A Semantics of User System Interaction.

Mike Holcombe,
Department of Computer Science,
Regent Court,
University of Sheffield,
Sheffield, S1 4DP.
U.K.

ABSTRACT.
This article examines the mathematical foundation for an important component of a formal conceptual model of a user. The fact that a software system can be formally specified with respect to its functionality is of particular value for several reasons; we can consider the possibility of designing experiments that have user tasks that are precisely defined, we can consider the possibility of deriving realistic mathematical results about user-system interactions and we can attempt to integrate the understanding that we may obtain about user behaviour into a system development methodology based on formal methods. This paper is concerned with the formulation of a model of an interaction as a, possibly goal-directed, experiment and the sort of structures that might be suitable for representing and reasoning about information from such interactions.

$1. Introduction.

The problems of defining a formal framework for the construction of models of system user-behaviour is addressed in this article and related work. This is an important area where more research is urgently needed. Many safety critical and enterprise critical systems 'fail' because of user errors caused by the user/operator failing to understand the system and taking inappropriate actions. Recently there have been incidents involving 'fly by wire' aircraft, atomic reactors etc involving errors of this type which have led to death or serious contamination.

Recent research into the behaviour of pilots of modern fighter aircraft has indicated that for much of the time they do not seem to understand the system. According to recent research involving the recording of cockpit conversations the most frequent utterance during flight is of the form “What does this display mean?” Clearly the user's conceptual model of the system is inadequate and does not match the model assumed by the designers. The increasing need for the use of formal methods in the design of safety-critical systems, e.g. the MOD Def-stands. 00-55 and 00-56 are forcing us to look at more rigorous bases for the analysis of all aspects of the system. The foundation on which to construct methods for the design and analysis of hardware, software and some environmental aspects of systems is now fairly well developed but the analysis of user activity is still lacking a really general and rigorous theory. It is this general area that we are trying to address by constructing a formal framework for modelling and analysing the relationships between the user's interactions and the system. In particular we regard each interaction as an experiment which will affect the user's model (and understanding) of the system.

There has been much work on developing theories, models and tools for Human Computer Interaction (HCI) (see Johnson 1992). While much research in HCI has dealt with the testing of psychological theories of user task behaviour such as the goals, operators, methods and selection (GOMS) approach of Card, Moran and Newell (1983) there has been less work of solving the problem of building a formal semantics of HCI. There have been formal specifications of user-interaction task scenarios such as that developed by Johnson (1992, p. 201) and Johnson

and Gikas (1993). They argue that the formalism gives an interface designer a more precise means of specifying and reasoning about his design. The formalism is developed in several stages. First, the temporal relationships between goals and subgoals are formalized. Next, procedures (actions) are formalized in terms of pre- and post-conditions. Finally, the various objects and their properties referred to in the structure are refined. Then, procedures (actions) are further refined in terms of objects and actions. They claim that by formalizing task structures in this way it is possible to check that the model is consistent and complete at all levels. The formal language used to describe the relationships between goals and subgoals and between procedures is a propositional temporal logic. However, it is difficult to see how their formal model of task structures can be generalised to incorporate a comprehensive formal model of user interaction involving data manipulation, user goals and user plans. Their formal model is more one which focuses on formalizing user goals rather than the complete process of HCI. Other formal methods techniques are described in Dix (1991) and Harrison and Thimbleby (1989).

We have already formulated some preliminary observations on the use of a general class of machine models for the description of arbitrary system behaviour [1] and these models, based on the idea of an X-machine, offer some promising advantages over other paradigms. Amongst the advantages are full generality, ease of mathematical analysis and flexibility,. Also, mechanisms exist for their integration into system development processes. This seems an appropriate basis on which to build a detailed analysis of how a user might interact with a system by means of experimentation and analysis.

We consider the general architecture of the user and the interface to consist of several interlinked features, including an experimentation interface through which actions are initiated and results received. Another component is a goal space which is regularly updated and which reflects the user's view of tasks to be considered with respect to the system. A mechanism for formulating experiments and reasoning about their results, both actual and possible, is also postulated and we do so without any assumptions about the logical processing that might drive this unit.

We are prepared to consider belief or fuzzy logics, perhaps embedded in a 'society of minds' framework as eloquently proposed by [2],[3]. Another component will involve some storage mechanism for experiment results and this could take the form of a traditional database or, more likely, some parallel associative memory engine. We assume that these components exist together with suitable communication channels between them.

In this article we describe the form of experiments that might be carried out by the user, define the way in which information might be stored and accessed, and examine the connection between experimentation, goal desires and realisation. Previous work on the use of X-machines for defining the framework of communication between users and systems has been covered in Holcombe,[1].

Our first task is to define the form of the data which comprises the basic experimental findings of the user during interaction with a system. It is common to regard the user's view of a system to be based on an input-output relationship with some rationalisation of the potential existence of "hidden states". Such an approach can be formalised in simple cases using finite state machine models. However, these do not provide a fully computational model of the situation and are thus vulnerable to objections thatthey may not be applicable in many situations. The X-machine model does satisfy this requirement and is a major reason why we have adopted it as a central concept in our overall model. From the point of view of the user there seems to be little evidence that the user is abstracting the system behaviour in terms of such a general class of machines but this does not mean that the model is unsuitable for our analysis. We will as-

sume that the user has a mental image of a fundamental data type which will be processed by the system; in a word processor, for example, this would be the user's idea of a document, other, more complex, models will be appropriate for other application domains. The user's data type will be called X_{user} and is a dynamic type which may be subject to revision during the user's experiences. There is clearly a possibility of side effects caused by a transformation of X_{user} into some other type and this is worth investigating on the grounds that it could be a cause of 'residual confusion' in the sense that 'old' beliefs may contradict 'new' ones under such a change and the resolution of such clashes may not be achieved immediately.

$2. Background : X-machines.

We will concentrate our attention on explicit input-output connections with only rudimentary system state awareness on the part of the user. Let X denote the fundamental data type of the system. What this means is that the processing performed by the system can be regarded as being defined by functions on X. The X-machine model of Eilenberg involves the identification of a set of 'basic' functions (or relations) on X which are used as the labels of transitions in the machine's state diagram. A computation is then modelled by composing together those basic functions that appear as labels on successful paths in the state space. A successful path is one from a designated initial state to a declared final state. Any specific input output processing is dealt with using encoding relations from a suitable input data type and decoding to a suitable output data type.

Definition. An X-machine is a 6-tuple of the form:

$$M = (X, Q, \Phi, F, I, T)$$

where:

X is the fundamental data type,;

Q is a set of system states;

Φ is a set of relations on X, that is $\Phi : \rho(X \leftrightarrow X)$ (where ρ denotes a power set) - we call Φ the type of *M);*

there is a next state function F: $Q \to (\Phi \to \rho Q)$ (which is saying that from a state $q \in Q$ there may be a set of arrows going to other states each labelled with a $\phi \in \Phi$);

$I \subset Q, T \subseteq Q$ describe the sets of initial and terminal states.

Essentially an X-machine looks like a finite state machine with one important exception, the labels of the arcs are functions that operate on a fundamental data type, X (actually the labels could be relations in theory, in practice we usually use functions). It is sometimes necessary to introduce distinct input and output sets which are encoded and decoded into the fundamental data type at the start and end of computations. For most applications, however, this is not required.

