one; there is no opportunity for communication about any common goal (for the prisoners, communication is the response: no strategic planning can take place). We conclude that co-operation in the sense used in this paper does not embrace TFT and PDG.

ADVANTAGES OF CO-OPERATION IN PROBLEM SOLVING

To establish a performance base-line, the authors examined some of the advantages of co-operation in human-human problem solving. Some of the more notable findings include the following features. Joint effort during complex problem-solving tasks is reflected in both the process of achievement and the quality of solution (Deutsch, 1949, 1968; Laughlin *et al.*, 1968). In studies of group performance, of which co-operative behaviour is an example, Davis (1969) found that if each person possesses unique but relevant information, and the task requires several pieces of information, then the pooling of this information will allow groups potentially to solve problems that an individual cannot attack successfully. He also observed that if the emphasis is put on achieving a correct or good or early answer, then a group has a higher probability of achieving this aim (other things being equal) than does an individual. Furthermore, group problem solving can often yield more alternative solutions than can individual problem solving; this finding is very important to the application of co-operative principles in human-computer co-operation.

Co-operation can also create new motives, attitudes, values and capabilities in the co-operating parties. Dynamic social interaction during complex problem solving has the potential to reinforce or negate existing beliefs and to support the formation of new attitudes between and within the parties. It is this potential for mutual growth that makes co-operative behaviour so satisfying to the partners in human problem solving.

FACTORS WHICH INDUCE AND MAINTAIN HUMAN CO-OPERATION

Human co-operation can arise because there are problems to be solved (i.e. goals to be reached), or because there is satisfaction in working together, or both. People can co-operate either in reaching goals which any individual in the group could achieve independently, or in reaching goals which are beyond any one individual. In the first case, group synergy is not necessarily noted in faster task achievement, less expenditure of effort by individuals or in an increased rate of goal completions, but more likely in an increase in group cohesion, stronger common bonds and greater group identity. The non-social reward in such a situation is likely to be external to the task, and could take the form of material reward, e.g. money or the promise of reciprocal help later.

The maintenance of co-operation is also at risk in the first case because, for example, one party may become satisfied with the accumulated reward at some stage, and leave the group. In the second case, a considerably more stable form of co-operation can occur. This is the situation where superordinate goals are present and where the reward is inherent in the task. Superordinate goals are compelling for the individuals involved, but cannot be achieved by one individual through his or her own efforts (Sherif & Sherif, 1953; Blake & Mouton, 1962; Blake, Shepard & Mouton, 1964). The existence of superordinate goals in a variety of task situations is one of the strongest factors in the development and maintenance of co-operation, both between individuals and groups.

COMMUNICATION

Co-operation is not a fixed pattern of behaviour, but is a changing, adaptive process directed to future results. The representation (and understanding) of intent by every party is therefore essential to co-operation, and so the role of communication in co-operation is important (Oberquelle, 1984, Oberquelle, Kupka & Maass, 1983). A high level of communication, with its connotation of "clarity of action", is implicit within co-operative behaviour, and is defined as "a complex of social actions for the purposes of mutual understanding and for allowing co-ordinated actions to occur" (Axelrod & Hamilton, 1981).

Thus, co-operative behaviour between humans is maintained by a combination of factors such as open communication, exchange of information and the existence of superordinate goals.

Grice's "co-operative principle" served as a guideline, and his "rules" (Grice, 1975) were followed in designing the computer-human communication in the following way. *Quality*: the computer's suggestions and solutions were always based on the highest priority of rule in the system (e.g. safety), or the best practice from the knowledge base (described below). *Relevance*: the computer always addressed itself to the user's stated goals, except where safety (or other highest priority) was

being breached. *Manner*: this rule proved to be more difficult to interpret. Shure *et al.* (1965) showed that the style, as well as the content of communication between parties facilitates co-operation, but does not necessarily create co-operation. On the other hand, any type of communication which can be identified by the other party as reducing potential threat is best suited towards increasing co-operative behaviour (Deutsch & Krauss, 1960). To comply with these findings, the computer was not allowed to "interrupt" the user; instead it was given its own window in which to draw its own suggestions and solutions. Updating of this window was initiated whenever the user defined and entered a goal which the computer could recognize and act on (Thus, also controlling the quality of the output from the computer.) The user could also request an explanation of the computer's actions: the computer responded using textual statements derived from its own knowledge (like the "why" facility in an expert system). In this way, the criteria of the "manner" rule were met.

TASK FACTORS WHICH SUPPORT CO-OPERATION

Not all tasks will enable the potential of co-operative behaviour to be realized. For instance, tasks which require a single correct answer (i.e. mathematical problems) tend to provide little scope for co-operation. However, situations where more than one alternative is sought provide a problem-solving environment which favours co-operative behaviour. Empirical evidence to support this position has been provided by two experiments. Firstly, Thorndike (1938) confirmed that the superiority of the group over the individual will be highest for tasks which afford a wide range of possible solutions. Husband (1940) found that pairs were superior to individuals when working on problems requiring some originality or insight, but not on more routine arithmetic problems.

PERFORMANCE REQUIREMENTS FOR A CO-OPERATIVE COMPUTER

The outline performance requirements drawn up for the computer and the problem were:

The co-operative computer:

- must feature the elements of co-operation, and operate within an acceptable framework of co-ordination and timing.
- must be able to recognize and accept the user's goals, when declared.
- must be able, with the user, to work towards superordinate goals in solving complex tasks, in an interactive manner.
- must offer alternative solutions to the problem being addressed.
- must not exhibit behaviour which appears to be threatening to the user, and should operate to support the formation of new attitudes in the user towards the computer and the task.

