THE MEETING OF MAN AND MACHINE

Margaret. A. Boden

ognitive Science Research Report

erial no: CSRP 022

he University of Sussex
ognitive Studies Programme
chool of Social Sciences
almer
righton BN1 9QN

THE MEETING OF MAN AND MACHINE

Margaret A. Boden

The task of improving the relationship between man and computer can be conceptualized at two different levels, each of which has practical importance in a society wherein computers are in constant use outside research laboratories. On the one hand, we can design "friendly" programs for the public domain, programs which non-specialists will find easy, and even pleasant, to use. So we can ask, for example, how the comprehensibility, flexibility, and modifiability of specific systems can be increased. On the other hand, we can concern ourselves with the general image in the public's mind of man and computer, and of the relation between them. Is a meeting between man and machine likely to be dehumanizing, in respect of how people think about themselves and their fellows? I shall discuss the second, more general, level first, because some things follow from it with respect to how one might write individual programs for public use. To consider the practical problem of designing information systems for human beings without reference to these apparently abstract philosophical issues is to risk arousing both inappropriate resistance and inappropriate acceptance on the part of the nonspecialist user.

Many people assume it to be obvious that contact with "intelligent" computers, even more than familiarity with the natural sciences in general, will undermine our confidence in our own humanity. From Blake in the early nineteenth century to Marcuse in our own time, critics have attacked the natural scientific tradition, and technocratic societies moulded by it, for their one-dimensionality, their dehumanizing emphasis on objective facts and their concomitant failure to come to terms with subjective phenomena. Human beings are essentially subjective creatures, living out their lives from within their own idiosyncratic views (descriptions) of the world. Their humanity tends to be overlooked or subtly undervalued by approaches rooted in the traditional natural sciences, because these sciences have no theoretical concepts capable of expressing subjectivity. Statements within natural science cannot express subjective facts about the beliefs and desires of a psychological subject; rather, they express objective facts which can be stated without any reference to such a subject. Since the scientific tradition has no way of stating truths about subjective phenomena, and so must ignore them, its enormous success within its own sphere has inevitably had a subtly dehumanizing influence.

In my view, the prime philosophical importance of programmed models of the human mind or intellectual capacities is that such models themselves are best understood as symbolic, representational processes. That is, a complex program must be conceptualized as a system which stores, constructs, and infers from descriptions of the world (including descriptions of its own goals, priorities, and abilities). It is the program's internal descriptions which determine what it does, what decisions it takes, what inferences it considers plausible. This is why a (badly) programmed librarian, instructed to search for literature about the tailoring industry, might recommend us to read Dorothy L. Sayers, due to its assumptions about the likely relevance of books with the word "Tailors" in the title.

It follows that a program is in an important sense a non-objective system. That is, it is to be construed as a representation of reality (and possible realities), whose faithfulness or truth can only be decided by reference to epistemological principles extrinsic to the representation itself. As with any representational system (including human minds), the degree of match between the descriptions stored within a program and objective reality can always in principle be questioned, as can the degree of match between its inferential strategies and what we regard as truly rational reasoning. The imaginary librarian just mentioned may have its facts right (Dorothy Sayers did indeed write The Nine Tailors), but its inference rules are clearly faulty: it is not the case that all books mentioning tailors in the title are relevant to tailoring.

This point can be expressed by saying that a program is significantly analogous to a subjective system, insofar as it is a representational system embodying symbolic descriptions of the world (and possible worlds), as opposed to something (like sticks and stones) that exists in objective reality. But it is important (as we shall see later) to remember that programs are only metaphorically subjective. Their "subjectivity"[11] is parasitic on the genuine subjectivity of human programmers, their "beliefs" and "inferential strategies" being derived from those of the programmer concerned. This is true even in the case of those programs that are so heavily dependent on the self-modifications of previous programs that they cannot straightforwardly be ascribed to any particular human authors. Most programs for public use in the foreseeable future will be ascribable to such authors, so that the specific nature of their subjectivity, or their "subjectivity" if you prefer, will have been determined by identifiable human individuals. The importance of this point for our current concern will appear presently.