The way in which a computation is carried out is defined as follows:

select a start state (usually there is only one) and an initial value for the data type X;

construct a path from the initial state to a terminal state (in many examples every state is considered a terminal state and so all we are doing there is defining a path through the state space);

the path labels form a sequence of functions that operate on X; we apply them in turn to the initial value of X and end with a final value of X.

The transformation from the initial value $\tilde{x} \in X$ to the final value $\underset{\sim}{x} \in X$ represents the computation carried out by the machine. A specimen computation in a general X-machine is illustrated in Fig1.

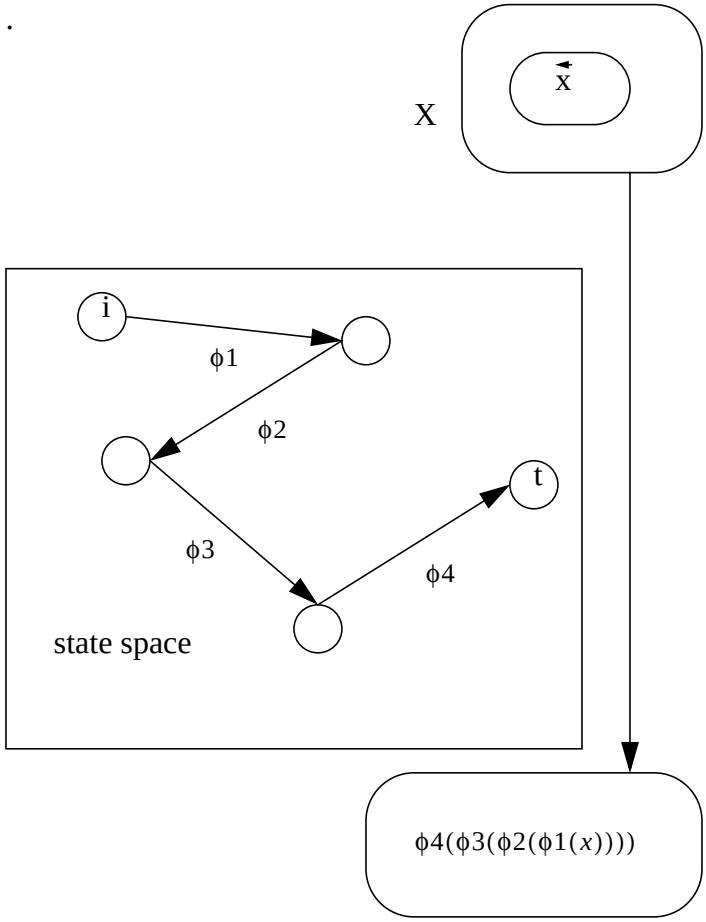


Fig.1. A typical X-machine computation.

One benefit of the model is the clear distinction that it makes between the control state of the system and the state of the data being processed at any particular moment. This is reflected in the way in which computations are defined by sequences of functions that are applicable in certain states only and to data satisfying specific conditions. Another benefit is that this model provides a fully general comoputational model as powerful as the Turing model. We will use this fact in our formalisms since we are convinced that any formal model of user interaction must be based within a rigorous computational framework. We have explored the use of X-machines in some depth for the specification of a number of system interfaces including a generic CASE tool (Duan & Holcombe,[4]) and a graphical, mouse driven diagram editor (Laycock,[5]). In both cases the power and elegance of the representation were confirmed. It is possible to construct functional verifications and test set generation based on the specifications.

A simple example of the kind of data type X that could be involved is obtained if we consider part of the interface of a word processing system. There is a type *Document* which describes the content and format of a document in the system. The user will have some model of this data type and will, we assume here, be carrying out reasoning on the basis of this model. The connection between the user's model of the document and the actual system type *Document*

will be the subject of some of our analysis. It involves modelling the connection between the user's model and the physical view that the user is given of the actual *Document* type through the available output displays. The treatment of the two types of state, the system state which defines which function is available for example ’file transfer’, ’edit’,’print’ etc. (indicated in Fig.2) or the data state which is the exact form of the document at that moment is both clear and convenient. In some cases the state of the control is determined from aspects of the screen display, for example a highlighted menu item, an open window in the foreground etc. and the state of the data is also indicated, in part, by the display, such as the contents of a window or area of screen which might be text in the case of a document or an annotated diagram in the case of a diagram editor. The role of the *Display* type is thus crucial as is its link with the two state types.

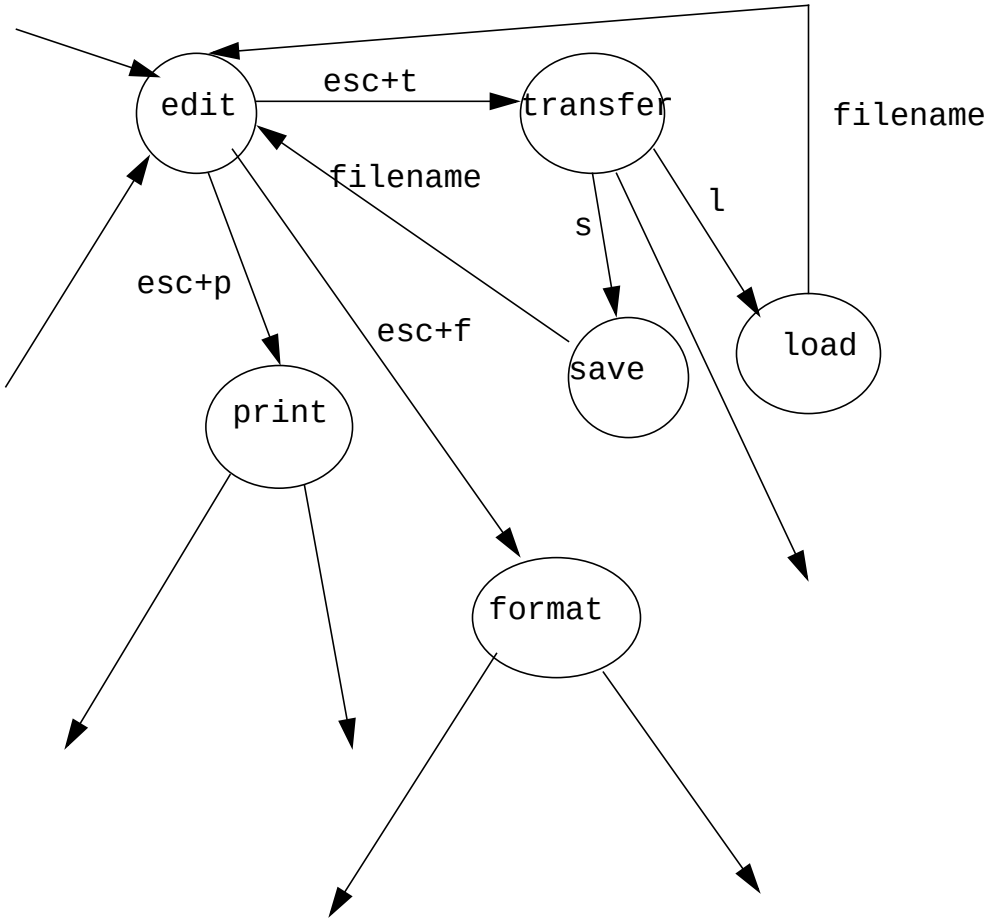


Fig.2. Part of the interface of a popular word processor.