The problem to be solved:

- must require some originality or insight from the user.

THE USER

A type of user was identified who would most likely be able to benefit from engaging in co-operative behaviour in problem solving with a computer. In studies

of ability, Laughlin (1978) showed that homogeneously high-ability groups perform better than mixed ability groups in problem solving, and that high-ability persons perform better in co-operative groups with high ability partners than alone. It was therefore decided to orient the machine design towards the requirements of a generic high-ability user category, the highly skilled, but not necessarily computer literate, professional user. This user is characterized as having (a) detailed, task-specific knowledge and (b) the requirement or desire to undertake a wide variety of complex tasks.

A number of models prominently featuring the user, developed for the study of human-computer interaction, were examined as a basis for representing the user in the software. Layered models seemed best in general, representing as they do aspects of the user in layered form, from the general to the specific, or from the abstract to the concrete. In these models, a given layer is usually serviced by the layer beneath it, which other model types sometimes omit. Examples of layered models include those developed by Moran (1981), Nielsen (1984) and Clarke (1986).

The model by Clarke (1986) was applied in the present work because it gives a description of the user's goal orientation, and mental, and sensori-motor characteristics, together with the interfacing of these to the features and mechanisms of the computer, and the tasks.

THE USER AND THE INTERFACE

Our approach to the design of the user-computer interface attempted to parallel human interaction within the artificial constraints imposed during interaction with machines, following Scrinivasan and Dascher (1977) who identified the need to "develop a language structure that will embrace a broad diversity of the kinds of communication that users normally use". Thus, we drew on the social psychology of interpersonal behaviour in providing a clearer picture of how humans communicate during complex tasks. The work within social psychology was not seen as a solution in itself, but as another perspective from which we were able to view the problem of designing the interface.

The study of interpersonal behaviour includes the identification of factors based on individual and mutual needs which motivate the behaviour and modify the communication of parties in a task. Human communication is a highly complex interaction motivated by both task-related and social goals (Murray & Bevan, 1984). Tasks requiring either joint or dependent actions from parties can lead to the occurrence of either co-operative or competitive behaviour. Several factors have been identified which affect this choice of behaviour; these include the nature of the task, the expectations of the parties and the personal goals of either party. Kelley and Stahelski (1970) claim that there are two basic personality types: those expecting co-operation from others and those expecting competition. Co-operation was to be the model of machine interaction, and not competition, so it was essential to identify the task-related and social factors which induce such behaviour, since these factors would be providing a basis for the design of the co-operative computer system.

CO-OPERATION AND THE COMPUTER

Co-operation is an active process. A computer which incorporates this technique must overtly participate with the user during a task. A co-operative machine, while

remaining focused on an expressible goal state, should, to be productive, prompt the human partner to adopt a more divergent style of thinking during problem solving. Such interaction could foster a greater interdependency between the human and the machine, with the consequence of increasing the quality of solutions and encouraging greater user satisfaction.

A principal feature in co-operative behaviour during problem solving is the mutual creation of a common or shared environment, where the refinement of solutions can be based on logical argument, and where the resolution between differing perspectives takes place. It is through these discussions that the nature of the problem is revealed. Examples of techniques developed to foster divergent thinking in groups, and which were considered for use, include Brainstorming (Osborn, 1953) and Syntetics (Gordon, 1961).

The ability to generate and communicate alternative solutions is essential to human-computer co-operation, as it is these which spark the iterative solution process implicit within co-operation. It follows from this that different knowledge sets (but sympathetic views) are vital to a successful and productive co-operative relationship. The importance of such a dynamic interaction during complex problem-solving behaviour is reflected in Feyerabend's statement that "progress can only be brought about by the active iteration of different theories" (Feyerabend, 1965). Although only a part of the complex relationships involved in human-human co-operation, the *generation of alternative solutions* was chosen to provide a starting point toward representing the co-operative relationship between a human and a machine.

It became clear that our purpose would be best served by the development of a single-user co-operative system, where the generation of a satisfactory solution could be enhanced by a computer having the capability to generate alternatives and supplementary information, based on (i) the task, (ii) the user's defined goals and (iii) the current state of the user's solution. Further, in attempting to harness the potential of human-human co-operation, it also became clear that it was necessary to view the machine, not purely as a provider of information on request (as in an expert system) or simply as an agent carrying out specific work on demand, but as *an active partner working in real-time, in synchronization with the user*.

Our contention was that machine-generated alternatives could act as catalysts, and so play a more active rôle in the formation of ideas through changing the context in which the user would perceive the problem. This would provide a "greater perceptual span" (Jones, 1970). Thus, the refined aim of the work was to provide a problem-solving environment based on a co-operative paradigm, aimed at professional users.

THE UNDERLYING MECHANISMS OF A CO-OPERATIVE MACHINE

The first stage of the work identified those key factors which characterize, induce and maintain productive co-operative behaviour between human partners during problem-solving tasks. The next stage addressed the problem of how best to translate these factors (which are essentially behavioural characteristics) into mechanisms which could be successfully represented within a machine.

The first factor is the existence of a specific common goal and the working towards it. (Adopting superordinate goals subsumed the rewards requirement.) The process

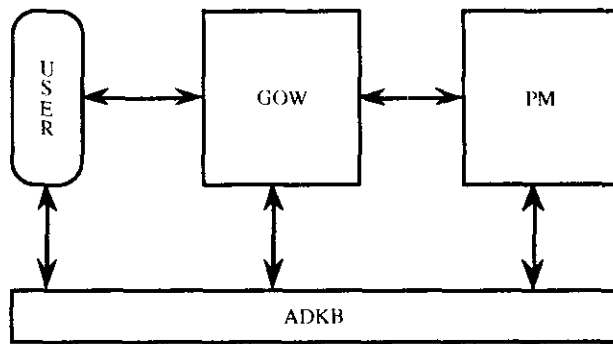


FIGURE 1. Relationships between the user and the mechanisms of the co-operative computer. GOW = goal-oriented working, PM = partner model, ADKB = agreed definition knowledge base.

or mechanism which represented this in the machine was called the *goal-oriented working mechanisms*, or GOW.