(Programs are, of course, always "objective" in a different sense, as humans sometimes are too: they are impartial in applying criteria. People are often implicitly influenced by factors such as sex, class, race, religion, or personal appearance, in situations where they will admit that these factors are really irrelevant. In the sense of "objectivity" that denotes impartiality, then, a program is capable of greater objectivity than is a human being. However, since it is possible for inherently unfair criteria to be applied with scrupulous impartiality, we cannot assume that a decision made by a computer must be a fair one. Sexist programs, for instance, will give substantively unfair judgments even though – unlike people – they are not swayed by personal moods and preferences, or by social prejudices.)

The essentially representational (non-objective) nature of computational systems has two implications that are relevant to our problem. First, this approach if correctly understood is not only not dehumanizing, but is positively humanizing. For it offers a scientifically grounded way of accounting for and giving due weight to subjectivity. Consequently, it can admit that hermeneutic (interpretative) psychologies – such as Freud's, for instance – are worthy of attention. Materialistic and behaviouristic ways of thinking, like the natural scientific tradition in general, lead people to assume that such theories must be inherently mysterious or absurd – as opposed to being conceptually unproblematic, though perhaps false. But in his account of neurosis, dreaming, and slips of the tongue Freud viewed the mind as a symbol-manipulating system, which is to say that at base he was dealing with computational issues that in principle can be addressed

by people working in an information-processing paradigm. (Whether his theory is true and/or clinically helpful is another matter.) In general, theories about consciousness, and about psychological processes which though unconscious are of essentially the same (viz. subjective) character, are admissible in a computational paradigm. And comparing the mind to a complex program can even illuminate deeply puzzling conscious phenomena, such as the dissociated consciousness typical of "split[11] or "multiple[11] personality C13-

Second, since programs are analogous to subjective systems rather than to objective things, it follows in principle that they can always be questioned by other subjective systems – such as human beings. Even if a program has no bugs in it, its data and also its inferential processes are in principle open to epistemological debate, just as human knowleage and reasoning are. So, for instance, we can question the inferential assumption made by the program mentioned earlier, namely, that every book with "Tailors" in the title must be relevant to the rag trade. In particular, the public should not allow themselves to be bullied by remarks like "The program says so, and after all the program reasons objectively, and can't be wrong." We have already seen that this sense of "objectivity" denotes, not factual truth, but impartiality in applying criteria when making decisions.

Informatics professionals have a responsibility to alert the public to these facts. In principle, these issues can be brought to general attention in two different contexts – within the specific programs used by members of the public, and within the general propaganda about computers that is bruited in society.

The way in which one should attempt to influence the latter context is reasonably clear: one should make these points (about the essential non-objectivity, or subjectivity, of programs) explicitly in one's descriptions of programs and discussions about "the computer society," reminding people of them when they appear to have been forgotten. And, most important of all, one should try to ensure that they enter the educational experience of the average person (not just the computer specialist).

Several universities are already running courses for non-computer science students with these aims, among others, in mind, and some people are already doing comparable work with school pupils. As these courses have shown, it is possible to alert totally naive users, on their first day of programming experience, to the facts that even an "intelligent" program is incapable of doing many things that one might prima facie expect it to be able to do, and that even the nonspecialist user can alter the program so as to make it less limited. That is to say, the system is neither godlike nor unalterable. In addition, of course, the students are given the beginnings of confidence in the activity of programming, with the realization that it is they who are altering these complex and (up to a point) impressive systems. These educational projects *are* of the first importance, for "computer literacy" will be necessary if people are to be able to take advantage of this new technology rather than being taken advantage of by it. (This is not to say that people are never taken in by the printed word, whether in a book, newspaper, or advertisement – but the common, mistaken, assumptions about the objectivity of computer programs compound the problem in this area.)

In their discussion of the future with microelectronics, lann Barron and Ray Curnow point out that, in addition to vocational training and adult retraining, we shall need contextual education to ensure that everyone is aware of the technology and its potential consequences. They remind us that, as users get less and less expert, there will be an increasingly urgent problem of producing relevant (though nonspecialist) courses in higher education, and they conclude that "It should perhaps be a target that every graduate has the capability to use computer systems and a thorough understanding of their potential " [23. My contention is that "a thorough understanding of their potential[11] includes, importantly, an understanding of the sense in which they may be regarded as (metaphorically) subjective systems. The more this non-objectivity is recognized by the general public, the less chance there will be of insidiously dehumanizing influences like those attendant on the natural sciences, and of people's being mystified and "bullied[11] by computer programs – or, rather, by their commercial or political sponsors.