The machine computes over the following data type X:

First let Char denote the set of all characters including space,

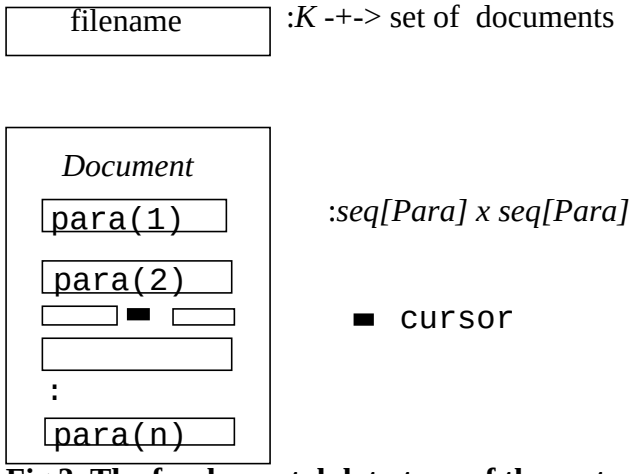


Fig.3. The fundamental data type of the system in Fig.2

punctuation, end_of-Line etc. Let *Font* denote the set of possible font types {bold+italic+times, standard+straight+courier, etc.}.

A document will consist of a number of paragraphs with the cursor located somewhere in the middle of the document. Let *Para = Seq[Char x Font]* denote the set of all sequences of character-font pairs.The *Document* type is then a pair of sequences of *Para* with the interpretation that the cursor is positioned at the end of the first sequence of paragraphs.The filing system consists of a partial function from *K* to the set of documents, where *K* is the set of filenames. The Data type X is then the set consisting of:

[K-+->set of documents] x [seq[Para] x seq[Para]] which corresponds to the diagram in Fig.3.

Suppose that a user wishes to transform a document so that *para(i)* is interchanged with *para(j),* where *i* and *j* are different and *para(i), para(j)* $\in$ *Para*. This is a standard "cut and paste" activity. If the state of the document is defined by an element of the set X then the "goal" of the user is to transform the state to another state with the two paragraphs interchanged, thus, if initially we have:

x = (d,(<para(1),para(2),..,para(i),..,para(n)> <para(n+1),..,para(j),..para(m)>)) $\in$ *X*
where *d* $\in$ *[K-+->set of document],*
<para(1),para(2),..,para(i),..,para(n)>, <para(n+1),..,para(j),..para(m)> $\in$ *[seq[Para]x seq[Para]*

(assuming that the cursor lies between the two paragraphs to be switched, it is simple to adjust the representation otherwise).

The intended state of X is then:

x' = (d,(<para(1),para(2),..,para(j),..,para(n)>,
<para(n+1),..,para(i),..para(m)>))

Thus we can indicate the desired goal of the user to be a function (possibly partial) that is defined from X to X. To attain the goal the user has to find a path through the machine's state

space with label $\phi_1, \phi_2 ...,\phi_n$ whose composition, as a function, equals the goal function. This is our fundamental representation of goal semantics.

$3. Fundamental interaction structures.

Consider the type "*Displays*" as defining the possible interface information available to a user during system activity, for example possible screen displays etc. There is a function (or relation) between X and the state set Q, and *Displays*, which is intended to provide the user with the appropriate information to allow them to deduce the state of the system processing, both control state and data state.

We could make an initial assumption that the set *Displays* has a particular architecture that can be described by using a collection of specific data structures which could include:
graphical objects such as windows, icons and other shapes, text of different styles often described using sets of suitbale sequences and so on. The overall architecture could then be derfined as an array or a finite product of basic sets of graphical objects or sequences of symbols. Thus

$Displays = A_1^* \times A_2^* \times ... \times A_n^* \times B_1 \times B_2 \times ... \times B_m$

where $A_1, A_2, ..,A_n,B_1,B_2,..B_m$ are given output types the *A*'s generating sequences and the *B*'s being of other types. The specific strucure of this set *Displays* and the relationships between the constituent parts will be described by various mathematical relationships, but the basic concept is to regard the set *Displays* as having a precise mathematical structure about which we can reason. Thus, the syntactic shape will be defined by the type $A_1^* \times A_2^* \times ... \times A_n^* \times B_1 \times B_2 \times ... \times B_m$ together with a semantics inherited from the individual semantics of $A_1, A_2, ..,A_n,B_1,B_2,..B_m$ and the interrelationships between them mentioned above.

Naturally, the semantics of *Displays* should reflect, as far as is appropriate, the state space semantics, that is there could be a natural sequence of states which are generally followed in interactions. There are thus a number of possible display types that might be involved in the type *Displays* giving information concerned with the control state and the data state which are of a graphical nature.The complete set *Displays* could thus have a rich mathematical structure which may provide vital information during the formal analyis of user system interaction and which shoould be considered in any theory of interaction. It is sometimes convenient to separate out the aspects of *Displays* that indicate the current system state and the state of the data being processed. This can be done in a variety of ways, the advantage of the X-machine model is the clear and flexible ways in which this can be done.
Thus we postulate the existence of a function

$H : X \times Q \text{ -+-> } Displays$

which is a fundamental part of the interface. It represents the way in which the current state of the data type and the control of the system is displayed to the user. The usual projection functions $p_1: X \times Q \text{ --> } X$ and $p_2 : X \times Q \text{ --> } Q$ will be assumed. From this function we assume another function that allows the user to infer the conceived state of the processing from the state of the *Displays*. This function will probably vary with time and the user's understanding of the system. So we postulate the existence of a function of the form

$infer: Displays \text{ -+-> } X_{user} \times Q_{user}$

where Q_{user} denotes the user's conceptualisation of the system state space. (It might eventually be regarded as a fuzzy function i.e. a set of functions accompanied by a measure of the certainty that the user attaches to this function being a correct transformation.)
Note that we do not assume that the users view of the data, X_{user} is the same as the actual system data type, X, nor that the user view of the system state is the actual one. It is conceivable, for example, that the user may have a very unclear understanding of the system state and only through interaction will it become apparent that the system reacts differently to similar actions thus demonstrasting the existence of internal state. By reasoning that the same actions can have different resuls the user will have started to recognise the existence of a state space and this would be reflected in the formulation of a, probably very incomplete, user model of a dynamic system whereby part of the Displays information is state based. We will identify the users model of state and of data by constructing a dynamic model of the form $X_{user} \times Q_{user}$. First we note the fundamental link between the user's actions and goals can be described in a commutative diagram:

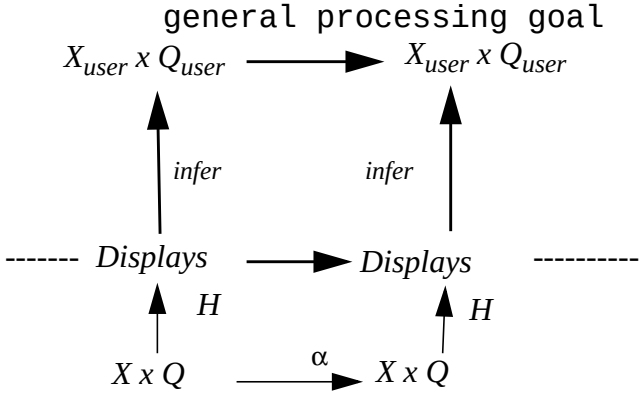


Fig.4. A general description of user processing.