The second factor incorporates distributed responses, social co-ordination, and knowledge of the task domain. The mechanisms for this was called the *partner model* (PM), and took the basic form of a representation of the computer partner's specific task domain knowledge.

The third factor is communication. The core of communication is "meaning" but the representation of meaning was obviously outside of the scope of this work. Communication was therefore represented by (a) a language common to both partners, containing commonly held definitions, and (b) access to and use of the language by both partners. This mechanism was called the *agreed definition knowledge base* (ADKB). Figure 1 shows the relationships between the user and these mechanisms. Thus, although acknowledging that only a subset of the complexity of behaviour displayed during human-human co-operation might be seen in the interactive functioning of the mechanisms, it was decided that machine representations of GOW, PM and ADKB would provide an initial architecture with which to study, and subsequently implement, human-computer co-operation.

A METAPHOR FOR A CO-OPERATIVE TASK

It was initially intended to produce "general-purpose" co-operative mechanisms, because it appeared that general co-operative techniques and their associated advantages could be applied to any task. Two task domains were selected for consideration because of their richness of detail and logical sequencing: highway design and electronic circuit board design. However, analysis of these task domains made it clear that the knowledge-representation overhead associated with them would be beyond the scope of the work, if they were to support rudimentary co-operation. For example, the study of highway engineers in the field (Clarke, Woodcock & McDaid, 1986) showed that the highway design function was not carried out by a single person but by a number of individuals, each contributing specialist knowledge, and as such was not a suitable subject for the single-user human-computer co-operation task being considered. Nevertheless, our analyses showed that the design task is a valid application for co-operative techniques. Consequently, a number of smaller studies were made, (Shuttleworth, 1988; Smyth,

1987, 1988) which investigated how designers define problems and generate solutions. It was found that designers are interactively solution-focused, that is, they pose tentative solutions to a specific problem and by so doing learn about the problem.

Fischer, McCall and Morch (1989) reached a similar conclusion, although they came to it from consideration of the support that designers need. Their approach was that design environments need knowledge-based "critics", a critic being an intelligent support system which detects and criticizes partial solutions constructed by the designer, based on knowledge of design principles.

Gaines (1981), observed that "to make the system understandable is to maximize the possibility of the user forming, with the minimum of effort, a model of the system which aids his effective use of it". Such models are difficult to achieve if the interaction style of the system bears no resemblance to that of the human user. (To some extent this problem has been alleviated by the use of metaphors during interaction, where the metaphor acts as a link to the user's previous experience and thereby assists in the prediction of machine behaviour: this fact was used by us, and is described later.) It became clear that the early aims of producing general co-operative techniques were impractical, and that there was a need for an easily understood, familiar, specific task which could act as a vehicle for the detailed analysis of co-operative behaviour. As a preliminary to the identification of such a task domain a number of requirements were identified.

These were that:

- (a) the task should be achievable,
- (b) there should be a number of possible solutions within a finite set of solutions, so as to permit the production of alternatives by the co-operating parties,
- (c) task goals should be identifiable and expressible at the outset of co-operation,
- (d) explicit rules should be involved in the production of solutions,
- (e) the task should support the generation of partial solutions and
- (f) the task should support graphic-based interaction.

The problem domain selected was room layout design† which, as an instance of spatial design, met the requirements. The decision to employ a task metaphor proved to be a key decision because it focused analysis, enabled the research dialogue, and provided a vehicle for the eventual implementation of the mechanisms and the user-computer interface of the co-operative computer.

THE DESIGN PROCESS AND CO-OPERATIVE PROBLEM SOLVING

Design can be described as the successive application of constraints until only a unique product is left. Laurel (1986), states that constraints, rather than restricting the process of design, "provide the security net that enables people to make imaginative leaps". Co-operation, with its agreed common goals and behavioural norms, provides a secure environment for making such leaps. The potential

† It is stressed here that the co-operative software computer produced was never intended to be viewed as a room layout application, since sophisticated software solutions already existed for this particular problem. It was strictly for use as a medium for exemplifying the function of the three underlying mechanisms of co-operative behaviour. Room design, as such, acted as a metaphor for the co-operation between humans and computers.

advantages of co-operative behaviour within the design process have been indicated by several researchers. Broadbent (1973), concluded that, "in creative thinking, face-to-face group members may 'spark off' ideas against each other by an assembly effect so that finally the quality of ideas is higher than the individual could have achieved". While Middleton (1967), in observing that hypothesis testing is an area which benefits from group activity, highlighted the "distinction between original thought, which is a lonely activity, and the testing of hypotheses, a logical process where the group can participate with advantage". From this, it was clear that design would be ideally suited to co-operative work, because it is a task requiring joint or dependent action from all the participants.

Rules of design were then identified, particularly those of spatial layout, which could be incorporated in the partner model. These rules would be at the core of the mechanism with which the co-operative machine would generate meaningful alternative solutions, a factor vital to achieving synergy within the co-operative dyad of human and computer. This point is also made in Fischer, McCall and Morch (1989).