It is less obvious how one should try to affect the former context: what positive steps might one take to incorporate these insights into specific programs, so that they are somehow brought home to the nonspecialist user? This question would have practical importance even if the educational improvements just mentioned were already widespread, so that the average person had a reasonable understanding of programming and of programs. Since they are not, it is vital that informatics professionals think about how to write programs so as to minimize the mystification of the general public. How can the user be encouraged to consider the possibility that the data, preferences, and/or inferential strategies relied on by the program may in principle be faulty, and might be abandoned in favour of others? The basic principle is that one should include reminders, whether explicit or implicit, that at base the user is dealing with a non-objective, inherently challengeable, system. This might be done in a number of different ways.

For example, wherever appropriate one should try to incorporate credibility judgments into the program, taking care to make this feature of the program explicit for the sake of the user. Even without any knowledge of the criteria of plausibility involved, users may be prevented from seeing the program as the propounder of absolute truth by its use of such terms as "maybe,[11] "perhaps," "probably," "likely," "usually," and so on. (There is admittedly a danger, however, that such terms may give a spurious impression of judicious wisdom.)

Moreover, probability judgments should preferably not be expressed numerically in the output of the system, even if they are calculated numerically within its computations. For to the nonspecialist user, numbers and mathematical formulae have a supreme power of mystification. The ordinary language terms just mentioned should be used rather than numbers, perhaps with the option of requesting a numerical representation of the program's degree of confidence if this is specifically required. (Such a representation will not be interpreted by naive users in a purely Bayesian sense, for people's everyday judgments of probability are not Bayesian [33. Whether computers should always be programmed to reason in this way is a controversial matter.)

As for when credibility judgments are "appropriate," I suggest a broader interpretation than might be suggested on narrowly practical grounds, one which would take account of the epistemological nature of

the statement concerned. I would not recommend that we go so far as Karl Popper, according to whom even the circulation of the blood cannot be taken as a hard-and-fast truth [4]. But we would do well to favour a stronger version of fallibilism than we do in everyday life. That is, we should remind ourselves (the users of the program) that there are whole classes of judgment that are open to serious debate, quite apart from the epistemological scruples of professional philosophers.

For instance, I would not normally say that Shakespeare is "probably" England's greatest playwright, or that Californian summers are "likely" to be hot -- but perhaps a program should, especially in the former case where the (aesthetic) criteria of judgment are in principle more open to challenge. Nor would I normally say it is "very probable" that cigarette-smokers run a greater risk of lung cancer than other people do, for I regard this proposition as established beyond reasonable doubt; moreover, I prefer to err on the side of practical rather than philosophical caution here, dissuading people from smoking rather than leading them to risk running the fatal risk. This preference complicates the issue of what we should have our program say. If we know that the users may be turning to it for information on which to base their life-practices, we may be similarly loath to include the epistemologically pure "probably" in its output. If we are moved by such considerations, then programs for the doctor's use may differ from programs for the patient's use not just in the degree of technical jargon they tolerate but also in their epistemological profile. If we wish to avoid this sort of asymmetry, we should consider putting the phrase "very probably" into both programs rather than into neither.

It may be that such a policy would start by giving computer programs a specious air of hesitancy, and end by devaluing the currency of terms such as "probably" and "perhaps." So if this suggestion is to be taken seriously then we have to think carefully about which contexts can most usefully play host to "probable," where usefulness is judged in terms of alerting users to the essential fallibility of programs. In general, perhaps (sic: you are hereby invited to consider disagreeing with me), we should more readily mark the general computational functions of the program with these query-inviting qualifiers than attach them to specific propositions (like the one about smoking) which are not in practice regarded as dubious.

One way of doing this is to make heavy use of the word "evidence." That is, instead of the program's referring to the "facts," or even to its "data," it should refer to its "evidence." The legal connotations might play a useful role here, prompting the mind to conjure up visions of possibly innocent people at the mercy of tricky counsel, largely unreliable witnesses, and biassed and/or incompetent juries. The word "evidence" in the output allows for, and even invites, the possibility of the user's asking what the evidence is, on whose testimony it is based, and how the argument is supposed by the program to run on the basis of this evidence ("What makes you think, Mr. Holmes, that the cigar-ash is evidence of a woman's presence?"). Similarly, saying that the evidence "indicates" something is less compelling than saying that it "shows" it, so this verbal distinction might sometimes be used by the programmer to remind people of their epistemological responsibilities.