Now the interpretation of this is that in certain control states the user will initiate actions which directly or indirectly test the model of the system and the assumptions that the user may have about the actions required to satisfy a given goal. A general processing goal takes into account the initial state of the data and of the system as perceived by the user and transforms them into new perceived states. A goal, however is regarded as a function (or more generally, a relation) on the conceptual data type X_{user} only. This interpretation relies on a model of the *infer* function and its range type and of the relationship between a goal and an action sequence. The subsequent comparison of the resultant inferred processing changes with the predicted changes, i.e. the processing goal, is then available and can be used for revising the user's model of the *infer* function, the X_{user} type or value, the effect of the action sequence or the goal definition.To provide a complete analysis of the situation we need to bring in the true system data state X somehow but we are not quite ready for that yet. For the moment the connection with the true data state is through *infer* and the *H* function. To extract the data state from the user's combined model of system and data state we use a projection function $p_1 : X_{user} \times Q_{user} \dashrightarrow X_{user}$.
Much has been written about the representation of such information, the processes of updating and analysis that might occur and the type of logical framework (probabilistic, fuzzy or other type of belief logic) that is the basis of such activity. We will make only minimal assumptions about these matters and concentrate, instead, on the formal frameworks in which such discussions might be placed. The processes of formulating experiments, analysing results and reasoning about their consequences for the user's conceptual model of the system are fundamental on-

going activities. We begin to examine these next, albeit in a few simple basic cases. Two basic types of activity are recognised in this first analysis and another aspect dealt with later. We start with a few thoughts about the nature and purpose of simple interactions. What does an interaction involve? There are at least two basic activities to be considered;
ACT1. formulating and understanding models of X_{user} and Q_{user} and postulating the form of infer,
ACT2. constructing action sequences to satisfy a goal.

Under certain conditions the user will also be doing other tasks which may need system exploration and experimentation including the analysis of possible relationships between goals, in particular, the relationships between new goals and established goals. Consequently we will need to formulate a mechanism for describing each of these activities as well as provide a procedure to store and access information relating to such interaction data. We will consider these activities in turn.
ACT1. We will assume that the user obtains information about the system data type through observing the behaviour of the system under certain interactions (actual or virtual) initiated from an appropriate control state. Let us assume that we start with an initial model, Y_0, of X_{user} and an assumed *infer* function together with a goal *g* and an action sequence α that satisfies that goal.
If the display is initially $d = p_1(H(x,q)) \in Displays$, where $(x,q) \in X \times Q$, then we expect,
$g(p_1(infer(H(x,q))) = p_1(infer(H(\alpha(x,q)))$ and $p_1(infer(H(x,q))) \in Y_0$.
Failure of this relationship causes the questioning of the assumed model Y_0 and/or the definition of *infer*. It could also raise doubts about the other assumptions which might invoke other investigations. The precise nature of the updating of the model of X_{user} and *infer* is not clear and must, to a large extent, depend on the application domain and the nature of the possible data models.

ACT2. To discuss the formulation of action sequences to satisfy a given goal will involve many different processes, not least of which will be the examination of relationships between various goals. We will examine this later and restrict our attention now to summarising the basic points.
During the course of an interaction the user will receive the information contained in the current *Displays* state and formulate a strategy for the satisfaction of some processing goal in such a way that the result of the sequence of user actions initiated from an appropriate control state causes a transformation of the *Displays* state which is compatible with the inference function infer. Thus the user will be seeking to complete the commutative diagram by constructing a suitable 'actions' function during ACT2. Here α is the system processing function defined by the actions of the user initiated from the control state. This function is dependent on both the state of the system (Q) and of the data (X) and can produce changes in both. Part of the process that we have not yet made explicit is the attempted deduction by the user of the appropriate state that the system needs to get into to make the desired processing possible. The user's view of the set of possible states is likely to be different from Q and is also subject to change over time, we call it Q_{user}. This is an added complication that needs to be dealt with eventually. Furthermore, there will be a need to recognise that a certain goal can only be achieved from certain control states and reaching these is a subsidiary problem

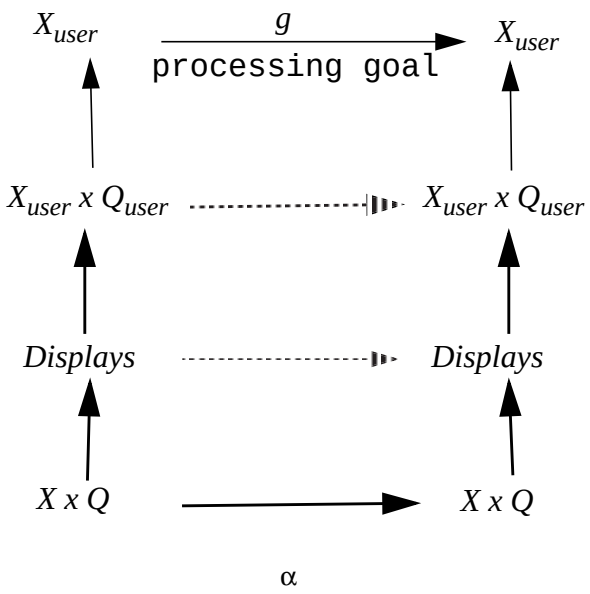


Fig. 5. ACT2 processes in diagrammatic form.

The role of the interface designer is to make informed predictions about possible models of X_{user} and *infer* and to construct the disp function and the *Displays* set to allow for the most effective user correction and learning processes.

An important point to consider, as before, is the sort of information that is being used here during this activity. The user has an assumed goal g, an *infer* function and a display and supplies an action sequence α. The result is either apparent success or failure depending on whether the last diagram commutes appropriately. The results of such interactions are stored in some data space as we shall discuss shortly. The possible form of storage of information derived from actions is considered in $4.

Given a goal, $g:X_{user}\text{-->}X_{user}$, there are certain values of X_{user} for which the goal is not meaningful, for example interchanging two paragraphs in a document with only one or none. In thses cases the underlyng user data types should not be altered, thus $g(x)=x$, This leads to the definition of the scope of a goal.

Define $scope(g) = \{x \in X_{user} \mid g(x)=x\}$

Given $g:X_{user}\text{-->}X_{user}$ then the user's attempts at achieving this goal consists of the construction of a state $q \in Q_{user}$ and an action sequence $\beta_1,...\beta_m \in$ Actions which satisfies the condition: $\forall x \in scope(g), \exists x^* \in X, q^* \in Q$ such that

$$inferH(x^*,q^*) = (x,q)$$

and

$$p_1(infers(\beta_m..\beta_1)(x^*,q^*)) = g(x).$$

So we define, for each goal $g:X_{user}\text{-->}X_{user}$, and each $x \in scope(g)$ a pair:

$(q, \beta_1,...\beta_m) \in Q_{user}$ x $Actions^*$ satisfying the condition above. We call this pair the "state action pair" for *g*. The user's belief in different mechanisms for achieving a goal *g* will

partly be concerned with attaching strengths of belief to such state action pairs and revising these over the course of time. Given a goal *g* the user may recognise the existence of a necessary initial control state needed to achieve that goal. Entering this initial state, which we will call $IS(g) \in Q_{user}$, for the goal *g* will be a major subsidiary aim of the goal.