Designers are often confronted with the task of choosing, from among an extremely large number of possible arrangements, an aesthetically satisfying composition of objects within a given space. Some choose to rely on the intuitive application of their skills. Tufte (1983) suggests, in reference to graphic design, that "there are no compositional principles on how to create that one graphic in a million". Nonetheless, some designers reduce the number of possibilities in compositional tasks by applying rules which govern the placement of objects in a few initial sketches. They then choose to observe the rules, apply further rules or develop compound effects in response to the resulting compositions. A fundamental compositional design approach is to arrange objects along lines of symmetry. The "Golden Section" rule (Holt, 1971) was implemented in the present machine, because established design rules would need to be made available to the partner model if it were to co-operate successfully with the designer, and the Golden Section was readily available and understandable in its application. However, since the role of the computer was paradigmatic, and thus called for only a few archetypical rules to illustrate the principles of co-operation, it was not necessary to research the area exhaustively.

In adopting the room layout metaphor, it followed that the requirements of the agreed definition knowledge base (ADKB) could be met by the inclusion of a subset of generic furniture objects, a subset of generic room objects and definitions of the relationships (permitted and forbidden) between the objects. These could then be acted on by the goal definitions and design rules represented in the computer.

Representation of co-operative mechanisms

GOAL-ORIENTED WORKING

It is essential in co-operation that the co-operating parties know what the desired goal is, and work towards attaining it. Two parties working towards different or conflicting goals will not achieve co-operation. Thus, if a human-human or human-computer co-operative dyad is to achieve success, there must be agreement,

reached via a process of communication, about the goal to be accomplished. In so doing, the parties declare their intent to each other, a necessary precondition for co-operation.

In English vernacular, expression or declaration of goals most commonly takes the form of an operational statement; e.g. "We want to design a co-operative computer". What would be generally inferred from this statement is that the speakers want to have a design, complete and finished, for a co-operative computer. However, the speakers may wish to be continuously engaged in the process of designing. Thus, there is ambiguity in the usual way of expressing goals.

To avoid that problem a definition was developed for this work which enabled goals to be declared to the machine in a form suitable for expression in a high-level language. It is given here, (the "OR"s are exclusive):

"A goal is (a) the intended state of an object, or (b) the intended relation between two objects or more, or (c) the intended maintenance of a process".

The action of achieving a goal is thus directed toward goal objects or processes which may either be physical or virtual, and can result in their being created, eliminated, modified or sustained. Each goal object has a number of attributes, such as colour or dimension, which may be altered in the act of achieving a goal. For example, the goal "I want to sharpen a pencil" may be restated in the form:

Goal is: to_have_pencil_in_sharpened_state.

(where "to_have" represents the futurity or intent in the goal declaration).

In this example, the goal object is the pencil and the achievement of the goal will result in the attributes of sharpness and length being changed, (the one being intended, and the other [perhaps] not being intended). Provided that the object "pencil" and the attribute "sharp" is definable, then the goal can be represented in the machine.

A similar approach is taken for declaring goals of relationship. For example:

Goal is: to_have_telephone_on_table.

THE AGREED DEFINITION KNOWLEDGE BASE

Another of the important features of co-operative behaviour is the ability of either party to communicate the current state of the interaction. Here it is critical that both the co-operating parties share the same object definitions. This factor is emphasized during human-human co-operation, where a large amount of time is taken up with the mutual identification of, and agreement of, terms of reference to be adopted during the process. The definitions do not have to remain static as long as changes are agreed by both parties. If a co-operative machine is to be fully developed it must support the interactive updating of the shared knowledge base.

THE PARTNER MODEL

The ability of parties to generate alternative solutions within a problem-solving task is an important part of co-operative behaviour. To represent this ability in the machine, a mechanism was constructed (the partner model [PM]), which could

access a knowledge base common to both the machine and the user (the ADKB). The model consisted of domain-specific rules. The model's rules were applied to the goal-related definitions in the ADKB, and the results were then communicated to the user. This process generated viable alternative solutions, and thus presented a similar behaviour observed in human co-operation. (Note that the partner model is not an embedded user model, in that while it possesses similar domain information to the user, it does not necessarily reflect the working style of any particular end user.)

The mechanism not only generated alternatives; it also facilitated interaction with the user, who was able to query the rule base. This was essential, for if the user could comprehend the processes involved in generating alternative solutions, then it would be possible to incorporate the PM's techniques into the user's own design strategy. Communication of alternative configurations of a design solution within an interactive environment might facilitate the development of a user's task expertise. More specifically, during interaction with the machine, the human partner could begin to adopt the task skills represented within the PM.

The requirement thus emerged for a PM which would be autonomous, and which would demonstrate the importance of the differences between parties. (This is in apposition to the embedded user model, which attempts to extract characteristics exhibited by the user during the interaction and ultimately to mimic or predict the user's solutions; a situation which, the authors hold, runs contrary to the philosophy of human-computer co-operation.)

The co-operative computer

SYSTEM DESIGN DECISIONS

Three system design decisions were made at the outset of the implementation of the co-operative computer:

The first, pragmatic, decision acknowledged the reality of the difference in the spheres of competence between the human and the untested machine, and potential user rejection of initiatives taken by the machine. It was decided to place control in the hands of the user at those points during the interaction where aesthetic and functional judgements would need to be made (e.g. decisions about what constitutes a "satisfactory" layout solution).

The second was to minimize explicit, broad-band communication between the user and the machine, and instead, wherever possible, to use an implicit, graphically-based, communication technique; namely, the visual representation of task information.

The third was to build the user interface around the concept of an "interaction event", i.e. a piece of information generated by a range of internal or external circumstances, such as a key-stroke, mouse event, network activity or an output from the system clock. The system was implemented on a Hewlett Packard 3500 workstation, using C-Prolog.