There are already programs which display their evidence and their reasoning, if asked, and which thus invite correction by the user. These are the so-called "expert systems," whose use in Western society

is steadily increasing and which, it has been suggested, would be worth introducing into Third World countries where the relevant human expertise is very rare. A good example is the MYCIN program, which diagnoses infectious illnesses and prescribes drug-therapy accordingly (taking into account the history of the individual patient); in addition, MYCIN asks for missing information it thinks might be relevant, so that if it is not available then this fact is brought to the attention of the user [5]. And some programs take account of the inferential histories of particular propositions when assessing their credibility. The weighing together of evidence and inference in the evaluation of credibility is a topic that merits much further study, not least because computer systems that indulge in it thereby display their essentially human epistemological fragility.

In everyday life, credibility judgments are made largely on the basis of the source of the statement we are being asked to believe, or on the nature of the authority that has undertaken to legitimize the evidence. That is, what really is evidence, still more what is conclusive or persuasive evidence, is in principle open to debate, and some authorities are regarded as more reliable (or, more reliable in certain contexts) than others. Sherlock Holmes' genius was in being able to legitimize apparently wild inferences, both by providing extra evidence (unnoticed by everyone else) and by reminding us of relevant background considerations that we all accept. Consequently, we are ready to believe his assurances even in cases where he does not supply this specific legitimization. But if Dr. Watson assured us that the cigar-ash denoted a female presence, we should -- quite reasonably -- be loath to believe him without explicit justification.

Accordingly, it is worth asking whether reminders of the legitimizing source relied on by the program might usefully be provided. It would be quite easy (though it would require legislation) to ensure that all explicitly political programs had their provenance clearly marked (just once, in the small print, or on every page?). But many "tutorial" programs would be implicitly ideological without their writers or sponsors even being aware of this fact (liberalism too is an ideology). How many tutorial programs in philosophy, sociology, economics, or civics, for example, would be -- or even could be -- free from all ideological bias? However, even if we were merely to provide continual reminders to the user that the program was written and/or sponsored by Mary Bloggs (as opposed to some known individual or group), this could serve as a reminder that ultimately the responsibility for assessment of the program lies in human hands -- not merely those of Mary Bloggs, but of the user too.

In cases where there is a recognized difference of opinion, both judgments might be offered. So MYCIN might sometimes say "In the opinion of Professor Smith, aureomycin should be prescribed, but according to Dr. Jones you should use gentocil." Then the users, doctors themselves, could choose one or other drug according to their evaluations of these two authorities. This sort of identification of advice-sources could of course lead to complicated issues of legal responsibility, but these will anyway have to be faced as advisory programs proliferate in the public domain. More blandly, a program might tell us "The consensus is that...," or "It is generally believed that...," adding that "Some authorities, however, disagree." Even if the precise nature and ground of disagreement were not accessible to the user, the fact of its existence would have been signalled.

A program for public use should also identify its financial sponsors or originators.This principle is followed in Parliament, where members have to "declare an interest" when they are speaking on matters with which they are financially involved. Other members are invited by this practice not to disbelieve them, but to bear in mind that – even though they are comparatively expert – they may be indulging certain biasses in speaking as they do, and that this possibility should be allowed for in assessing the credibility of what they say. Similarly, advertisements which look like ordinary text have to be marked as "Advertiser's Announcement." And it is often argued that research papers financed by the cigarette manufacturers should be clearly identified as such on publication, so that their findings (which funnily enough may cast doubt on the causal relation between smoking and cancer) can be assessed with this in mind. The adjudication between the accounts given by rival programs will certainly be no less difficult than it is between human experts, but at least we should avoid any program's being regarded as more expert just because it is a program.

The evidently "subjective" programs I have recommended could help to improve the general view of programs held by the public at large, and could also help prevent individual users from adopting an uncritical attitude toward individual programs. People would experience such systems as being less oppressive than programs of a more "objective" mien, since they would be less readily viewed as inflexible sources of control external to humankind. The White King's sense of helplessness and frustration, on having the unseen Alice write her own (not his) memorandum with his pencil, is comparable to that felt by people in the grip of an inhuman, inflexible system. In addition, programs of this type could contribute to the aim of designing "friendly" computing environments, in the sense that a clearly "subjective" program would probably be easier to interact with in familiar human ways than is an apparently objective, godlike system.