The user's plan for the achievement of a specific goal $g{:}X_{user}\text{-->}X_{user}$ can be formulated in the following algorithm:

```
initial environment : Xuser, Quser;
goal | g:Xuser->Xuser;
initial state : q0 ∈ Quser, x0 ∈ Xuser;
{if x0 ∈ scope(g)
then
{find a valid action sequence a1,a2,...,an ∈ Access such that q0a1a2...an=IS(g);
 construct a state action pair (IS(g), b1b2...bm) for g;
if{p1(infers(bm...b2b1)(x',IS(g))=g(x0)
where inferH(x',q')=(x0,IS(g)
then reinforce belief weighting}
else ???
}
        endif
else
do nothing
end.
```

Suppose that a given goal $g{:}X_{user}\text{-->}X_{user}$ of the user corresponds to a particular processing goal $g'{:}X\text{-->}X$, identified by the designers as being achievable through the action sequence α' starting from a given data state x and control state q. If the infer function is faulty then we can expect the action sequence α used by the user to differ from α' and the result of the action different from the intended goal state *g'(x).* This may, or may not, be apparent to the user and in particular the design of the *H: X x Q --> Displays* function is critical here. If, for example $H(\alpha(x,q))=H\alpha'(x,q)$, then there is no evidence for the user to assume that the correct result has not been achieved. If the user's goal, $g{:}X\text{-->}X$ is not achievable from the particular system state (x,q) then there ought to be some mechanism for informing the user of this possibility. There are many opportunities for systematically analysing possible action sequences, their achieving of basic system goals and the amount of information available to the user through the displays. It is conceivable that a theory could be developed which looks at the differences between expected user actions to satisfy a goal and the intended actions expected by the user. A sophisticated system could indulge in some training interactions with a user by asking them to carry out certain standard tasks (goals), analyses the differences between actual and expected action sequences and dynamically alter the system interface, either by providing corrective informaton, better display information or even (although this may be a little far-fetched at present) regenerate the interface architecture to allow the user's action sequence for the particular goal. We present these thoughts as possibilities once a formal model of user interaction and experimentation has been developed.

$3. Fundamental assumptions about experimentation.

We have described the broad strategic outline for our model and we will now embed this into a framework for the acquisition and storage of information obtained from interactions with the system. The role of the control state is perhaps worthy of brief mention, first. Part of the action sequence input during an interaction is aimed at the accessing of certain positions in the system hierarchy at which particular processing operations are available. This might be illustrated by the activity of traversing a series of nested menus or something similar. The definition of an X-machine makes this concept explicit and further details are to be found in Holcombe [1].

Before we go any further we will indicate some of our basic simplifying assumptions.
1) 'Knowledge' about a system is obtained as a result of experiments carried out by the user, or by deductions made according to the user's belief system based on the experiments made. We include amongst experiments the use of manuals and 'expert' guidance so that a user may not physically carry out a specific experiment directly; but naturally such secondhand information may not receive the same 'weight' in the belief system as first hand information.
2) Experiments can involve the following assumptions:

user observes the current display state to be *Displays*;

user assumes that the fundamental data type is $x \in X_{user}$;

user assumes that the inference function is *infer: Displays* --> X_{user};

the context of the experiment is *c*;

user has a goal *g* to satisfy;

user supplies an input sequence $\alpha \in$ *Actions*;

user observes the apparent output of ß $\in$ *Displays*;

Note that X_{user} and *infer* will change over time but, possibly, only slowly.

3) All experiments begin with the system in a single start state. This can be relaxed later to include the use of 'common' start states in lower level basic command spaces in the system.

4) The input is considered to be a function that transforms the system's fundamental data type representation *X* in some way. The input itself could be a sequence taken from some suitable input set, *Actions*, of command and input actions open to the user. The formal description of this language of inputs will depend on the system in question, we assume that this can be done so that legal input strings are well-defined with a precise semantics available, defined in terms of the functional effect of the sequence on *X*. The user will have a conceptual model of what these actions do to the conceptual data type model X_{user} which satisfies the diagram defined in ACT2.

5) There is a set of relevant 'contexts' within which some experiments are carried out. This could include the fact that a specific goal for the experiment exists but it could also denote other 'modes' of experimentation such as random system exploration.

6) The user 'forgets' information at a rate dependent on time and on the context, to allow for different experiments being perceived to have different import or relevance.

7) The system under investigation is deterministic. This may be relaxed in later work.

8) Experiments are carried out one at a time.

There are some remarks that should be made at this stage; the set of contexts under discussion is a dynamic set that will change with time and encapsulates all of the relevant contexts related to the specific experiment. These could involve, for example, the goal or set of goals under consideration at the time as well as 'similar' or related experiments carried out earlier. Since goals are explicitly modelled in the experiments we will remove them from consideration in the field of contexts. Thus relevant context information might include the physical and mental state of the user, the degree of task concentration etc. and environmental influences. At this stage we will simply observe that such a concept as a context exists and in some applications will be important but perhaps specific to that application.

$4. Data storage.

Now we consider the way in which we may store information about the interactions as they take place.

We postulate that data is stored in a network consisting of specific interaction instances linked in such a way as to provide a rich semantic structure that reflects the perceived relationships that hold between the elements. Thus we store related information grouped together according to some meaningful relationship in some sort of network of connections which may or may not be a semantic network. What these relationships are may be open to debate but it must be emphasised that likely principles, such as proximity in time of interaction, similarity of goal, type of display values experienced etc. could be important. What is likely is that the information will be stored together with a number of 'pointers' which indicate perceived relationships between interaction instances and the associated data.

Each interaction could be stored as an array recording the parameters together with the result and appropriate pointers to other elements of the data space which, for some reason, have been connected with that interaction. These data space elements will be indexed with the time of the interaction purely for convenience and to provide a mechanism during recall processing that can determine the time that has elapsed since the information was obtained. This is important where memory degradation is concerned. Accordingly it is useful to denote the value of, say the *infer* function at time *t* by $infer_t$ etc. The use of a purely temporal logic for this analysis would be inadequate because the period between interactions is important, not just their relative temporal positions.

The result of an *ACT2* interaction involving the initial data state $y \in X_{user}$, $\alpha \in$ *Actions*, $g \in$ *Goals*, $\tau \in$ *Displays* and the *infer* function will be added to the database together with connections between this interaction and previous interactions in the database modulated by suitable connection strengths representing the belief strength of the connection. The sets *Goals* and *Actions* are the sets of all possible goals and action sequences respectively. A similar process will ensue for other actions.

A further aspect of the model is the recall and manipulation of this type of experimental data. To allow for the possibility that data in the long term memory degrades and becomes more difficult to recall accurately we postulate the existence of a set of simple decay functions

$r_t : R^+ \dashrightarrow [0, 1]$ for each $t \in R^+$ which starts decaying at the time t when the observation was made. We note that the following conditions may apply to these functions, the exact form of which are not specified here,

$r_t(0) = 1; \quad \lim_{y \to \infty} r_t(y) = 0;$

$y << y' \Longrightarrow r_t(y) \geq r_t(y')$ for all $y, y', t \in R^+$.

Included in this function r_t might be a 'concentration factor' which incorporates a measure of the certainty attached to the results by the user. This factor can be influenced by the unexpectedness of the results, the level of concentration of the user at the time or even physical and mental alertness and is reflected in the context, c_t, at the time.

The data space of interaction information is thus organised in terms of the two basic types of *ACT* and the data that these 'experiments' involve together with a memory degradation function, the time of the interaction and connections of various types to other elements in the data space and the current state of the goal space.

It is likely that, at some stage, we will consider a neural network representation of this information but for the moment we will proceed in an abstract mathematical fashion.

The simple operations on the basic data spaces involving the updating of the data space and goal space are not, in themselves, described in detail here and we do not intend to consider all of the activity which might be found in the user on the completion of a simple experiment on the system and the intimately connected activity in the goal space which will affect and be affected by experimentation. At a later stage, some formal descriptions of database operations needed to update and process the memory store of user experiments will be made. We are not, at this moment, providing algorithms for the implementation of these functions, nor do we consider the implementation of such a memory store. It is, however, worth noting that the use of associative memories and neural processing architectures would seem to be a natural avenue for further exploration.

Specific access functions could be defined in terms of a small set of basic recall functions, for example we will expect the user to attempt to recall an experiment by supplying various parameters and the ease of recall will be determined by the quantity and quality of this parametric information moderated by the state of the memory decay at the time of the attempted access. The formal definition of these functions will involve the definition of these parameters (in the function declaration) and a search through the data space.

The data space could be partitioned into a union of possibly disjoint subspaces which are defined by the state of the *infer* function and X_{user} type at the time of the interaction. Each component will have a set of possible relationships with other components and within each component there will be substructures describing subsets of related interactions.

Suppose that at time t the data space is D_t and this is composed of the subsets

$D_t = Z_1 \cup .... \cup Z_n$

where each Z_i is a set of observations made under the same assumptions concerning *infer* and X_{user}. We call Z_i the understanding *i*. Then we can define, for each *i* and $g \in G_t$ the set of all interactions carried out in the furtherance of goal *g* up to that time. Denote this by $Z_i[g]$ and call it the interaction history defined by *g* at understanding *i*. For each set of the form $Z_i[g]$ we can decompose further and define, for each action sequence $\alpha \in Actions$, $Z_i[g,\alpha]$ to be the set of interactions involving the application of α in the furtherance of *g*. The rich semantic relationships between different action sequences and between goals will be reflected in relationships between these subsets. A topological description of this data space is thus a promising possibility.

In the case of an *ACT1* operation the process of revising the *infer* function or X_{user} will result in the creation of a new Z_i understanding and the possible transfer to this set of some previous interaction data, although possibly in an incomplete way. At the same time the networks of relationships will be revised.

$5.The classification of actions.

It is clear that the set of possible actions open to the user has an important structure that needs to be examined before any general conclusions can be reached about the relationships between actions and goals. The concept of an X-machine is vital here. Let us assume that we have an X-machine representation of the system and that the start state is q_0. We first make some general definitions concerned with the nature of user actions.

Definition. The set, *Actions*, of user actions is partitioned into three mutually exclusive sets:

$$Actions = Access \cup Commands \cup Entry$$

where:

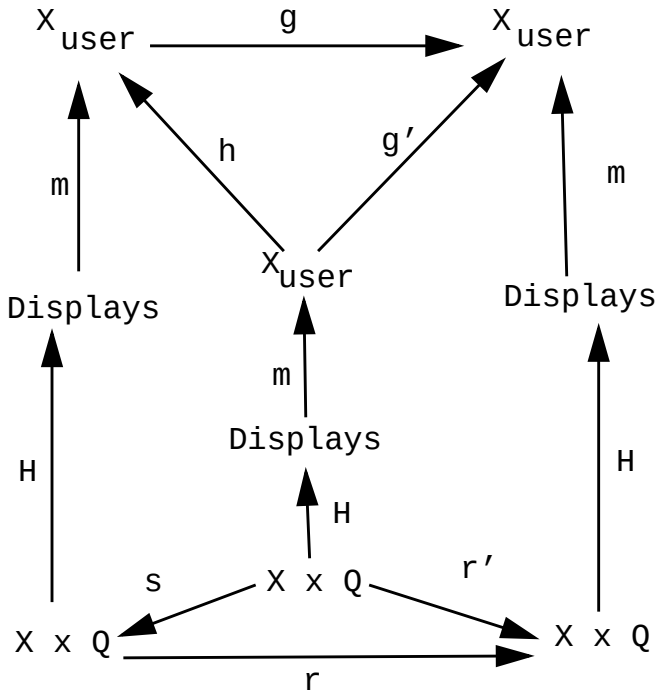
Access consists of all relations defined on the data type X which behave as state change operations without altering the data state $x \in X$ that exists at the start of the action,

Commands is the set of all system relations that carry out some type of processing of the data type element $x \in X$ but which do not involve the entry of data information,

Entry consists of all actions that are involved with the entry of a specific data element of a given type.

Thus *Access* actions could, for example, be menu choices that prepare the system for a specific command application by changing the control state of the system; *Commands* carry out some operation on the underlying data, such as deleting a paragraph in a word processor etc.; and *Entry* actions involve the supply of values of the correct type required by the system according to its current state.

The connection between the attempt to deduce a plan and the achievement of a goal is dependent on the complex relationship between the actual state of the system and the actual state of the data type X; and the perceived states of the system and the data. The link here are the *display* and the *infer* functions. We can postulate the following diagram:



where *m*: *Displays* --> X_{user} is the function that allows the user to deduce the data state from the display information formed by projecting *infer* onto X_{user}.[p and p' are functions that allow for the decomposition of the goals into subclass which may involve the transformation of the system into an appropriate state for the desired processing. r and r' represent the actual processing function defined by p and p'.] Thus the user is in a given state $(x,q) \in XxQ$knows that to carry out goal *g'* from the current perceived state *m(H(x,q))* the sequence *r'* is performed. To establishe the required sequence of actions *s* followed by *r* a subgoal *g* of *g'* has to be identified with known action sequences *r* and *s* such that

$$m(H(s(x,q))) = h(m(H(x,q)));$$
$$r(s(x,q)) = r'(x,q);$$
$$m(H(r'(x,q))) = g'(m(H(x,q)));$$
$$g(m(H(x,q))) = m(H(r(x,q)));$$
$$goh(m(H(x,q))) = g(m(H(s(x,q))) = mH(r(s(x,q)))$$

where $g'=g_o h$, $r'=r_o s$.

The user's model is defined to be *sequentially consistent* if the above diagram is commutative (ie. $mH(r(s(x,q))) = m(H(r'(x,q)))$)

If *Hom* is injective then $g'=gh \Rightarrow r'=rs$. This leads to the definition that the user's model is *consistent* if *Hom* is injective.

Now consider the following relationship $gomoH= moHor$ from the diagram, for *r* to be well defined we must have *H* and *m* invertible.

These issues are important if we are to understand better the realtionship between the model of the system according to the current user and the actual system as designed.

Our next issue is concerned with the classification of types of input.

Definition. An action sequence $\alpha = a_1...a_n \in Actions$ is said to be valid iff

$$q_0.a_1...a_i \neq q_0$$

and

$$q_0.a_1...a_i \neq \emptyset \text{ for } i \in \{1,...,n\}.$$

The second condition means that each action can be properly applied to the system, in other words each state $q_0.a_1...a_i$ lies in the domain of a_{i+1}.

Definition. An action sequence ß is applicable to an action sequence α if αß is valid.

The next concept is concerned with the identification of action sequences which are identical up to the entry of different instances of a specific data type element.

Definition. Two entry actions μ and μ' of the same type are said to be related, written μ ≈ μ', if
for any α ∈ Actions, μ is applicable to α iff μ' is applicable to α.

Definition. Two Action sequences α, α' are said to be congruent, written α ≅ α', iff for any decompositions α = ßμσ , α'= ßμ'σ with μ,μ' ∈ Entry and ß,σ ∈ Actions then μ ≈ μ'.

This concept is meant to identify those action sequences which differ only by the data values entered during the course of the action sequence.

The importance of this idea will be found in the analysis of the goal space and its relationship with action sequences that might realise a given goal. The system designer must ensure that failure to carry out a desired action sequence in the pursuit of a goal does not result in unfortunate side effects and a precise analysis of the X-machine model of the system interface will provide a mechanism for this task. Identifying the valid action sequences and protecting the user from the consequences of inadequate conceptual models or of simple clumsiness will help to ensure the design of a safe system.

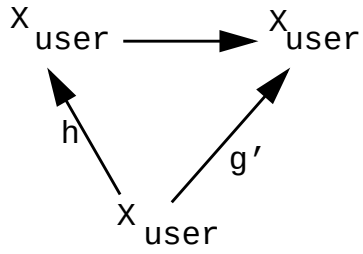
$6.The dynamics of the goal space.

An important component of the system is a dynamic collection of explicit user goals. These constitute part of the goal space. As described above, we propose to model these goals as relations on the user's data type X_{user}

$$g : X_{user} \dashrightarrow X_{user}$$

and remark that the goal space will have a close relationship with the set of contexts.

Definition. Let g, g' be goals i.e. relations on X_{user} then g is a subgoal of g', written g ≤ g', if ∃ $h : X_{user} \dashrightarrow X_{user}$
such that h g = g' or in terms of commutative diagrams



Under such conditions g' is also called a *supergoal* of g.
The purpose of introducing these concepts is to deal with the decomposition of goals into simpler goals which might constitute one approach taken by users in trying to satisfy goals. Other relationships probably occur and the sort of assumptions made by users and how degrees of certainty are attached to possible goal relationships come later.

New goals are received as environmental inputs from time to time and so the set of user goals is also a dynamic set.

We expect there to be a variety of relationships which describe certain interconnections that may exist between some goals. It has already been pointed out that the subgoal and supergoal relationships would be natural concepts to investigate. Others are likely to arise also. We thus model the goal space at time t as follows;

$G_t \subseteq$ Rel X_{user}, the set of all relations on X_{user}, and for each t and G_t we define a set H_t of relations on G_t, so that

$H_t \subseteq$ Rel G_t,

subject to the conditions

$$t < t' \Rightarrow G_t \subseteq G_{t'} \wedge H_t \subseteq H_{t'}$$

where we use the natural extension of relations on G_t to $G_{t'}$.
Thus the H_t describe the perceived relationships, if any, between the conscious goals at a given time t.

This means that goals are not lost to the system, although at some time it may not be possible to consciously recall them, and that the goal space and its relationships 'grows' with time.

We will represent the goal space G_t at time t as a directed labelled graph with the elements of G_t as the nodes and a labelled arrow

$$g_t \xrightarrow{h_t/s_t} g'_t$$

under the conditions when $(g_t, g'_t) \in h_t$, and $h_t \in H_t$; the value $s_t \in [0,1]$ represents the 'strength' of this relationship in the sense that it measures the belief of the user in the relationship existing, so that
$s_t = 1$ indicates the strongest possible belief in $(g_t, g'_t) \in h_t$ and $s_t = 0$ is when it is known that (g_t, g'_t) does not belong to h_t.

Naturally in the latter case no connection would be indicated in any diagrammatic representation of the goal space.

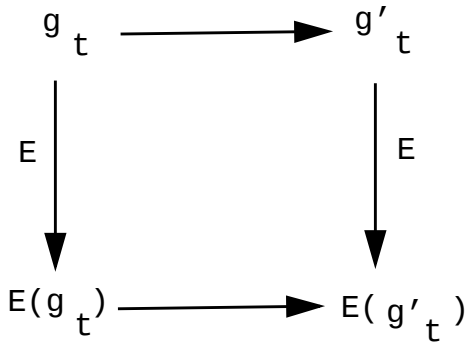
If we consider the sort of recall operations that might exist in the database we will recognise that much of the problem of describing goal realisation processes is concerned with these ac-

tivities. The relationships that exist between the goals and the action sequences in the database can be formalised in several ways.

Suppose that we consider goal $g_t \in G_t$ then there exists a set of events

$E(g_t) = \{ (\alpha, y) \mid \alpha$ is an action sequence applied to conceptual data state $y \in X_{user}$ in the successful furtherance of goal g_t and providing that degradation has not reduced the memory below a given threshold value$\}$ - starting from a fixed initial state q_0.

So $E(g_t)$ is the set of relevant recallable events associated with goal g_t.(If we regard the experiments as labels of nodes in some network which describes all of the apparent connections between goals then $E(g_t)$ gives all those nodes that can be recalled and are definitely believed to have resulted in the achievement of the goal g_t. Note that the set $E(g_t)$ involves all previous interactions and is not restricted to those interactions that belong to a specific understanding, Z_i.) Further relationships can be defined that link related goals with related action sequence event sets. Thus function diagrams of the form



will provide a suitable means of moving between the analysis of relations on the goal space and relations on the event sets.

During the process of searching the goal space for related goals the user could proceed along paths leading from well understood goals to those goals with the greatest strength at each node or with the greatest 'path' strength. This will be considered in more detail at a later time.

The goal acquisition stage will involve the possible inclusion of new goals and/or the revision of relations and strengths between goals.

Returning to the problem of classifying the user's recollection of previous interactions associated with a given goal, we consider the process to be divided into two basic operations one of which is concerned with the current state of belief about the infer function and the other is a reserve process which might be invoked if the use of the previous method results in failure to achieve the goal and which calls into question possible previous 'understandings' of the system data type X_{user} and the infer function.

Essentially the former uses the $Z_i[g]$ history decomposition of the interaction data space and the latter is concerned with the sets E(g). We will briefly look at the first case here.

Suppose that a given goal g exists within the goal space and there is a history of active interactions, $Z_i[g]$, within the current understanding of the infer function. Clearly, if the history $Z_i[g]$ is consistent in the following sense then it is likely to be straightforward to construct a suitable action sequence to realise the goal g.

Definition. Let g be a goal and $Z_i[g]$ a current history.

We call $Z_i[g]$ consistent if all of the action sequences involved in the history are congruent. What this means is that the application of any action sequence in the furtherance of the goal g differs only in data entry actions.

We must emphasise that here, as in all of this article, we qualify all statements by referring to the possibility that any 'fact' believed to be true by the user is really a pair consisting of the fact and a measure of the user's certainty or belief in the truth of the fact. So histories are 'fuzzy' sets in reality. The formal generalisation of this material may well obscure some of our arguments at this stage.

So it is clear that if the history associated with a goal g is consistent then realisation should be reasonably easy. However it is possible that a goal can be achieved using essentially different action sequences, that is the state diagram allows for alternative paths. If this happens then we will find that the history $Z_i[g]$ can be decomposed into disjoint subhistories associated with the different methods.

Definition. Let g be a goal and $\alpha \in$ Actions then we define the subhistories
$Z_i[g,<\alpha>] = \{ m \in Z_i[g,\alpha'] \mid \alpha \cong \alpha' \}$

Each such subhistory represents a 'method' thought to result in the satisfaction of g. The selection of an interaction from one of the sets of subhistories associated with g will then act as a framework for the actual realisation of the given goal g in the current interaction under contemplation.

The problem of achieving a goal that is distinct from any known satisfied goal is rather more complex. If it is possible to decompose a goal into a sequence of 'subgoals' as described above then it may be possible to deduce, in some way, a procedure for realising the goal. Whatever the approach it is likely that the specific nature of the system will be crucial. Suppose that g and g' are goals such that $g < g'$. If $Z_i[g]$ and $Z_i[g']$ are consistent then we might expect that any action sequences and ' associated with $Z_i[g]$ and $Z_i[g']$ respectively be related by the equation

$<\alpha'> = <\alpha>.<\alpha">$

for some $\alpha" \in$ Actions. (Here we are considering a concatenation operation to be induced on the congruence classes, which is valid since is a congruence on the free semigroup generated by the actions.) If this is the case then some mechanism may exist which enables the user to realise that the achievement of goal g' occurs once g occurs together with some extra actions 'up to congruence'. Further research along this direction would seem to be worthwhile. Other relationships that exist between goals may imply further processes of goal reduction and deduction.

Naturally, we must consider the possibility that no knowledge about a desired goal exists in the interaction data space. Under these conditions the user will either resort to expert advice, which we will regard as an interaction with an extended data space or carry out some 'random' tests on the system. We look at this aspect next.

$7.Testing processes.

Each interaction between the user and the system is really an experiment, the results of which either produce new 'information' or reinforce old 'information'. Therefore we will not distinguish between testing and other interactions.

A testing element consists of an array
(α, x, g_t) where $\alpha \in$ Actions, $x \in X_{user}$,
$g_t \in G_t$ for some time t. We assume a fixed initial state as before.

There are two major decision processes that we will consider here;

(i) goal-directed testing,
(ii) system-exploration testing.

In the second type we will just choose testing sets at random and observe the results. For this we need a random test function which defines an input element and a test data element from the set X_{user}.

Thus there is a function which randomly allocates such a pair

$$\delta : R^{+} \dashrightarrow \text{Actions} \times X_{user}$$

At a given time t we will then invoke this function to provide a testing environment (α, x, random), where $\delta(t) = (\alpha, x)$ and random is a special element of the set G_t.

For the first type of testing, namely goal-directed testing we postulate several mechanisms.

Suppose that a goal $g_t \in G_t$ is given and the user is trying to find a way of realising this goal. If a related goal $g'_t \in G_t$ exists in the data space under the context field then it may be possible to devise an experiment which is in some sense related to the experiment or experiments linked to g'_t. This is most easily done mathematically in the situation where the two goals are related under the subgoal/supergoal relationship and this is what we will concentrate on next.

We suppose that we have goals g_t and g'_t in the goal space at time t, with g_t a goal about which we have some experimental information in the data space, and we are looking for a connection between them. There is a possibility that they are in disconnected parts of the space and so we cannot necessarily make use of the experimental knowledge relating to the realisation of g_t in the realization of g'_t. What happens then is dependent on a variety of factors including the availability of help in some form which may resolve the problem by indicating the existence of a related goal with a known solution. Thus the goal space will be updated together with the data space in some cases. However it is possible that there are sequences of goals

$$g_t \xrightarrow{h_t / s_t} g^1_t \xrightarrow{h^1_t / s^1_t} \cdots \longrightarrow g^m_t \xrightarrow{h^m_t / s^m_t} g'_t$$

where all of the intermediate goals are contained in some way in the data space of experiments. If the relations between the goals are in some sense reflected in the sort of relations that we have on the Actions set, for example subword or superword relations, then we can possibly make use of the information to generate an experiment that will realise g'_t. Searching the goal space may seem a complex activity but the use of neural processing techniques might shed further light on this problem and render it more realistic.

Given a realised goal g_t and a new goal g'_t we will look for a 'strong' path between them. The data space is searched for an experiment involving, say g^1, and experiments involving g^1 are analysed for 'clues' and conclusions about possible 'follow-up' experiments. To carry out the computation to achieve goal g^1_t we then apply action sequence α^1 with the desired initial data type element y. If g^1_t is not what is required we move on to g^2_t and consider that, knowing that the new goal is a subgoal of the previous one and is therefore related to it.

The problem here is the need to relate the relations h_t to be found on G_t with some sort of natural structure on the sets Actions, X_{user} and Displays. The possible nature of these sets is, naturally, of importance in the consideration of the relations that are defined on goals. We thus

need to consider the specification of the system and the applications that it is being put to in some detail, only then can some precise statements about these relations be made.

$8. Conclusions.

We have attempted to formulate a foundation for a mathematical analysis of user interaction and experimentation with a computer system. In such a complex situation there are bound to be aspects which we have not been able to take into consideration. If this article does no more than stimulate others into examining these issues formally then we will be satisfied. If, however, it is possible to devise a design methodology for system interfaces that has the capacity for predictive analysis of human-computer interaction then we will be even more pleased.

There is clearly a need to examine, in great detail, a realistic system and examples of user behaviour with the system in the light of the concepts presented here before it will be possible to determine if the model is at all realistic as a basis for future progress.

References. (Need reorganising)

[1] M.Holcombe. "X-machines as a basis for dynamic system specification." Software Engineering Journal. 1. 1988, 69-76.
[2] H.Levesque."A logic of implicit and explicit belief." Proc. AAAI 1984, 198-202.
[3] Halpern & Fagin. "Belief, awareness and limited reasoning." Artificial Intelligence. (34) 1988, 39-76.
[4] Duan & Holcombe. "Traceable X-machines as Models for describing User Interfaces." Proc 5th Int. Symp. Comp. & Inf. Sc. Cappadocia, Turkey, 1990, 599-608.
[5] Laycock. "Theory and practice of specification based testing." Ph.D. Thesis, Department of Computer Science, University of Sheffield, 1992.
(1991)
[6] Dix, "Formal methods in human computer interaction",Academic Press, London
[7] Gikas and Johnson (1993), "Formalising task models in design." Interacting with computers
[8] Harrison, M.D. and Thimbleby, H. (1989)," Formal methods in human-computer interaction." Cambridge University Press, Cambridge, England
[9] Thimbleby, H. (1990), "User interface design." Addison-Wesley, England.