VISUAL REPRESENTATION OF TASK INFORMATION AND THE INTERFACE

The authors' observation and analysis of the design and communication activities of human designers co-operating during a complex problem-solving task enabled them

to construct a user-computer interface having two dedicated graphical windows. In terms of the room layout metaphor, these windows represented the floorplan as perceived by the user and the partner model respectively. This was a simple solution, which overcame the problem of simultaneously presenting alternative layouts to the user, while not causing a distraction from the current task in hand.

Objects in the user's window were positioned by dragging with a mouse, while other actions, such as the construction and declaration of a goal, the prioritization of the partner model rule base, and the creation of objects, were achieved through software buttons.

THE CO-OPERATIVE COMPUTER'S MODES OF OPERATION

The co-operative computer had two modes of operation, "Interactive" and "Solve". The Interactive mode was the primary mode, and in operation, the activation of the partner model was controlled by the user's selection of object locations. The Solve mode was used during software development and so was not central to the development of the co-operative relationship. When the Solve mode was activated the PM immediately generated an alternative solution based on the current active goals and the location of objects in the user's solution space; this was used to test the validity, for example, of the software under development.

MANIPULATING THE AGREED DEFINITION KNOWLEDGE BASE (ADKB)

The room design metaphor contained a deliberately constrained object set, presented visually to the user throughout the interaction. Selection of an object type causes the creation of an instance of that object to be placed in the inventory. The user is then at liberty to incorporate that object instance within a goal construct, or simply to place a graphic of the object in the user window at the desired location. Both the user and partner windows represent a notional floorplan. As objects are located in the user window the PM generates alternative positions based on its rules, and displays its option in the partner window. Figure 2 shows the PM's range of possible responses to a user-declared goal, "Desk_near_wall1, Chair_near_desk".

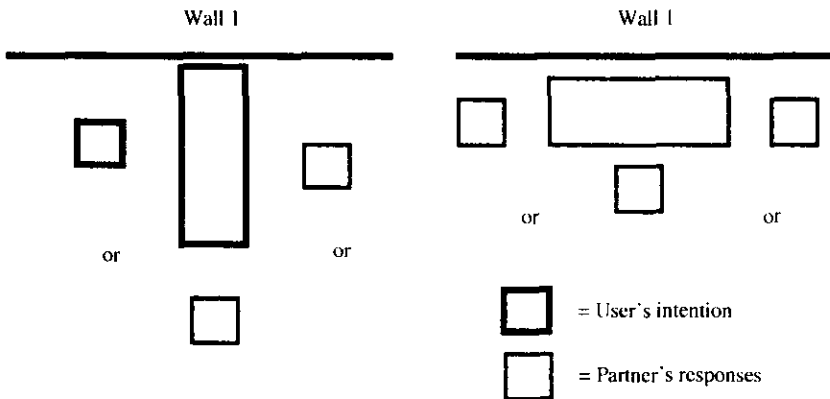


FIGURE 2. The partner model's (PM's) range of possible responses to the goal *Desk_near_wall 1, Chair_near_desk*. All solutions are valid within the definition.

The user may have had one solution in mind (shown in bold in Figure 2), but has expressed it ambiguously, and the PM has responded accordingly. Unlike a conventional computer, the co-operative computer is able to respond to ambiguities in a productive way; the PM interprets the declared goal according to its own rules and is able to generate a number of solutions (valid in context), showing them one at a time.

This behaviour of the co-operative computer, as we implemented it, underlines the conclusion reached by Fischer, McCall and Morch (1989) who, in developing and evaluating intelligent support systems for subjective problem domains such as design, observed that there is no "optimal solution" but instead, a set of alternative solutions which satisfy, (i.e. solutions which are dependent upon a particular metric and a particular viewpoint).

Throughout this process, parallel ADKBs are being continually updated as the user and partner generate alternative layout solutions using the available objects. This adds to the knowledge base, and enables the PM to respond in an ever-increasingly sophisticated way.

The standard object data structure is as follows:

```
object(<user/partner>,
      object_type(object_instance),
      location,
      <permanent/moveable>).
```

Transfer of data from the databases is at the discretion of the user, who can either freeze objects in the user's window (thereby barring input from the partner model), or transfer object configurations generated by the partner model into the user's solution, where they are incorporated. Both of these techniques have similarities to those observed in human-human co-operative behaviour.

INTERFACING WITH THE GOAL-ORIENTED WORKING (GOW) MECHANISM

A software technique (developed in LUTCHI), which makes it possible to construct goals visually, provided the enabling technology for the representation of goal-oriented working (GOW) within the computer. By using the technique, the user is able to construct spatial relationships between objects, which are then translated into a Prolog rule base. This enables the PM to manipulate them as goals. In this way, the user can declare his or her goals to the GOW mechanism in the co-operative computer, which in turn can then work towards the, by now, common goals. The availability of this method was significant to the development of the system software and its use.

A representational number of spatial relationships were created in the system, and users were able to construct and delete goals as required during the interaction. The relationships were object-sensitive; e.g. the concept of nearness is quantitatively different when applied to a desk-wall combination than to a desk-chair combination.

The GOW mechanism also included simple error checking which alerts the user to possible sources of goal conflict. Once the goal list has been created, the software works out the order in which to place the objects based on the logical properties of the objects and their declared relationships.

So, for example, if the following goals were created in the following order:

1. . . . phone_on_desk
2. . . . desk_near_wall

the desk would have to be positioned first before the initial goal could be achieved. If goal fulfilment was to be object-independent, then the order of creation of the goals was taken to be the deciding factor. Examples of user constructed goals, shown in the format actually used on screen, are given in Figure 3.

INTERACTING WITH THE PM

The PM basically consisted of rule sets, i.e. subsets of design knowledge concerned with the proportionality of the relationships between objects in a finite space: in this case the location of furniture within a room. Examples of the rule sets mounted in the PM included; Symmetry, Golden Section and Safety (the latter reflecting a different style of rule). Application of the safety rule, for example, would always keep an access route clear from the window to the door as a priority. (The user could change the priority of rules but the computer could not.) The user can select the particular set to be applied by the PM. This enables the user to view a number of alternative object configurations, each generated by the PM, exhibiting a different emphasis or style. Division of response is thus placed in the hands of the user, and he or she can require the PM to place a high priority on safety, for example, so being free to pay more attention to the stylistic aspects of the design task. Figure 4 shows how representative design solutions appeared when the PM observed safety priorities and the user did not. (Here, one would expect the user to adopt the PM's solution.)

At each stage in the user's development of a solution, the PM interrogates the resulting object configuration. It then applies its own rules (i.e. the rule set) and generates a composition of the objects derived from the user's current solution. This takes place within the limitations of the framework of the user's declared goals. Incorporated within the PM's solution strategy is a mechanism for the identification and consequent rejection of any alternative solution which might break fundamental constraints implicit in the task domain. The mechanism, for example, operates using the statements "objects must remain within the bounds of the room", and "objects must not occupy the same space". An exception to the latter is the object

cabinet1	near	wall2
bookcase1	near	wall3
bookcase1	near	wall4
desk1	near	window1
chair1	in_front_of	desk1
chair2	in_front_of	desk2




FIGURE 3. Examples of goals constructed by the user, in the interface format.

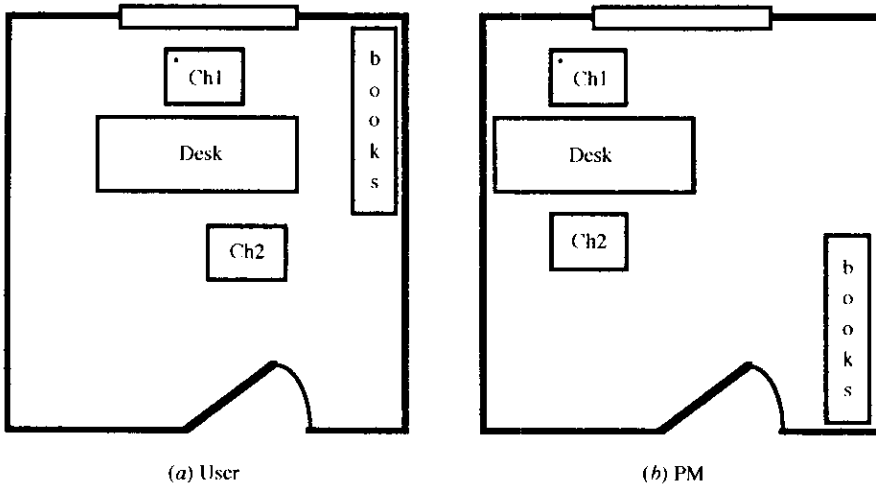


FIGURE 4. User and partner model (PM) worked to the same goal, but the PM also implemented the safety rule "keep a clear emergency route from door to window".

"telephone", which is permitted to occupy the same space (i.e. in plan) as any furniture object. The "telephone_on..." goal is not commutable, i.e. the telephone can be put on any object within the walls, including the floor, but nothing may be put "...on_telephone".

PERFORMANCE EVALUATION

Although the perceived way in which the machine achieved declared goals conformed with the behavioural features associated with co-operation, it was clear that the reaction of some representative users would be informative if a balanced view of the machine were to be obtained. Accordingly, a small number of designers were invited to use the co-operative computer in an informal evaluation.

Because of the difficulty during the time of the research in obtaining rule-sets, the set employed in developing the machine was used, with all its shortcomings, in the evaluation. Consequently, the usefulness of the machine was limited in the evaluation. This was explained to the designers, and after a brief introduction to the method of declaring goals and operating the computer, they attempted a range of design problems. Their overall judgement was that the co-operative computer provided rather poor design solutions (due to the limitations of the rule-set used), but that the solutions offered by the machine at various stages in the joint design process provided a very powerful prompt to their own imaginations, and helped them to define and express their own goals in a clear and structured way, to an extent that they had not experienced before. (This, of course, is one of the advantages of co-operation.)

They felt, initially, that the interface placed restrictions on them, but they soon became used to it, and then started to develop some facility with the system. They also noted a distinct and different type of interaction between themselves and the co-operative computer, in comparison with their experience of other "ordinary" computers, in that not only were they manipulating data in a different way, but also

the machine was separate to, but “with them” in trying to solve the same problem. This is an important result, albeit from a limited sample of users, because it is an indicator of the behavioural *gestalt* observed in the best type of co-operation. The designers found this difference intriguing but in no way threatening.

The designers enjoyed using the machine, and were able to accept its way and speed of operating, and its alternative solutions. They were quite prepared to carry out field trials of such a machine, provided that rule-sets more representative of their own task domain and experience were made available in the PM. (Fischer, McCall & Morch [1989] also let computer science students, design students and a kitchen designer evaluate their CRACK system [a knowledge-based critic]. Their performance also suggested that students could produce better designs and learn about design principles through the use of this type of system.)

Discussion and conclusions

OVERALL CONCLUSION

The primary objectives of the work described in this paper were to identify the mechanisms central to the development of a co-operative computer and to exemplify these mechanisms in software. It is concluded that these objectives have been achieved, and that a prototype computer that could properly be called “co-operative” was successfully constructed.

CONCLUSIONS ABOUT THE PERFORMANCE EVALUATION

The goal-oriented design solutions offered to the user by the partner model are viable, alternative configurations of the user's objects, developed using a different rule set to that employed by the user; the solutions are, however, neither “right” nor “wrong”, but increase the user's understanding of the design problem.

It is concluded that the most potent effects of the partner model's solutions or “suggestions”, are (a) to increase greatly the user's opportunity to interact with the design problem, thus leading to an increased understanding of the problem, and (b) to spur the user's creativity; an important feature of the co-operative machine, and one which differentiates it from an expert system. For example, Fischer, McCall and Morch (1989) found that designers often got “lost” during extensive browsing in VIEWPOINTS (a hypertext system containing useful information about the principles of kitchen design, in the form of a “default” knowledge base of design principles) since there were a large number of “issues”, i.e. specific problems, in the information base, but no facility available for guiding designers in the right direction.

Thus, in interacting with the PM, the user can try out alternative solutions, make comparisons with the PM's proposed solutions, or incorporate all or some of the PM's solution into an acceptable outcome. In this way, it is concluded, the co-operative computer can either provide a direct input to the problem-solving process, or can act as a Rogerian sounding-board against which ideas can be explored and evaluated.

CONCLUSIONS CONCERNING SOFTWARE DEVELOPMENT

The degree of co-operation achieved by the machine is a result of interaction between the underlying mechanisms, and not just a result of their individual actions. In principle, the computer-human interface makes this co-operative behaviour manifest to the user. However, the authors note that the interface software requirements only became fully definable when the room design metaphor had been adopted. It is concluded that although an overview of the functionality of the mechanisms of the intended software is important, a complete definition of a task metaphor is a prerequisite for the development of software for a co-operative machine. By using "visualization of knowledge" techniques based on the metaphor, the resulting interface enabled the user to more readily (a) define the task objects, and (b) declare the required goals.

Thus, the user interface of the co-operative machine enabled the user to enhance the agreed definition knowledge base, improve the process of generating alternative solutions, and increase the degree of co-operation.

CONCLUSIONS CONCERNING THE PM

Co-operation between humans is primarily a dynamic relationship, because both parties are learning or reinforcing the concepts being expressed during the interaction. However, it is possible that the co-operative relationship could, over time, become static. Without the introduction of new sources of information, a co-terminous commonality of knowledge could arise between the partners, so reducing the effectiveness and, ultimately the creative potential, of the relationship. In human-human co-operation, information relevant to the task domain can come from a multitude of external sources. However, in the co-operative computer being described, the PM is static, and does not develop during the interaction. So, as the user becomes cognizant of the PM rule set, the machine-generated alternatives will become predictable, and then will no longer serve as an aid to the user's imagination. It is concluded that the ability of the human partner to learn during co-operative interaction implies (at best) or guarantees (at worst) a limited period of usefulness for a static partner model in a single-user system.

The main conclusion drawn from modelling a co-operative partner within the machine is as follows. If the PM simply mimics the user, then there is a risk that if the user is unable to solve a problem, then neither will be able to solve the problem. On the other hand, if the knowledge or expertise of the parties is too disparate, there will be little or no communication between them. The most fruitful relationship will occur when both parties have similar, but not identical, capabilities, and are capable of learning from the co-operative interaction.

FUTURE DEVELOPMENT OF THE CO-OPERATIVE COMPUTER

Successful future development of the co-operative computer must address a number of issues, chief of which are:

- (a) Because the extent of co-operation varies between tasks, the existing problem-defining and problem-solving processes adopted by human partners in a particular task must be clearly identified before attempting to build a co-operative machine with which to address the same or related tasks.

- (b) The mechanisms of GOW, the PM and the ADKB, as described in this paper, provide a viable outline architecture for future research into, or development of, co-operative machines.
- (c) The quantity and quality of machine co-operation perceived by the user depends upon the interaction of the software mechanisms and the way in which the machine's resulting behaviour manifests itself to the user, and, as a corollary to (c).
- (d) The knowledge bases of the PM and the ADKB should be capable of being updated, either from each interactive session with the user, or from other external sources, or (preferably) from both.

The work described was carried out as part of the Alvey Human-Computer Co-operation Project MMI/062, and was completed in 1989. Thanks are due to our colleagues at the LUTCHI Research Centre: in particular, the grant holders, Ernest Edmonds and Steven Scrivener, plus Steve Bell, Mark Buckley, Tom Bayley and Andrée Woodcock. Also to Mark Shuttleworth of ICL, and to Dr John Pinkerton, our Technical Monitor, for his encouragement.

References

- AXELROD, R. & HAMILTON, W. D. (1981). The evolution of co-operation. *Science*, **211**, 1390-1396.
- BIRENBAUM, A. & SAGARIN, E. (1976). *Norms and Human Behavior*. New York: Praeger Publishing.
- BLAKE, R. R. & MOUTON, J. S. (1962). The inter group dynamics of win-lose conflict and problem solving collaboration in union management relations. In M. SHERIF, Ed. *Intergroup Relations and Leadership*. pp. 94-140, New York: John Wiley & Sons.
- BLAKE, R. R., SHEPARD, H. A. & MOUTON, J. S. (1964). *Managing Inter Group Conflict in Industry*. Houston, TX: Gulf.
- BROADBENT, G. (1973). *Design in Architecture*. London: John Wiley & Sons.
- CLARKE, A. A. (1986). A three-level human-computer interface model, *International Journal of Man-Machine Studies*, **24**, 503-517.
- CLARKE, A. A., WOODCOCK, A. & McDAID, E. (1986). *The highway engineer; a typical user?* HCC Internal Report HCC/L/9, Loughborough, Loughborough University of Technology, UK.
- DAVIS, J. H. (1969). *Group Performance*. Wokingham: Addison-Wesley.
- DEUTSCH, M. (1949). A theory of co-operation and competition. *Human Relations*, **2**, 129-152.
- DEUTSCH, M. (1962). Co-operation and trust: some theoretical notes. *Nebraska Symposium on Motivation*, pp. 275-230.
- DEUTSCH, M. (1968). The effects of co-operation and competition upon group processes. In D. CARTRIGHT & A. ZANDER, Eds. *Group Dynamics*, pp. 461-482. New York: Harper and Row.
- DEUTSCH, M. & KRAUSS, R. M. (1960). The effect of threat on interpersonal bargaining. *Journal of Abnormal and Social Psychology*, **61**, 181-189.
- FISCHER, G., McCALL, R. & MORCH, A. (1989). Design environments for constructive and argumentative design. *Proceedings of CHI'89*, pp. 269-275. Austin, Texas, April 30-4 May.
- FEYERABEND, P. (1965). Consolations for the specialist. In I. LAKATOS & A. MUSGRAVE, Eds. *Criticism and the Growth of Knowledge*. Cambridge: Cambridge University Press.
- GAINES, B. R. (1981). The technology of interaction—dialogue programming rules. *International Journal of Man-Machine Studies*, **14**, 133-139.
- GORDON, W. J. (1961). *Synetics, the Development of Creative Capacity*. London: Harper & Row.

- GRICE, H. P. (1975). Logic and conversation. In P. COLE & J. L. MORGAN, Eds. *Syntax and Semantics*, Vol. 3, pp. 41–58, "Speech Acts". London: Academic Press.
- HOLT, M. (1971). *Mathematics in Art*. London: Studio Vista.
- HUSBAND, R. W. (1940). Co-operation versus solitary problem solution, *Journal of Social Psychology*, **11**, 405–409.
- JONES, J. C. (1970). *Design Methods: Seeds of Human Features*. New York: John Wiley.
- KELLEY, H. H. & STAHELSKI, A. J. (1970). Social interaction basis of co-operators' and competitors' beliefs about others. *Journal of Personality and Social Psychology*, **16**, 66–91.
- LAUGHLIN, P. R. (1978). Ability and group problem solving. *Journal of Research and Development in Education*, **12**, 114–120.
- LAUGHLIN, P. R., MCGLYNN, R. P., ANDERSON, J. A. & JACOBSON, E. S. (1968). Concept attainment by individuals versus co-operative pairs as a function of memory, sex and concept rule. *Journal of Personality and Social Psychology*, **8**, 410–417.
- LAUREL, B. K. (1986). Interface as mimesis. In D. A. NORMAN & S. W. DRAPER, Eds. *User-Centred System Design: A New Perspective on Human-Computer Interaction*. pp. 67–85, Hillsdale, NJ: Lawrence Erlbaum Associates.
- MARWELL, G. & SCHMITT, D. (1975). *Co-operation—an Experimental Analysis*. London: Academic Press.
- MIDDLETON, M. (1967). *Group Practice in Design*. London: Architectural Press.
- MORAN, T. P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, **15**, 3–50.
- MURRAY, D. & BEVAN, N. (1984). *The social psychology of computer conversations. Proceedings of Interact'84*, pp. 268–273.
- NIELSON, J. (1984). A virtual protocol model for computer-human interaction. *DAIMI PB-178*, Aarhus, Aarhus University: Computer Science Department.
- OBERQUELLE, H. (1984). On models and modelling in human-computer co-operation. In G. VAN DER VEER *et al.*, Eds. *Reading in Cognitive Ergonomics*. pp. 26–43, Berlin: Springer Verlag.
- OBERQUELLE, H., KUPKA, I. & MAASS, S. (1983). A view of human-computer communication and co-operation. *International Journal of Man-Machine Studies*, **19**, 309–333.
- OSBORN, A. F. (1953). *Applied Imagination*. New York: Charles Scribner's Sons.
- ROTHSTEIN, S. & PIEROTTI, R. (1988). Distinctions among reciprocal altruism, kin selection and co-operation, and a model for the initial evolution of beneficent behavior. *Ethology and Sociobiology*, **9**, 189–209.
- SHERIF, M. & SHERIF, C. W. (1953). *Groups in Harmony and Tension*. New York: Harper.
- SHURE, G. H., MEEKER, L. J. & HANSFORD, E. A. (1965). The effectiveness of pacifist strategies in bargaining games. *Journal of Conflict Resolution*, **9**, 106–117.
- SHUTTLEWORTH, M. (1988). *Office layout: a user/task analysis*. HCC Internal Report HCC/I/25, Loughborough, Loughborough University of Technology, UK.
- SMYTH, M. (1987). *Toward a co-operative design system*. HCC Internal Report HCC/L/20, Loughborough, Loughborough University of Technology, UK.
- SMYTH, M. (1988). *Articulating the designers mental codes*. HCC Internal Report HCC/L/24, Loughborough, Loughborough University of Technology, UK.
- SRINIVASAN, C. A. & DASCHER, P. E. (1977). Information systems design: user psychology considerations. *MSU Business Topics*, **25**, 51.
- THORNDIKE, R. L. (1938). The effects of discussion upon the correctness of group decisions when the factor of majority influence is allowed for. *Journal of Social Psychology*, **9**, 343–362.
- TUFTE, E. R. (1983). *The Visual Display of Quantitative Information*. Connecticut: Graphic Press.
- YOUNG, J. Z. (1979). *An Introduction to the Study of Man*. Oxford: Oxford University Press.