However, this brings me to a note of warning. We should remember that the natural world and human society are complex eco-systems, and that technological intervention may have counter-intuitive and damaging effects. For example, if we invent a toxin to kill 99% of the caterpillars who eat our cabbages, the predators who gobble up the caterpillars will all die out, and the remaining 1% of caterpillars will have no natural foes – result: no cabbages. And if we produce high-yield varieties of wheat which require intensive use of chemical fertilizers, the people who benefit most – indeed, the only people to benefit at all, aside from the chemical manufacturers – may be the landowners who can afford to introduce the necessary technology. In general, then, we should beware of or even deliberately avoid certain prima facie "improvements," because of the indirect effect they may have on the social fabric. More specifically, in making computers "friendly," we should beware of producing a friendliness dangerously akin to that which the wolf showed toward Red Riding Hood.

One of the unwelcome effects on our social relations that might result from widespread public familiarity with increasingly friendly computing environments is already apparent – and not only in specialist research laboratories, whose natural denizen the computer "hacker" has been so tellingly lampooned by Joseph Weizenbaum C63. Even more disturbing than Weizenbaum's caricature is a recent newspaper-article devoted to the unfortunate domestic effects of people's falling in love with their computers C7D. Anecdotal though it is, this article suggests

some grim consequences of making computing environments more seductive. It reports a letter from a lonely wife who thought her husband was having an affair with another woman when in fact he was spending his nights with his computer, and an interview with a one-time professional programmer happy to have escaped from his computer's clutches, who said "I was warned, when I started, that you should only spend a few years with computers. Otherwise, they take over. The divorce and alcoholic rate amongst the men who work in the field was higher than in anything else except astronauts."

Even this is perhaps only the beginning: anyone who doubts the social isolation and alienation that could result from overly accommodating computer systems, available in every living-room, should meditate on this quotation (which was not composed in an ironic or disapproving spirit): "it may be possible for intelligent machines of the future to supply not only intellectual stimulation or instruction, but also domestic and health care, social conversation, entertainment, companionship, and even physical gratification" [8].

It may be objected that people have always been fascinated by new toys and that, regrettable though this may sometimes be, it is no specific concern of the informatics professional. If the husband had not been making love to his computer he might have been sleeping with a vintage car, and who's to say which is the worse?

This objection misses an important point: the computer is a toy of quite a different order from any previous plaything, in its ability to satisfy -- or to appear to satisfy -- various deep-seated human needs. It offers us the illusion of total control, and it appears to give us its full attention, often being immediately responsive to every remark addressed to it. It can tease us with challenging questions, if we wish, but it never turns away in disgust at our stupidity (at least if its "error messages" are tactfully phrased). The emotional blandness of most programs -- Ken Colby's artificial paranoiac being an exception [9] -- means that they can be relied on not to vilify or swear at their interlocutors, nor to impugn their motives in making any particular remark. These flattering responses are rarely found in human beings, and not reliably even in dogs. If we like to appear a Napoleon to our four-footed friends, how much more we may relish appearing a Zeus to our computer. It is this which gives computers their insidiously lupine attractiveness.

In view of these points, the technological dream of a population of perfectly reliable programs, comprising the most amicable computing environment one could imagine, might be too much of a good thing. Of course nothing I have said counts against the sort of "friendliness" that is based in greater intelligibility, modifiability, and flexibility of programs [10]. But we should think twice before building speciously friendly (that is: apparently personal) characteristics into our programs. And we should remember that (prior to the sort of universal computer-education I have recommended) the epistemological characteristics ascribed to programs by non-specialist users are likely to be rather different from those ascribed to them by programmers.

This is why I earlier omitted to mention the most obvious way of signalling the subjectivity (non-objectivity) of individual programs, which would be to have them admit it, explicitly or implicitly, by the use of epistemological terms familiar in ordinary speech. If a program

were to print out "I believe that," "I am convinced that," "I doubt whether," "I would guess that," and the like, this would indeed remind the user that it is not infallible, that its ideas and inferential strategies are in principle open to question. (Most users would be well aware that, just because someone claims to be "certain" that something is the case, that it really is the case may nonetheless not be certain.) But these words would have an especially strong tendency to encourage the user to think of the machine as a quasi-human system, as something which they can describe in human terms and interact with in a familiar human fashion.

Programmers often have recourse to "plausibility tricks," by which I mean aspects of the program that make it look to the user as though it can do something which in fact it cannot. To some extent, of course, all programs for nonspecialist users will incorporate such tricks, for the details of the way in which the program works (and does not work) are deliberately hidden. But we should avoid excessive plausibility, and spuriously "personal" responses on the part of the program.

For example, we should be prepared to insert warnings to the user, making the computer's lack of knowledge or inferential capacity quite explicit. This sort of thing is done by human beings when someone says "Of course, I'm not a doctor, but... " or perhaps "You are not one of my patients, so I don't know your individual history, but ... " Explicit disclaimers about specific areas of expertise, with or without advice to seek the help of a human expert, could easily be written into programs, and the need for them fairly well predicted. What is less straightforward is the appropriateness of more general disclaimers, ranging from the vague "Don't forget I'm just a computer program" to the more specific "I know nothing about your political priorities," or "I can't make reliable guesses on that sort of question." Even the catch-all "I don't know" would be better than nothing, but it may mislead the innocent user by suggesting that if only the unknown item of information were supplied to the system then it would be able to make a sensible judgment on the matter.

As for the use of first names and colloquial expressions, it is irresponsibly mystifying to have the program greet naive users with remarks like "Hi, Maggie! Hope you're feeling fine today!" This sort of specious friendliness should be used very sparingly (for instance, only in the user's earliest encounters), if at all. If the conversation as a result is somewhat stilted, sobeit. In short, one's understandable impulses to produce systems that make the user feel comfortably at home, not to say goggle-eyed with astonished admiration, should be strictly curbed.

The overly anthropomorphic interpretation of programs by naive users is not only positively dangerous (in that it encourages the alienating substitition of computerized companionship for human conviviality), but negatively damaging too. That is to say, it prevents the computer from providing people with the possibility of interacting with a clearly impersonal system, in contexts where personal interactions may be experienced as oppressive.

For instance, human doctors are commonly perceived by patients as threatening in various ways, because of such factors as sex, class, age, personality, mood, or fatigue. It is perhaps not surprising, then, that diagnostic computer programs -- with which patients interact as though

fitting in a form by tetetype – have actuatty been preferred to human doctors, being described as more "patient[11] and "polite" than the average physician C113. Questionnaire-programs have eticited emotionatty taden topics from psychiatric patients with tess attendant anxiety than is aroused by human questioners C12D. And young chitdren can team to accept the chattenge of faiture in a constructive way, partty because the computer is not seen by them as a potentiatty disapproving adutt but as a toy with which to ptay around and have fun C13D.

In generat (as has been pointed out by Trevor Pateman), the sociat psychotogicat category of "ptay[11] hetps one appreciate the liberating potentiat of computer systems, to which (not to whom) one can direct remarks which (as in ptay) witt not be understood in their usual way, as carrying their usuat sociat consequences, because they witt not be understood at all C143. So it is highly appropriate to view computers as toys, more interestingly analogous to human minds than toy soldiers are, but unwise to assign them the role of surrogate friends since this may invite the sort of face-saving manoeuvres which, in interpersonal contexts, can inhibit the creative exploration of ideas. (Pateman argues that we therefore should avoid the use of psychological terms in describing programs. Certainly, one should always remember that any psychological term, from "subjective" to "knows" or "infers", is being used only metaphorically when applied to programs. But their functioning will inevitably be described –a fortiori by the non-specialist user – in quasi-psychological terms such as these, so we must do what we can to alert naive users to the differences between the implications carried by such terms in computing and in personal contexts.)

Other steps one might take, to keep the friendly wolf from the door, will require careful observation of the effects of widespread access to computing environments. For instance, research on the impact of such environments on young children's play-patterns is currently being planned [153. If any unwelcome changes in play-behaviour appear to ensue, then perhaps such effects can be forestalled in future cases. (We should not assume that any changes must be unwelcome: for example, the greater self-confidence that can follow a child's experience of computing might lead to less anti-social behaviour in playtime.) It might also be worth considering whether people's response to the words "probably," "perhaps," and "maybe" is in fact likely to be what I have suggested. For, instead of making them aware that the program outputting such terms is in principle open to challenge, this output might make them put even more trust in the judiciousness and wisdom of the system than before. This example shows that it is too early to be sure just what measures would be most helpful in preventing mystification of the user. But it is not too early to start thinking about these issues, and in the absence of widespread computer literacy it is important that we do so.

Last but not least, we should consider how one might exploit the potential that programmed systems have for facilitating interpersonal communication, between one human being and another. I am thinking not of computer-dating services and the like (which are not altogether to be sneered at), but of the potential for human interaction that is offered by a shared computing system – whether several users on one machine, or a network of machines. A survey of the use of the ARPA computer network (which is available only to highly specialist users) showed that a very large proportion of the messages in the net were not programs but personal mail. Whether or not this is pleasing to ARPA, it should be

pleasing to us. Certainly, computer mailing is no substitute for f to-face meetings, but it could be more convenient and so more often than are the telephone and postal services. And for people who housebound by illness or domestic responsibilities, the sense community and instant accessibility that is fostered by membership o shared computing environment could be liberating indeed.

In sum, the meeting between man and machine can if prop situated take place reassuringly near to the man's philosophical h ground, so avoiding the alien territory occupied by behaviourism and natural sciences. The best way to ensure that the meeting is prop situated is to provide the general public (viz., schoolchildren) suitable education about computers, and practical experience of numerical programming. In particular, the "subjective" – or, at le non-objective – nature of computer programs as representational sys must be stressed. This should be a top educational priority for future. But in addition, the specific man-machine dialogues ensuing this confrontation can, if prudently planned, serve to underscore ra than to undermine the subjectivity of both participants in encounter. Continual reminders that programs can in principle rebutted, argued against, mistrusted, or ignored – just as people and for largely similar reasons – could help to avoid the mystifica and sense of helplessness so common in people[1]s attitudes today, less mystification, the less threat. But perhaps we should delibera preserve a measure of threat, or at least rein in our understand enthusiasm for producing friendly – not to say seductive – compu environments. The wolf, after all, was friendly, and Red Riding would have been wise to have been more wary.

REFERENCES

1. For a detailed discussion of a program based on Freud's theory defence mechanisms, see Chapters 2 8 3 of M. A. Boden, Artifi Intelligence and Natural Man. Hassocks, Sussex: Harvester Pr 1977. For an outline suggestion - of how "dissociation consciousness" can be understood in computational terms, see Ch of M. A. Boden, Purposive Explanation in Psychology, Cambri Mass.: Harvard University Press, 1972.

2. Iann Barron £ Ray Curnow, The Future With Microelectror Forecasting the Effects o1 Information Technology, Milton Key Open University Press, 1979. P. 231.

3. E. H. Shortliffe & B. 6. Buchanan, "A Model of Inexact Reasoning Medicine," Mathematical Biosciences, 23 (1975), 351-379.

4. K. R. Popper, Conjectures and Refutations: The Growth jyf Scient Knowledge, London: Rout ledge & Kegan Paul, 1963.

5. E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN, York: Elsevier, 1976.

6. Joseph Weizenbaum, Computer Power and Human Reason: From Judgment

Calculation, San Francisco: Freeman, 1976. Pp. 115-126.

7. Jane McLoughlin, "The men who fall for their computers," The Guardian, Tuesday Dec 9th, 1980.

8. Oscar Firschein, M. A. Fischler, L. S. Coles, & J. M. Tenenbaum, "Forecasting and Assessing the Impact of Artificial Intelligence on Society," Proc. Third. Int. Joint Conf. Artif. Intell. (1973), P. 117.

9. K. M. Colby, Artificial Paranoia, New York: Pergamon, 1975.

10. H. T. Smith & T. R. G. Green, eds., Human Interaction with Computers, London: Academic Press, 1980.

11. W. I. Card, Mary Nicholson, G. P. Crean, Geoffrey Watkinson, C. R. Evans, Jackie Wilson, & Daphne Russell, "A Comparison of Doctor and Computer Interrogation of Patients," Int. J. Bio-Medical Computing, 5 (1974), 175-187.

12. G. R. Kiss, University of Warwick (personal communication).

13. Seymour Papert, Mindstorms: Children, Computers, and Powerful Ideas, Brighton, Sussex: Harvester Press, 1980.

14. Trevor Pateman, "Communicating with Computer Programs," Language and Communication, 1 (1981), 3-12.

15. Robert Hughes, Cambridge University Psychological Laboratory (personal communication).

oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo