SOLVING MECHANICS PROBLEMS USING META-LEVEL INFERENCE

Alan Bundy, Lawrence Byrd, George Luger,
Chris Mellish & Martha Palmer.
Department of Artificial Intelligence,
University of Edinburgh,
Edinburgh, Scotland.

In this paper we shall descibe a program (MECHO), written in Prolog [14], which solves a wide range of mechanics problems from statements in both predicate calculus and English. Mecho uses the technique of meta-level inference to control search in natural language understanding, common sense inference, model formation and algebraic manipulation. We argue that this is a powerful technique for controlling search while retaining the modularity of declarative knowledge representations.

Keywords Natural Language, Mathematical Reasoning, Search Control, Meta-level Inference, Predicate Calculus, Mechanics.

1. Introduction

The work described in this paper addresses the question of how it is possible to get a formal representation of a problem from an English statement, and how it is then possible to use this representation in order to solve the problem. Our purpose in studying natural language understanding in conjunction with problem solving is to bring together the constraints of what formal representation can actually be obtained with the question of what knowledge is required in order to solve a wide range of problems in a semantically rich domain. We believe that these issues cannot sensibly be tackled in isolation. In practical terms we have had the benefits of an increased awareness of common problems in both areas and a realisation that some of our techniques are applicable to both the control of inference and the control of parsing.

Early work on solving mathematical problems stated in natural language was done by Bobrow (STUDENT - (i]) and Chamiak (CARPS - [5]). However the rudimentary parsing and simple semantic structures used by Bobrow and Charniak are inadequate for any but the easiest problems. Our intention has been to build on advances in natural language processing (eg [18]) in order to study parsing and problem solving in a domain which requires sophisticated knowledge about the world. The domain we have been working in is that of mechanics problems, which deal with idealised objects such as smooth planes, light inextensible strings, frictionless pulleys etc. The idealised nature of this domain made it feasible to consider building an expert inferential system which would be able to cope with a wide range of problems. To date, our program has tackled problems in the areas of: pulley problems, statics problems, motion on smooth complex paths and motion under constant acceleration. Our intention is to continue expanding this in order to force generality into our solutions. In recent years a lot of similar work has been in progress on Physics-type domains such as ours. (eg [13], [7], [15], [11]). We have been concerned to adopt methods developed by these workers into Mecho, and to solve mechanics problems tackled by them.

2. Description of the Program

The block diagram (fig 1) gives a very general overview of the structure of the MECHO program. Each block represents a closely related collection of Prolog clauses (procedures), the arrows between blocks

This work was supported by SRC grant number B/RG 94493 and an SRC research studentship for Chris Mellish.

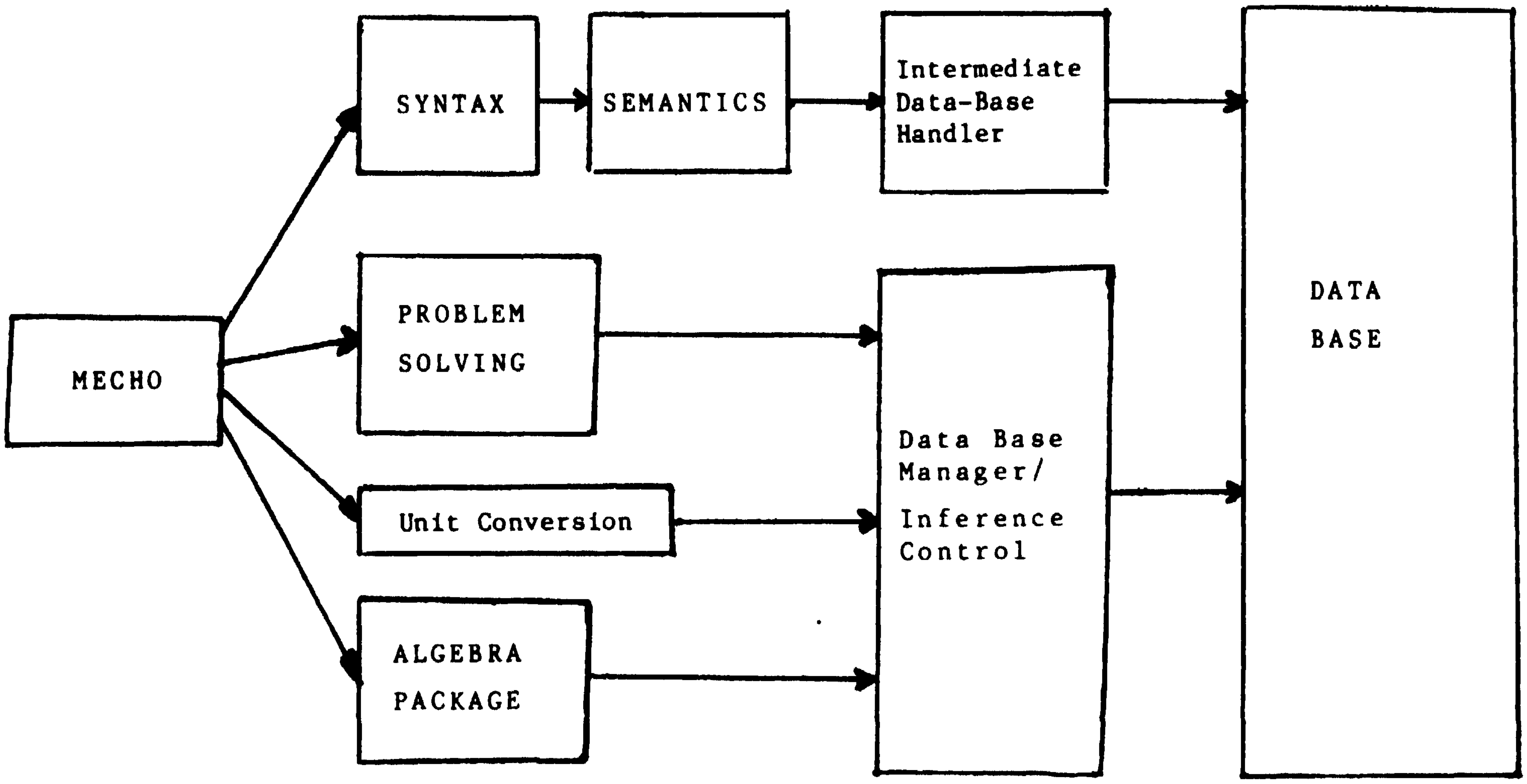


Fig 1 : Program Block Structure.

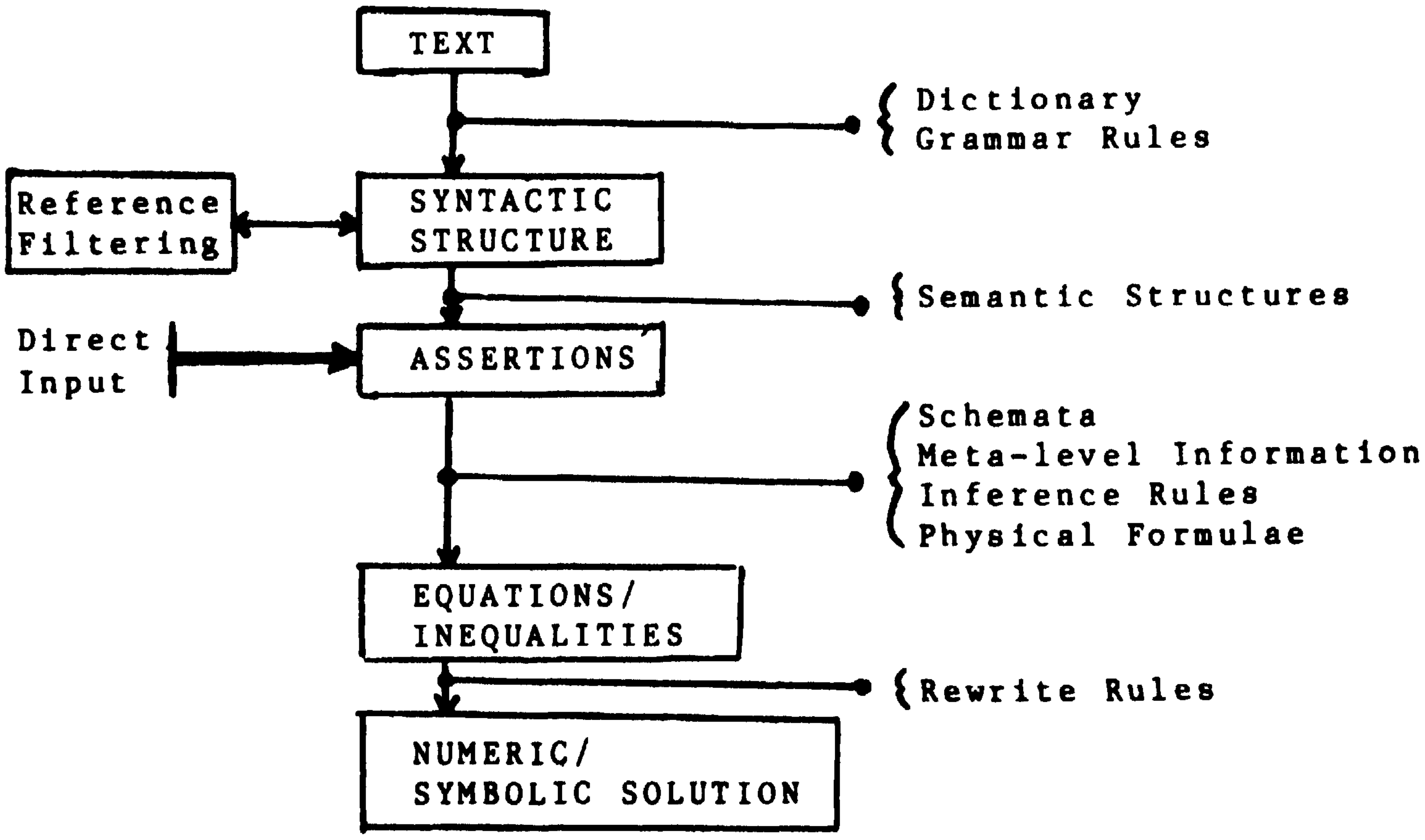


Fig 2 : Representation Structures

indicate invocation/communication links. (For practical reasons MECHO is split into three separate modules, but this is irrelevant to the overall structure). The accompanying diagram (fig 2) tries to capture the changes in representation and the various types of knowledge required during the execution of the program. The following discussion will elaborate on these.

Input to the program is in the form of English text. An example taken from the area of pulley problems would be:

> "Two particles of mass b and c are connected by a light string passing over a smooth pulley. Find the acceleration of the particle of mass b." (1)
>
> (Taken from [10])

The purpose of the natural language module is to produce a set of predicate calculus assertions which will enable the problem solver to solve the problem. This objective of producing a symbolic representation of the 'meaning' of the problem statement has been used by us as a vehicle for exploring syntax-semantics interaction. The syntactic parser calls semantic routines as soon as possible in order to interpret fragments of text and quickly reject inappropriate syntactic choices. The work of the syntax routines divides into clause syntax and phrase syntax.

The purpose of syntactic analysis at the clause level is to establish clause boundaries and, within each clause, to prepare the ground for the semantic analysis of the main verb. Clause analysis thus involves identifying the start of new phrases, assigning syntactic roles to the phrases and performing phrase analysis to interpret the internal structure of the phrases. The internal phrase analysis typically returns simply a referent (and some typing information) to the higher levels. This means that preliminary reference evaluation is carried out locally, with the information conveyed by a phrase being captured in assertions produced as 'side effects' by semantic routines called during the analysis. These semantic routines are responsible for interpreting what it means for a given object to have a certain property, and indeed for checking whether or not it can have the property. Domain specific information concerning typing, idealisation and object-property possibilities is used to answer these questions. Failure of the semantics indicates that the parse is invalid. The following (simple) example shows the kind of meta-level structures used in this process.

```
meaning(light,Object,mass(Object,zero)).

type_constraint(light,physobj).
```

The meaning predicate states that the meaning of applying the property light to an object, is that the object has a mass of zero. type__constraint asserts that being a physical object is a necessary condition to having the property light.

It is interesting to note that the declarative assertions which give the meaning of a phrase can be specified independently of how they will be used. Meta-level information concerning the state of the analysis is used to determine whether they are used to add new information or to test neccessary constraints on previous information. (This is basically the 'given'/'new' distinction discussed in [8]).

One of the aspects of natural language understanding that has interested us especially is the way in which criteria of semantic well-formedness can be used to resolve cases of ambiguity in reference evaluation. Our program incorporates a full deductive mechanism, as opposed to semantic markers, to capture the global semantic constraints that arise during the interpretation. Reference evaluation proceeds continuously during the combined syntactic and semantic analysis with semantic information being used to filter sets of possible candidates for referents. The method used to achieve this in a general way is basically that of Waltz filtering [16]. As can be seen, the referent returned by the phrase syntax is likely to be incompletely specified and for this reason all interaction between the semantics and the data-base is handled by an intermediate data-base handler which implements

The notation used here, and in following examples, follows the Prolog convention that names starting with an upper case letter are logical variables which are purely local to the structure (Prolog clause). Atoms, which are in lower case, and compound terms all stand for themselves. Rules are of the form 'P <— Q & R' meaning 'if Q and R then P'. Most examples have undergone slight cosmetic alteration.

the inference system and reference filtering system over these referents.

The syntactic structure built by the clause syntax specifies a main verb, and positions such as 'logical subject' and "logical object' are filled by referents. (We do not construct a complete parse tree as such). From this structure a set of assertions is generated by invoking semantic routines. The semantic analysis of the verb maps the main verb onto a base verb and establishes a mapping between the syntactic roles of the clause and the deep roles associated with the base verb. As a result the referents are fitted into conceptual slots in a way similar to conventional 'caseframe' analysis. The base verb then specifies the assertions (in terms of the referents) which follow from this mapping. Base verbs differ from case frames In that while they attempt to generalise collections of related verbs, they are not defined in terms of universal primitive roles or slots.

Given a satisfactory parse of a sentence, which produces a set of consistent assertions and disambiguates the referents, we are then able produce a set of assertions (by instantiating out the referents) about the objects in the problem. These are supplied to the problem solving module. As an example, the assertions produced for the above problem statement (1) would be:

```
    isa(period,period1)
    isa(particle,p1)
    isa(particle,p2)
    isa(string,s1)
    isa(pulley,pull)
    end(s1,end1,right)
    end(s1,end2,left)
    midpt(s1,midpt1)
    fixed_contact(end1,p1,period1)
    fixed_contact(end2,p2,period1)
    fixed_contact(midpt1,pull,period1)
    mass(p1,mass1,period1)
    mass(p1,mass2,period1)
    mass(s1,zero,period1)
    coeff(pull,zero)
    accel(p1,a1,270,period1)
    accel(p2,a2,90,period1)
    measure(mass1,b)
    measure(mass2,c)
    sought(a1)
    given(mass1)
    given(mass2)                                (2)
```

In addition the following schema is cued:

```
 cue pullsys_stan(sys1,pull,s1,p1,p2,period1)
```

The cueing of schemata is necessary to provide extra information, defaults etc. which are not given explicitly but are 'house rules' in this domain. (Eg That the pulley in a pulley system has negligable weight). We cue schemata, fairly simplistically, by recognising key words and certain object configurations. For example the following structure asserts that a pulley-system schema can be cued if objects can be found which satisfy the ideal-type contraints and have certain relationships between each other.

```
 sysinfo( pullsys,
      [Pull,Str,P1,P2],
      [pulley,string,solid,solid],
      [ supports(Pull,Str),
        attached(Str,P1),
        attached(Str,P2)
      ]).
```

The effect of this cue would be to invoke a schema such as:

```
 schema( pullsys,
      [Pull,Str,P1,P2], Time,
      [ constaccel(P1,Time),
        constaccel(P2,Time),
        cue stringsys(Str,[Lpart,Rpart]),
        (tension(Lpart,T1,Time)
             <-- coeff(Pull,zero) &
                 tension(Rpart,T,Time) )
      ],
      [ coeff(Pull,zero),
        mass(Pull,zero,Time)  ]).
```

This schema asserts that in a standard pulley problem the objects undergo constant acceleration, the tension in both parts of the string are equal if there is no friction (only one rule shown), and that the friction and mass of the pulley default to zero if not otherwise given. (This example has been somewhat simplified).

The predicate calculus notation can be used to input problems directly to the problem solver - and in fact research on the problem solver has resulted in it being able to tackle a wider range of problems than the natural language module can currently handle. The representational principles behind these assertions view the objects of Newtonian Mechanics in terms of simple zero and one dimensional objects (points and lines) which are typed and have properties and relations defined over them. For example particles, pulleys, spatial points, moments of time are all types of POINT while rods, strings, paths (trajectories), and periods of time are types of LINE. Physical quantities, such as length, velocity, force etc., form the other main branch of our type hierarchy, (see [A]).

The work of the Problem Solver divides into two types of task. There is the overall strategic task of deciding what to do, how to solve the problem by producing equations which solve for unknown quantities (including intermediate unknowns introduced during the solution). On the other hand there is the tactical task of combining the input assertions with general facts and inference rules in order to prove required goals. We shall discuss each of these in turn.

Our overall strategy is a general goal directed algorithm for equation extraction developed from a study by David Marples of student engineers (12). For instance, suppose al, the acceleration of particle pl, is the (only) sought unknown. (Here we continue the example started above (1) (2)). Resolution of forces about pl will be chosen to solve for al and this produces the equation:

$$\mathtt{-massl.g + tensionl = massl.al} \qquad (3)$$

All possible force contributions on pl are examined and since pl is attached to the end of the string this results in the string being considered. tensionl was formerly unknown but the function properties of the predicate 'tension' enable it to be created (see later) to allow the equation to be formed. We have to introduce tensionl as an unknown because it is not possible to solve for al without doing so. The next step is to solve for tensionl which is a force and involves the string sl. Again resolution of forces is selected - pl, pull, p2 being objects on the string which are possible candidates for resolving about. pl has been previously used and only p2 can be used without introducing unknowns. The result is the equation:

$$\mathtt{mass2.g - tensionl = mass2.al} \qquad (A)$$

These two equations can be solved to produce a solution for al.

The input assertions provide meta-level information about whether certain quantities are sought or given. The Marples Algorithm works by traversing the list of sought unknowns in a fixed order: the (quantity) type of each unknown being used to provide a shortlist of formulae that could solve for it, and the definition of the quantity (ie the assertion which introduced it) being used to find the physical objects, times and angles involved. Notice that there is a distinction made between 'formulae', which are composed of variables over quantities (eg 'F - M * A'), and 'equations' which are instantiations of formulae (eg (3) and (4) above). In the Marples algorithm we are reasoning about the properties of formulae in order to successfully produce appropriate equations.

Before the application of a formula to produce an equation, there is a stage of qualitative analysis where general facts about the problem are used to decide applicability. Our interest here is in exploring the selective use of meta-level reasoning to guide the equation extraction process. As well as deciding applicability we have to prepare a situation within which to apply a formula. This may involve, for example, collecting together all the objects connected to a particle if we wish to resolve forces about it. General independence criteria (eg 'You can't resolve forces about the same object in linearly dependent directions') are also used to eliminate redundant equations.

The following are (simplified) examples of the meta-level structures used during the above example:

```
kind(al,accel,
        relaccel(pl,earth,al,270,periodl)).

relates(accel,
        [resolve,constaccel-N,relaccel]).

prepare(resolve,relaccel(P,earth,A,Dir,Time),
                situation(P,Set,Dir,Time))
    <-- isa(particle,P) &
        findall(X, sameplace(X,P,Time), Set).

isform(resolve,situation(Obj,Set,Dir,Time),
               F = M * A )
    <-- mass(Obj, M ,Time) &
        accel(Obj,A,Dir,Time) &
        sumforces(Obj,Set,Dir,Time, F ).
```

The kind predicate asserts that al is a quantity of type accel defined in the given relaccel assertion. relates states that all the formulae whose names are given in the list, contain variables of type accel and therefore can be used to solve for acceleration, prepare gives the criteria for constructing the situation within which to resolve forces, and the isform predicate defines the equation by defining the meaning of its component variables.

The equation extraction algorithm is two pass in that it first tries to produce a solution which does not introduce new unknowns before allowing the introduction of extra (intermediate) unknowns which are added to the unknowns list and have to be eventually solved for. Notice that the quantities manipulated are purely symbolic; they can be introduced by the creation mechanism (see later) without their values being known at this stage. E.g. When the first equation (3) was formed in the above example, the quantity tensionl was introduced without the program knowing, or trying to find, its actual value. As can be seen, it is the Marples algorithm which will eventually produce an equation which solves for tensionl.

The data-base stores all the facts supplied by the English statement, but to bridge the gap between the explicit information derived from the problem statement and that needed to solve the problem the program requires a general knowledge of mechanics which is formalised in a set of inference rules. An example of such (object-level) inference rules would be:

```
relaccel(Pl,P2,zero,Dir,Period)
     <-- constrelvel(Pl,P2,Period).

constrelvel(Pl,P2,Period)
     <-- fixed_contact(Pl,P2,Period).
```

The first rule says that the relative acceleration between two points of reference is zero if there is a constant relative velocity between them (over a certain period), and the second rule says that two points of reference have a constant relative velocity if they are in contact (again, over a certain period). The inference rules are a set of horn clauses which have been hand ordered and contain certain typing information to guide selection. The search strategy is depth first, with pruning of semantically meaningless goals, and while this could be improved upon, current performance has' not yet necessitated such a step.

An important part of our work has been the investigation of search control mechanisms which will enable effective use of this wealth of implicit knowledge. All requests to retrieve assertions from the data-base, either directly or via inference, are handled by the Inference control module. This module uses information from the request along with meta-information and inference rules in an attempt to satisfy the goal. The first step involves normalisation of the goal to remove syntactic sugar or to express it in terms of a smaller set of underlying predicates. This is performed with a one pass rewrite rule set. The resulting goal is then classified according to the instantiation state of its component arguments and the possibility of using function properties and certain other mathematical properties of the predicate (such as reflexivity, symmetry and transitivity). This information is used to select appropriate proving strategies. (A basic strategy of 'unit preference' will always first check the data-base directly).

Our two most important strategies are the use of function properties to prune search and the use of equivalence class type mechanisms to direct it. The representation treats what would normally be considered functions as predicates with extra control Information. Being a function means that certain arguments are uniquely determined by certain other arguments. For example, in the predicate tension(String,T,Time)' The actual tension T is determined once the String and the Time have been given.

These function properties can be used to prevent useless inference if another (different) value of a function argument is already known (uniqueness property); to create new entities to satisfy a goal if all attempts at inference have failed (existence property); and to automatically eliminate backtracking by disregarding choices made during inference (uniqueness again). Examples of the meta-level structures used by the program in performing the above are:

```
rewrite( accel(P,A,Dir,Time),
         relaccel(P,earth,A,Dir,Time),
         strategy(dbinf)  ).

meta( relaccel, 5,
      [P1,P2,A,Dir,Time],
      [pt_of_ref,pt_of_ref,accel,angle,time],
      function( [P1,P2,Time] => [A,Dir] )  ).
```

The rewrite rule tells us that any accel predicate can be rewritten to a relaccel predicate with the earth as the other point of reference, and that the standard inference strategy is then applicable. The meta predicate specifies the argument types and the function properties of the predicate relaccel.

The second major strategy, which provides an alternative to using the inference rules, is a general similarity class mechanism based on equivalence class ideas. Predicates which are (pseudo-) equivalence relations and would normally produce self-resolving inference rules are defined in terms of this mechanism. A tree is used to represent similarity class membership and the goal (such as 'being in the same place') is proved by establishing equivalence of roots. This can be seen as an alternative (and less explosive) axiomatisation of these predicates. Our extension over traditional uses of this method has been to allow labelled arcs and calculation during the tree traversal. Thus predicates like 'vector separation' and 'relative velocity' which have pseudo-equivalence properties can also use this strategy. Here is an example of a structure used in these cases:

```
rewrite(  sameplace(P,Q,Time),
          [  sameclass(P,Q,touch(Time)),
             merge(P,Q,touch(Time))       ],
          strategy(simclass)   ).
```

This states that the predicate sameplace should use the general sameclass mechanism on the particular tree touch(Time). Also specified is an updating mechanism for adding new sameplace assertions; In this case it would involve merging two separate trees.

These general strategies can be applied to a wide range of predicates and often capture important facts about the domain (eg the fact that an object cannot be in two places at once is a fact about the function properties of 'at(Object,Place,Time)'). The explicit control of new object creation coupled with the goal directed backward reasoning method of the Marples algorithm results in a create/consider-by-need type of behaviour. Restrictions, such as 'don't create' or 'don't infer', can be added to the request for a goal to be proved and this enables the Marples algorithm to be selective over its use of the Inference Control in accordance with its needs at the time.

For some mechanics problems a process of prediction is required to answer questions like "Will the particle reach the top of the slope if it starts with velocity V ?". Each question about the motion of a particle on a complex slope unpacks into a series of questions about the behaviour on simple parts of the slope. Some of these can be answered immediately on the basis of the qualitative shape of the slope, but others involve the solution of inequalities containing unknown quantities. These unknowns are declared as sought and the equation extraction algorithm is called to solve for them. The inequalities can then be solved to answer the question. Our present prediction system is special purpose and built around problems similar to those in tackled by De Kleer, ie motion problems.

Since the equations produced by the equation extraction algorithm are in terms of symbolic quantities, there is a stage of Unit Conversion where the actual values are substituted and a final unit system is selected - conversion factors being added where appropriate. (Some problems involve a combination of all sorts of different units - feet, yards, miles......). The two equations produced above ((3) & (4)) are very simple in that no particular units are involved. The only step will be the substitution of b and c for massl and mass2 respectively giving:

-b.g + tensionl * b.al (5)

eg - tensionl ■ c.al (6)

The set of simultaneous equations and/or inequalities produced by the problem solving module is passed to the algebra module (PRESS) which will solve them to produce a final answer to the problem. Let us look at how PRESS produces a solution for al given (5) and (6). The two equations are solved by isolating tensionl in the second equation (which was intended to solve for tensionl), and then using the result as a substitution into the first equation. This final result can then be simplified with al being isolated on the left hand side to give the final answer:

al » g.(c-b) / (c+b) (7)

The extension of equation solving techniques to inequalities (there are interesting connections) has enabled us to solve the inequalities produced by the prediction problems, but in addition we have found that the information required to justify the use of certain rewrite rules is often of the form 'only if X > 0' etc. Solving and proving inequalities is therefore of direct use within the system.

However, PRESS was not developed purely as a service program for MECHO. It was intended as a vehicle to explore ideas about controlling search in mathematical reasoning using meta-level descriptions and strategies [3].

Rather than using exhaustive application over a large set of rewrite rules, it uses the meta-level stategies of isolation, collection and attraction to carefully control application of several different sets of rewrite rules. This selectivity has many advantages: principled methods for guiding search cut down useless work, identical rules may be used in different ways (eg left to right or right to left) in different circumstances without causing problems, and theoretical requirements such as proof of termination of the rewrite rules are made much easier.

When PRESS is used as an equation and inequality solver (ie as a module of MECHO), it classifies the equations (inequalities) to be solved so as to generate guidance information. An exciting area of research that we would like to expand on is that of designing inclusion and ordering criteria to classify algebraic identities which are produced by a theorem prover. This would enable the system to automatically learn new rules. The use of meta-level reasoning to place new rules into a framework where they will be selectively and correctly applied overcomes many of the obvious 'explosion' and 'looping' problems that would occur with haphazard additions to a large single rewrite rule set.

3. Discussion

Throughout the above description of the Mecho program we have constantly emphasised the importance of 'meta-information' in controlling search. This has been applied in the rejection of semantlcally meaningless parses, the control of inference, the extraction of equations and the guiding of algebraic manipulation.

The theme that has emerged from our work is the benefit to be gained from axiomatising the meta-level of the domain under investigation and performing inference at this level, producing object level proofs as a side effect. This is the methodology investigated by Pat Hayes in the GOLUX project [9], except that we have developed our meta-level representation for a particular domain rather than adopting general purpose representations based on resolution theorem proving systems.

In [2] we showed how GPS could be viewed in this way. At the object-level the search space can be viewed as an operator/state OR tree in which the states are nodes and the operators are arcs between them. At the meta-level the search space can viewed as a method/goal AND/OR tree In which the goals are nodes and the methods are arcs between them. A simple depth first search at the meta-level then induces a highly complex, middle-out search at the object-level.

In order to make clear the distinction we are drawing between meta-level and object-level representations in Mecho, we shall list below examples of the descriptions used at each level. When defining a notation it is usual to define the constants, variables, function symbols and predicate symbols of the language; and then to show how terms and formulae can be formed by composing them together with the logical connectives. We shall follow this type of outline in an informal fashion. (To avoid confusion with earlier terminology we shall use the words 'assertion' and 'rule' to replace 'formula').

At the object-level we have the following kinds of primitive:

constants	pl, endl, mass2, al, right, 90, 270, lbs, feet etc.
variables	Pl, Str, Period, Accel, F, M, etc.
function symbols	+, *, cos etc.
predicate symbols	accel, relaccel, mass, fixed_contact etc.

These are formed into terms such as 'M * A' and assertions such as 'F - M * A', 'accel(pl,al, 270,periodl)' etc. Finally, logical connectives are used to form these assertions into inference rules like:

```
relaccel(P1,P2,zero,Dir,Period)
     <— constrelveKPl,P2,Period) .
```

The only function symbols at the object-level are for straight forward arithmetic and trigonometric functions. This is because we have recorded function properties by making meta-level assertions about object-level predicates.

At the meta-level all these object-level descriptions are meta-constants along with additional meta-constants for schema names, formula names, object types, strategy types etc. As examples of meta-level primitives we have:

constants (Any object level description). physobj, pullsys, resolve, particle, line, length, dbinf etc.

variables Type, Eqn, Goal, Strategy, Exprl etc. (see later)

function symbols Constructors for lists, sets, bags etc.

predicate symbols meaning, sysInfo, schema, kind, relates, isform, rewrite, meta etc.

Again these can be formed into meta-terms and meta-assertions and we gave several examples during the program description. What we shall now examine are the meta-rules which are formed from these assertions. (These use the same logical connectives as the object-level rules). We shall take simplified examples from each of the four main areas of our work.

The first example is a rule used by the Natural Language module, which specifies the conditions for a property to be correctly applied to a particular entity.

```
attribute(Property,Entity,State)
  <— type constraint(Property,Type) &
       isa(Type,Entity) &
       meaning(Property,Entity,Assertion) &
       consistent(Assertion,State).
```

This rule states that a particular Property can be attributed to an Entity in a given State (of the parse), if the Entity satisfies the type_constraint of of the Property, and if the meaning of the attribution is consistent with all the other assertions in the current State. (If, for example, the Assertion was 'mass(si, zero)' then this would involve checking that no other mass was known for sl. It is here that we see one of the key connections with our work on Problem Solving, since this is precisely a matter of 'function properties'!).

The following is an example taken from the Marples Algorithm, and it defines the requirements for an equation to solve for a particular quantity.

```
solves_for(Q,Eqn)
  <-- kind(Q,Type,Defn) &
       relates(Type,Flist) &
       select(Formula,F_list) &
       prepare(Formula,Defn,Situation) &
       isform( Formula,Situation,Eqn) .
```

This rule states that Eqn solves for Q if Q has type Type and Formula is a formula that relates Type quantities to other quantities, and if Situation is the situation within which to apply the Formula given the Defn of Q, and if Eqn is the instantiation of the formula in the given Situation. It can be seen that this rule is a direct axiomatisation of our earlier description of how the Marples algorithm extracts equations. (The select goal would specify the qualitative guidance and apply the independence criteria (given extra arguments)).

In a similar way we give the following example of rules which describe how the Inference Control uses function properties.

```
is_satisfled(Goal)
  <— rewrite(Goal,Newgoal,Strategy) &
      decompose(Newgoal,Pred,Args) &
      meta(Pred,N,Args,Types,Func_info) &
      method(Strategy,Func_info,Newgoal).

method( strategy(dbinf),
        function(Fargs ■> Vals), Newgoal)
  <— allnbound(Fargs) &
      use_function_properties(Newgoal).
```

The first rule states that Goal is satisfied if it rewrites to Newgoal whose predicate symbol has certain Func_info, and if a method is used based on the Strategy and this Func_info. (Certain arguments to meta have been ignored). The second rule states that the normal inference method will prove Newgoal given its function properties if all the function arguments, Fargs, are bound, and if object-level inferencing is performed using function property pruning.

As a final example we take a rule concerned with algebraic equation solving. This rule is interesting in that while PRESS does not currently use it, it could be derived from rules PRESS does have. Automating this procedure would be an interesting area for study.

```
solve(U,Exprl,Ans)
  <— occ(U,Exprl,2) &
      collect(U,Exprl,Expr2) &
      isolate(U,Expr2,Ans).
```

This rule states that Ans is an equation which solves for U given Exprl if Exprl contains two occurances of U, if Expr2 is an equation derived from Exprl in which these two occurances have been collected together, and if

Ans is an equation derived from Expr2 in which U has been isolated on the left hand side.

All the above rules can be seen as classifying object-level descriptions and using this information in deciding what to do. However the effects are very different in the different areas. In the natural language processing meta-level rules monitor object-level assertions, rejecting semantically unacceptable consequences of a parse. In equation extraction the effect is to select equations using a means/ends analysis technique. In Inference Control the result is use of the most effective axiomatisation for the goal in hand, and in Algebraic manipulation multiple rewrite rules are selectively brought to bear on expressions. Thus relatively simple meta-level search strategies can induce a wide variety of complex object-level behaviours.

These meta-inference techniques were strongly suggested by our use of the programming language Prolog. The fact that Prolog procedures are also predicate calculus clauses and the fact that predicate calculus has a clear semantics, encourages the user to attach meanings to his procedures and these meanings are usually meta-theoretic. However, Prolog as a programming language only offers a single level of 'syntactic' structures (atoms, compound terms etc.), and a lack of care can lead to a blurring of theoretical distinctions. During the development of Mecho, a lack of emphasis (realisation?) of these distinctions resulted in a mixing of object and meta levels, (for example the use of Prolog variables to represent variables at both levels, the mixing of object and meta level assertions in rules such as i8form). We plan to remove these aberrations.

Weyrauch's work on the FOL system (See [17]), is of importance in relation to this need for an adaquate theoretical formalism. The distinction between the object-level and the meta-level is fundamental within his system, and his use of 'reflection principles' is designed to capture the relation between these levels. We feel that his work is of direct value to workers in the field of expert systems, such as ourselves.

The principle of utilising 'knowledge about knowledge' is becoming increasingly important in practical AI programs. Davis and Buchanan [6] classified four different kinds of meta-level knowledge used by their TEIRESIAS system. They represent knowledge about objects and the data-structures used to describe them in schemata and describe the argument type characteristics of their functions in templates. Their program can classify and build models of the inference rules it uses and meta-rules are used to guide the choice of inference rules to be used and the order of using them. In MECHO, the Natural Language and Inference Control modules both use information like that stored in TEIRESIAS templates. MECHO meta-level inference rules are similar in spirit to TEIRESIAS meta-rules except that the MECHO rules are more general purpose and they generate a variety of different search strategies in different contexts.

4. Conclusion

In this paper we have discussed Mecho, a program for solving mechanics problems. We have shown how the technique of using and controlling knowledge about the domain by inference at the meta-level, can be applied to a range of different areas. Many workers in the field (eq [9], [6], [17]), have argued that controlling search by using meta-level inference is superior to built-in, smart search strategies because the search information is more modular and transparant. The argument is for systems to make explicit the full knowledge involved in their behaviour, which in turn aids the modification of their data and strategies, thus improving their robustness and generality. This leads the way to systems which could automatically modify their strategies and explain their control decisions.

We conclude that meta-level inference can be used to build sophisticated and flexible strategies, which provide powerful techniques for controlling the use of knowledge, while retaining the clarity and modularity of a declarative knowledge representation.

5. References

[1] Bobrow, D.
Natural Language input for a computer problem solving system.
PhD thesis, MIT, , 1964.

[2] Bundy, A. "Computational models for problem solving," Learning and Problem Solving (part 3), The Open Univ. Press, 1978. units 26-27 of the Open University Cognitive Psychology Course D303

[3] Bundy, A. & Welham, R,
Using meta-level descriptions for selective application of multiple rewrite rules in algebraic manipulation.
Forthcoming working paper,
Dept. of Artificial Intelligence, Edinburgh., 1979.

[4] Bundy, A., Byrd, L., Luger, G., Mellish, C, Milne, R. and Palmer, M.
Mecho: A program to solve Mechanics problems.
Working Paper No. 50,
Dept. of Artificial Intelligence, Edinburgh., 1979.

[5] Charniak, E.
Computer solution of calculus word problems, pages 303-316.
IJCAI, 1969.

[6] Davis, R. & Buchanan, B.G.
Meta-level knowledge: overview and applications, pages 920-927.
IJCAI, 1977.

[7] De Kleer, J.
Qualitative and quantitative knowledge in classical mechanics.
Technical Report AI-TR-352, MIT AI Lab, 1975.

[8] Haviland, S.E. & Clark, H.H.
What's new? Acquiring new information as a process in comprehension.
Journal of Verbal Learning and Verbal Behaviour 13:512-521, 1974.

[9] Hayes, P.
Computation and deduction.
Czech. Academy of Sciences, 1973.

[10] Humphrey, D.
Intermediate Mechanics, Dynamics.
Longman, Green & Co., London, 1957.

[11] Larkin, J.
Problem solving in Physics.
Technical Report, Group in Science and Mathematics Education, Berkeley, California, 1977.

[12) Marples, D.
Argument and technique in the solution of problems in Mechanics and Electricity.
Technical Report CUED/C-Educ/TRI, Dept. of Engineering, Cambridge, England, 1974.

[13] Novak, G.
Computer understanding of Physics problems stated in Natural Language.
Technical Report TR NL30, Dept. Computer Science, Univ. of Texas, Austin., 1976.

[14] Pereira, L.M., Pereira, F.C.N, and Warren, D.H.D.
User's guide to DECsystem-10 PROLOG.
Dept. of Artificial Intelligence, Edinburgh, 1978.

[15] Stallman, R.M. & Sussman, G.J.
Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis.
Technical Report No. 380, MIT AI Lab, 1976.

[16] Waltz, D.
Generating semantic descriptions from drawings of scenes with shadows.
Technical Report MAC AI-TR-271, MIT AI Lab, 1972.

[17] Weyhrauch, R.W.
Prolegomena to a theory of mechanized formal reasoning.
RWW Informal Note 8.,
Stanford University, 1979.

[18] Winograd, T.
Understanding Natural Language.
Edinburgh University Press, 1972.

What is an Image?

Harold Cohen

University of California at San Diego

B-027, UCSD, La Jolla, CA 92093

Image-making, and more particularly art-making, are considered as rule-based activities in which certain fundamental rule-sets are bound to low-level cognitive processes. AARON, a computer-program, models some aspects of image-making behavior through the action of these rules, and generates, in consequence, an extremely large set of highly evocative "freehand" drawings. The program is described, and examples of its output given. The theoretical basis for the formulation of the program is discussed in terms of cultural considerations, particularly with respect to our relationship to the images of remote cultures. An art-museum environment implementation involving a special-purpose drawing device is discussed. Some speculation is offered concerning the function of randomising in creative behavior, and an account given of the use of randomness in the program. The conclusions offered bear upon the nature of meaning as a function of an image-mediated transaction rather than as a function of intentionality. They propose also that the structure of all drawn images, derives from the nature of visual cognition.

1. INTRODUCTION

AARON is a computer program designed to model some aspects of hunan art-making behavior, and to produce as a result "freehand" drawings of a highly evocative kind (figs 1,2). This paper describes the program, and offers in its conclusions a number of propositions concerning the nature of evocation and the nature of the transaction — the making and reading of images — in which evocation occurs. Perhaps unexpectedly — for the program has no access to visual data — some of these conclusions bear upon the nature of visual representation. This may suggest a view of image-making as a broadly referential activity in which various differentiate modes, including what we call visual representation (note 1), share a significant body of cannon characteristics.

In some respects the methodology used in this work relates to the modelling of "expert systems" (note 2), and it does in fact rely heavily upon my own "expert" knowledge of image-making. But in its motivations it comes closer to research in the computer simulation of cognition. This is one area, I believe, in which the investigator has no choice but to model the human prototype. Art is valuable to human beings by virtue of being made by other human beings, and the question of finding more efficient modes than those which characterise hunan performance simply does not arise.

My expertise in the area of ijnage-making rests upon many years of professional activity as an artist — a painter, to be precise (note 3) — and it will be clear that my activities as an artist have continued through my last ten years of work in computer-modelling. The motivation for this work has been the desire to understand more about the nature of art-making processes than the making of art itself allows, for under normal circumstances the artist provides a near-perfect example of an obviously-present, but virtually inaccessible body of knowledge. The work has been informal, and Qua psychology lacks methodological rigor. It is to be hoped, however, that the body of highly specialised knowledge brought to bear on an elusive problem will be some compensation.

AARON is a knowledge-based program, in which knowledge of image-making is represented in rule form. As I have indicated I have been my own source of specialised knowledge, and I have served also as my own knowledge-engineer. Before embarking on a detailed account of the program's workings, I will describe in general terms what sort of program it is, and what it purports to do.

First, what it is not. It is not an "artists' tool". I mean that it is not interactive, it is not designed to implement key decisions made by

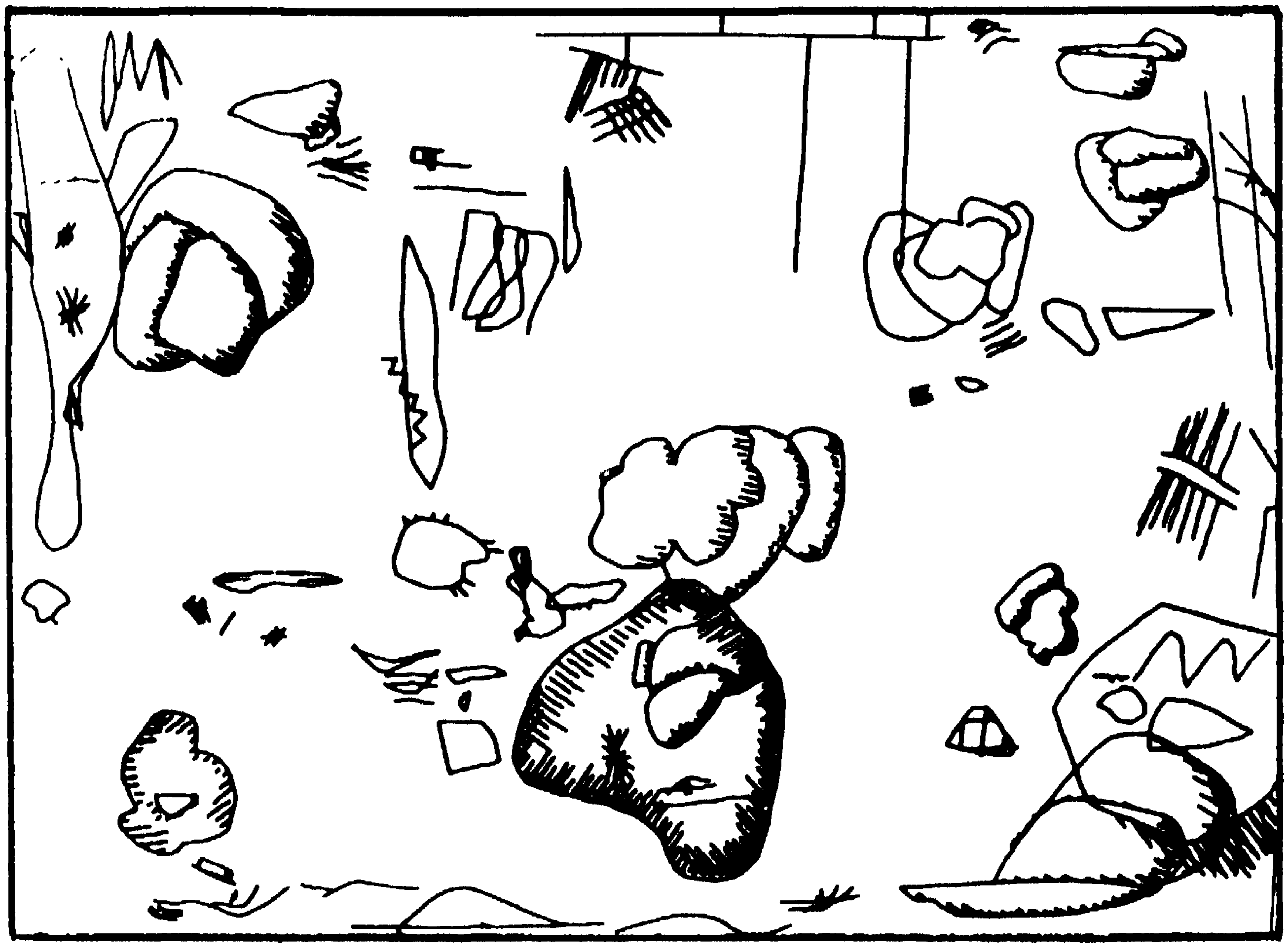


figure 1.

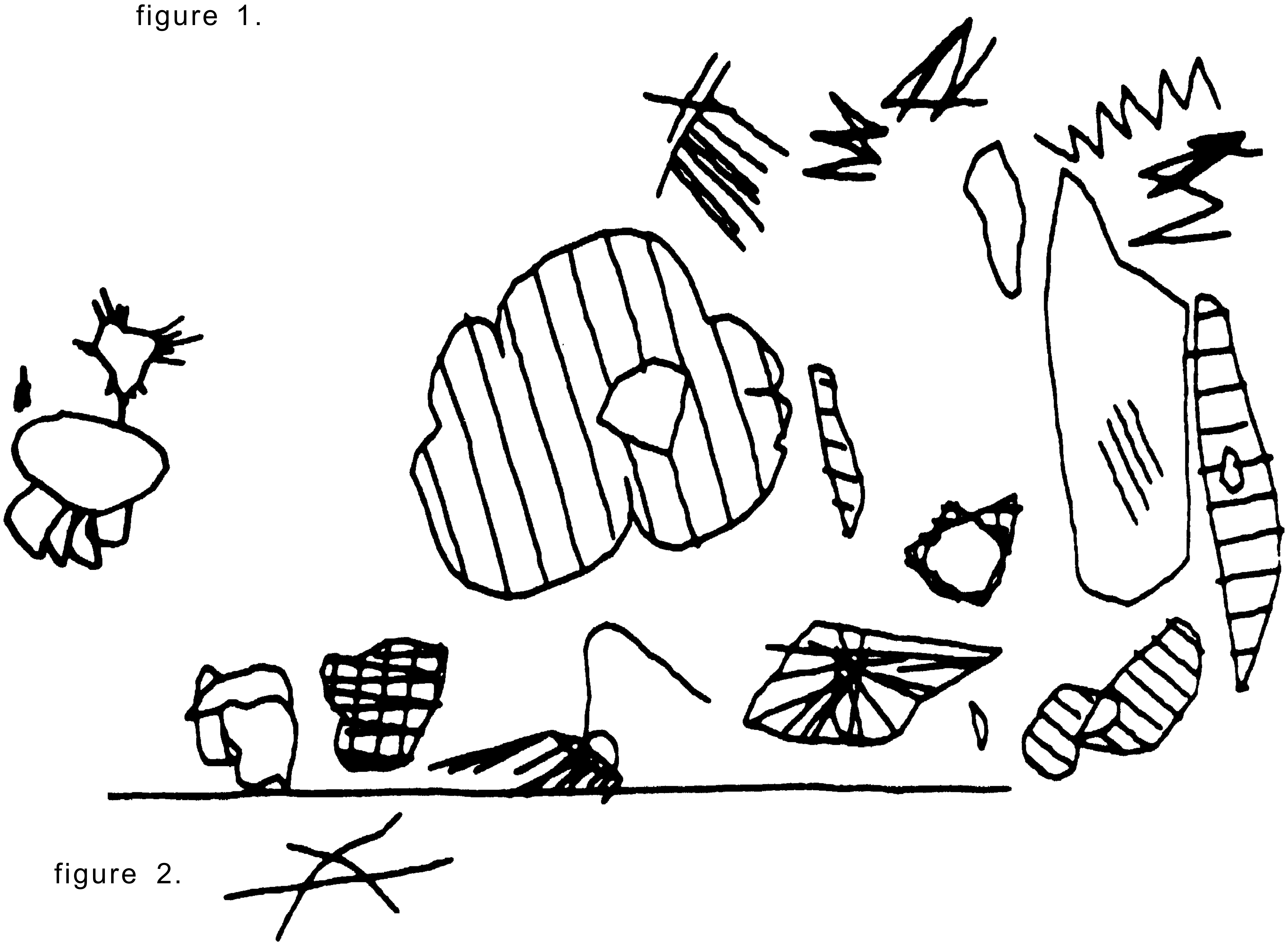


figure 2.

the user, and it does not do transformations upon input data. In short, it is not an instrument, in the sense that most computer applications in the arts, and in music particularly, have identified the machine in essentially instrument-like terms,

AARON is not a transformation device. There is no input, no data, upon which transformations could be done: in fact it has no data at all which it does not generate for itself in making its drawings. There is no lexicon of shapes, or parts of shapes, to be put together, assembly line fashion, into a complete drawing.

It is a complete and functionally independent entity, capable of generating autonomously an endless succession of different drawings (note 4). The program starts each drawing with a clean sheet of paper — no data — and generates everything it needs as it goes along, building up as it proceeds an internal representation of what it is doing, which is then used in determining subsequent developments. It is event driven, but in the special sense that the program itself generates the events which drive it.

It is not a learning program, has no archival memory, is quite simple and not particularly clever. It is able to knock off a pretty good drawing — thousands, in fact — but has no critical judgement that would enable it to declare that one of its drawings was "better" than another. That has never been part of the aim. Whether or not it might be possible to demonstrate that the artist moves towards higher goals, and however he might do so through his work, art-making in general lacks clear internal goal-seeking structures. There is no rational way of determining whether a "move" is good or bad the way one might judge a move in a game of chess, and thus no immediately apparent way to exercise critical judgement in a simulation.

This lack of internal goal-orientation carries with it a number of difficulties for anyone attempting to model art-making processes: for one thing, evaluation of the model must necessarily be informal. In the case of AARON, however, there has been extensive testing. Before describing the testing procedure it will be necessary to say with more care distinguishing here between the program's goals and my own — what AARON is supposed to do.

Task Definition.

It is not the intent of the AARON model to turn out drawings which are, in some ill-defined and loosely-understood sense, aesthetically pleasing, though it does in practise turn out pleasing drawings. It is to permit the examination of a particular property of freehand drawing which I will call, in a deliberately general fashion, standing-for-ness.

The Photographic "Norm"

One of the aims of this paper is to give clearer definition to what may seem intuitively obvious about standing-for-ness, but even at the outset the "intuitively obvious" will need to be treated with some caution. In particular, we should recognise that unguarded assumptions about the nature of "visual" imagery are almost certain to be colored by the XX th century's deep preoccupation with photography as the "normal" image-making mode. The view that a drawn image is either:

1. representational (concerned with the appearance of things), or

2. an abstraction (i.e. fundamentally appearance-oriented, but transformed in the interest of other aims) or,

3. abstract (i.e. it doesn't stand for anything at all),

betrays just this pro-photographic filtering, and is a long way from the historical truth. There is a great wealth of imagistic material which fits none of these paradigms, and it is by no means clear even that a photograph carries its load of standing-for-ness by virtue of recording the varying light intensities of a particular view at a particular moment in time.

It is for this reason that image-making will be discussed here as the set of modes which contains visual representation as one of its members. It is also why I used the word "evocative" in the first paragraph rather than "meaningful". My domain of enquiry here is not the way in which particular meanings are transmitted through images and how they are changed in the process, but more generally the nature of image-mediated transactions. What would be the minimum condition under which *a* set of" marks may function "as an image? This question characterises economically the scope of the enquiry, and it also says a good deal about how the word "image" is to be used in this paper, though a more complete definition must wait until the end.

Art-making and Image-making.

The reader may detect some reluctance to say firmly that this research deals with art-making rather than with image-making, or vice-versa. The two are presented as continuous. Art-making is almost always a highly sophisticated activity involving the interlocking of complex patterns of belief and experience, while in the most general sense of the term image-making appears to be as "natural" as talking. All the same, art-making is a case of image-making, and part of what AARON suggests is that art-making rests upon cognitive processes which are absolutely normal and perfectly common.

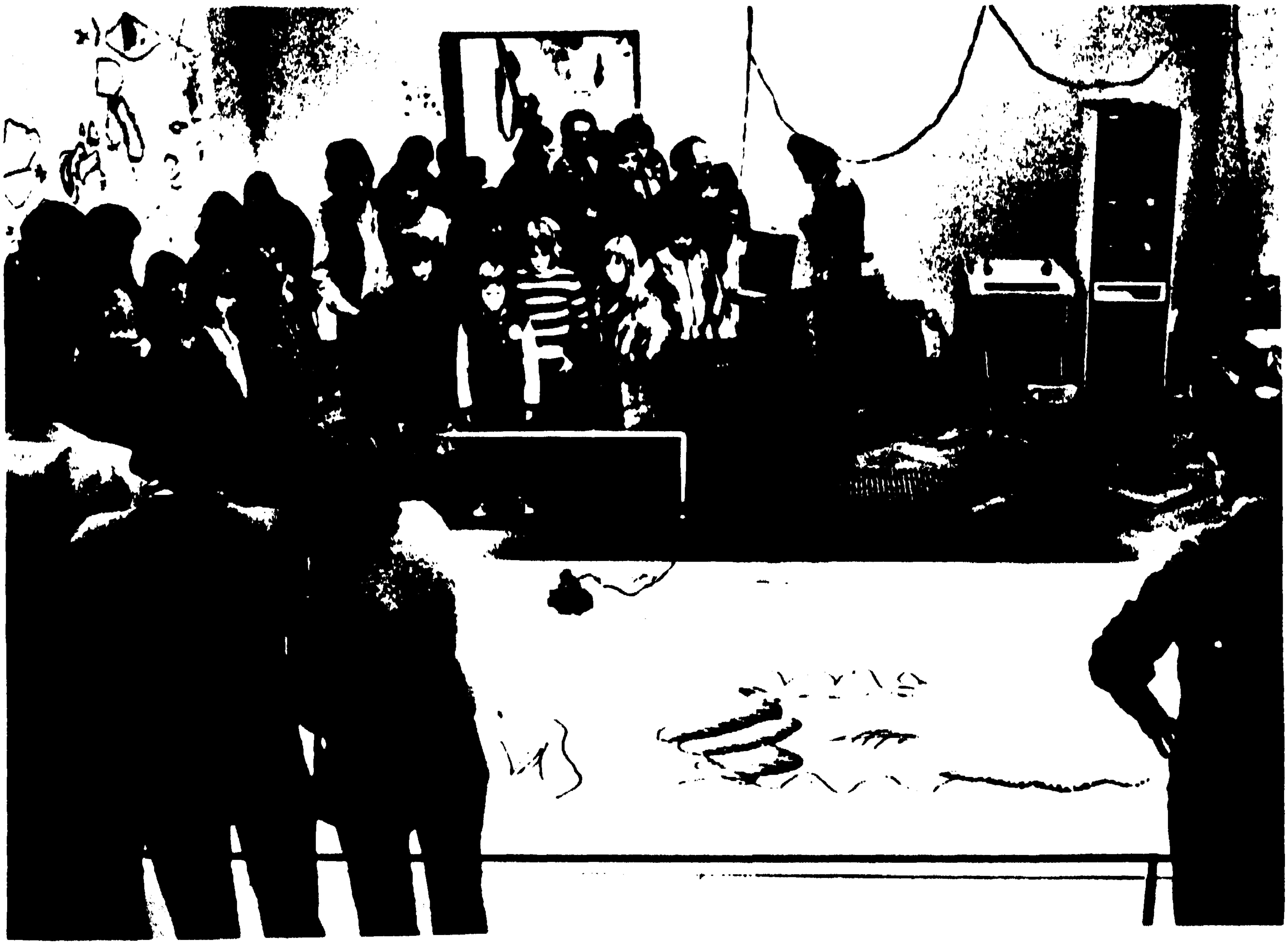


figure 3.

Evaluation.

A simulation program models only a small piece of the action, and it requires a context in which to determine whether it functions as one expects that piece to function. AAPON is not an artist. It simply takes over some of the functions which normally require an artist to perform them, and thus it requires the whole art-making process to be carried forward as a testing context. The program's output has to be acceptable to a sophisticated audience on the same terms as any other art, implying thereby that it must be seen as original and of high quality, not merely as a pastiche of existing work.

A valid testing procedure must therefore contain a sophisticated art-viewing audience, and the informal in situ evaluation of the simulation has been carried out in museum environments: the DOCUMENTA 6 international exhibition in Kassel, Germany, and the prestigeous Stedelijk Museum in Amsterdam, the two exhibits together running for almost five months and with a total audience of almost half a million museum-goers. In both of these shows drawings were produced continuously on a Tektronix 4014 display terminal and also with an unconventional hard-copy device (to be described later). A POP 11/34 ran the program in full view of the gallery visitors (fig 3).

In addition and at other times the program's output has been exhibited in a more orthodox mode in museums and galleries in the US and in Europe.

These exhibits were not set up as scientific experiments. Nor could they have been without distorting the expectations of the audience, and thus the significance of any results. No formal records were kept of the hundreds of conversations which took place between the artist and members of the audience. This report is therefore essentially narrative, but offered with some confidence.

<u>Audience Response</u>.

A virtually universal first assumption of the audiences was that the drawings they were watching being made by the machine had actually been made in advance by the "real" artist, and someow "fed" to the machine. After it had been explained that this was not the case viewers would talk about the machine as if it were a human artist. There appeared to be a general consensus that the machine exhibited a good-natured and even witty artistic personality, and that its drawings were quite droll (fig 4). Sane of the viewers, who knew my work from ray pre-computing, European, days claimed that they could "recognise my hand" in the new drawings. This last is particularly interesting, since, while I certainly made use of my own body of knowledge concerning image-making in writing the program, the <u>appearance</u> of my own work never consciously served as a model for what the program was supposed to do.

More to the point, while a very small number of people insisted that the drawings were nothing but a bunch of random squiggles, the majority clearly saw them in referential terms. Many would stand for long periods watching, and describing to each other what was being drawn: always in terms of objects in the real world. The drawings seem to be viewed mostly as landscapes inhabited by "creatures", which would be "recognised" as animals, fish, birds and bugs. Occasionally a viewer would "recognise" a landscape, and once the machine's home was identified as San Francisco, since it had just drawn Twin Peaks.

It might be correctly anticipated that on those other occasions when drawings have simply been framed and exhibited without any reference to their origins, the question of their origins has never arisen, and they have met with a typical cross-section of museum-goers responses

Even without formal evaluation, it might reasonably be claimed that the program provides a convincing simulation of human performance.

The next part of this paper is divided into five sections. In the first, a general description of the production system as a whole is given. The following three sections deal with particular parts of the production system: the MOVEMENT CONTROL part, the PLANNING part, and the part which handles the internal representation of the drawings as they proceed. The second of these, on PLANNING, also gives an account of the theoretical basis for the program. The fifth section has something to say about the function of randomness in the program, and also discusses to what extent it might be thought to parallel the use of randomness in human art-making behavior. The third and final part draws conclusions.

****

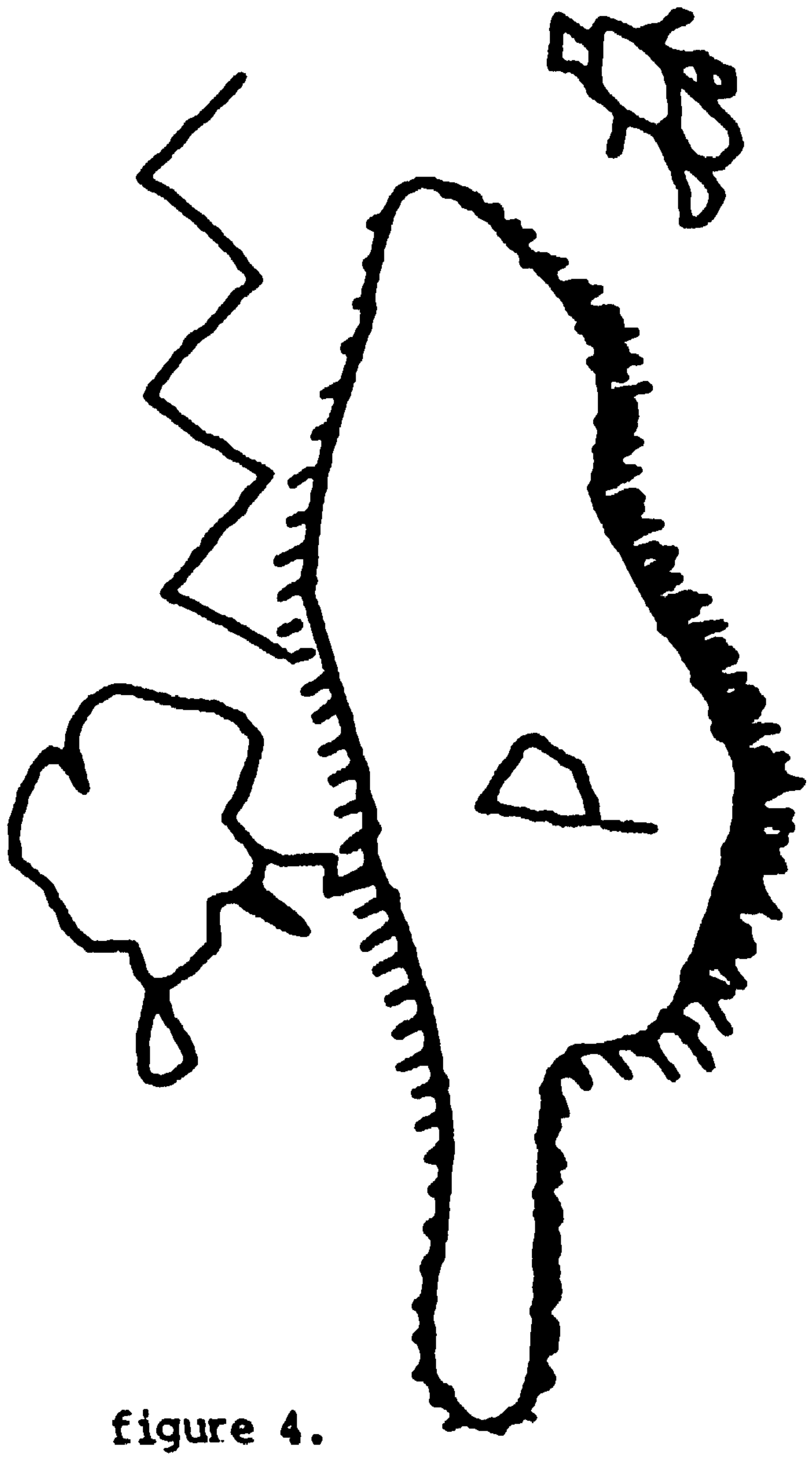


figure 4.

2. TOE PROGRAM "AARON"

24 THE PRODUCTION SYSTEM.

The main program (note 5) has about three hundred productions. Many of these are to be regarded as micro-productions in the sense that each of them handles only a small part — an -action-atom- — of a larger conceptual unit of action. For example, the drawing of a single line, conceptually a single act, actually involves twenty or thirty productions on at least three levels of the system. This fine-grain control over the drawing process subscribes both to its generality — most of these action-atoms are invoked by many different situations — and to its flexibility, since it allows a process to be interrupted at any point for further consideration by higher-level processes.

Levels of Organisation.

The organisation of the system is heirarchical, in the sense that the higher levels are responsible for decisions which constrain the domain of action for the lower levels (fig 5). Each level of the system is responsible only for its own domain of decison-making, and there is no conceptual homunculus sitting on the top holding a blueprint and directing the whole operation. No single part knows what the drawing should turn out to be like. There is some practical advantage to this level-wise splitting up of the system, but the program was designed this way primarily for reasons of conceptual clarity, and from a desire to have the program structure itself — as well as the material contained within it — reflect my understanding of what the human image-making process might be like. I believe that the constant shifting of attention to different levels of detail and conceptualisation provides this human process with some of its important characteristics. Thus the left part of each production searches for combinations of up to five or six conditions, and each right part may perform an arbitrary number of actions or action-atoms, one of which may involve a jump to another level of the system.

"ARTWORK"

The topmost level of the system, the ARTWORK level, is responsible for decisions relating to the organisation of the drawing as a whole. It decides how to start, makes some preliminary decisions which may later determine when and how it is to finish, and eventually makes that determination. The program currently has no archival memory, and begins each drawing as if it has never done one before. (One can easily imagine the addition of a higher level designed to model the changes which the human artist deliberately makes in his work from one piece to the next: this level would presumably be called EXHIBITION.)

ARTWORK also handles some of the more important aspects of spatial distribution. It is my belief that the power of an image to convince us that it is a representation of some feature of the visual world rests in large part upon the image's fine-grain structure: the degree to which it seems to reflect patterns in the changes of information density across the field of vision which the cognitive processes themselves impose upon visual experience.

Put crudely, this means, for example, that a decision on the part of the reader of an image that one set of marks is a detail of another set of marks rather than standing autonomously, is largely a function of such issues as relative scale and proximity. This function is quite apart from the more obviously affective issue of shape (and hence "semantic") relationship. It is the overall control of the varying density of information in the drawing, rather than the control of inter-figural relationships, which is handled by ARTWORK.

"MAPPING" and "PLANNING-

All problems involving the finding and allocation of space for the making of individual elements in the drawing is handled by MAPPING, though its functions are not always heirarchically higher than those of PLANNING, which is responsible for the development of these individual figures. Sometimes PLANNING may decide on a figure and ask MAPPING to provide space, while at other times MAPPING may announce the existence of a space and then PLANNING will decide what to do on the basis of its availability. Sometimes, indeed, MAPPING may override a PLANNING decision by announcing that an appropriate space is not available. A good example of this occurs when PLANNING decides to do something inside an existing closed figure and MAPPING rules that there isn't enough room, or that what there is is the wrong shape.

MAPPING will be referred to again in relation to the data-structures which constitute the program's internal representation of what it is doing, and PLANNING also as one of the centrally important parts of the program.

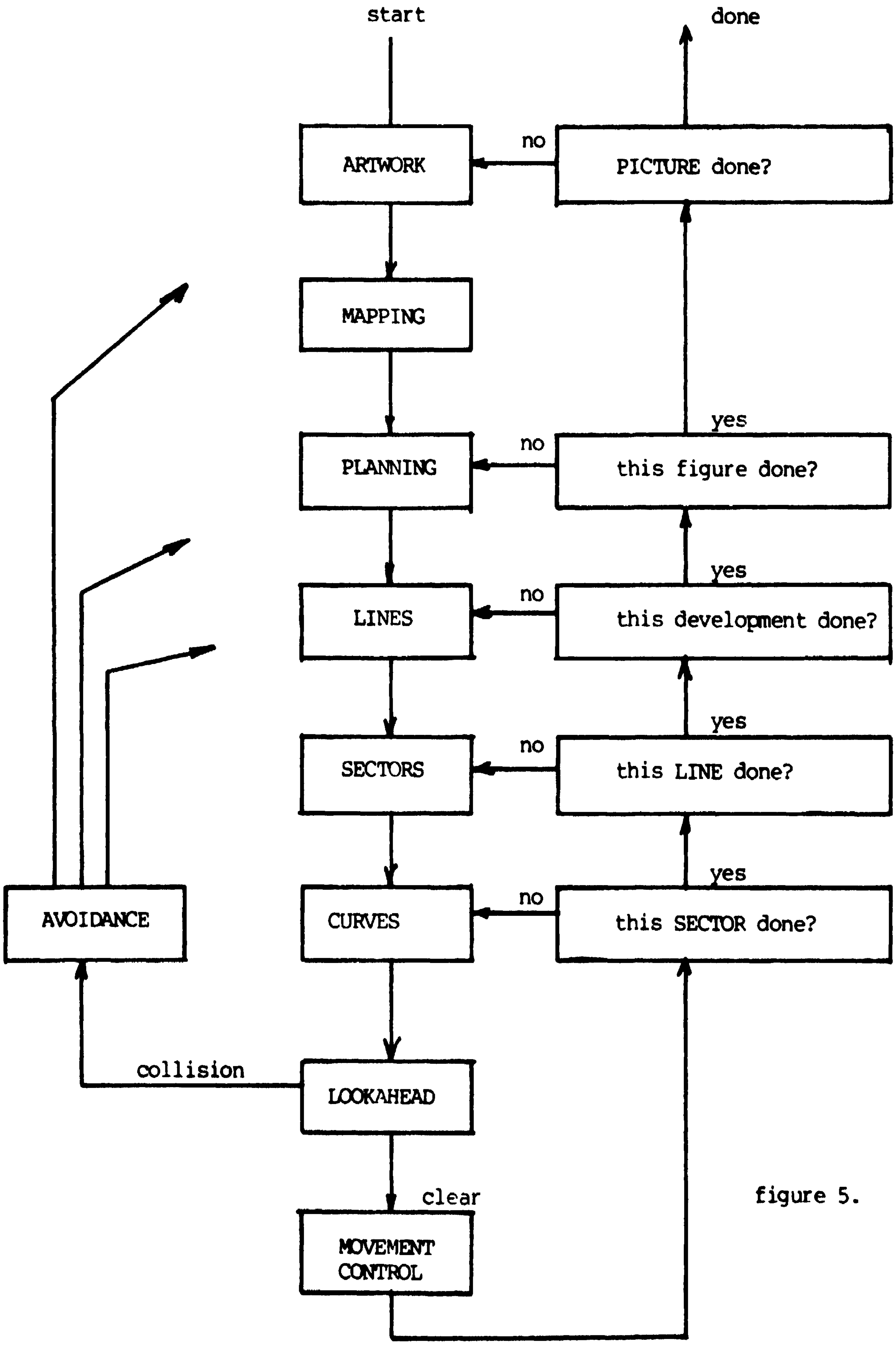

start
done
ARTWORK
no
PICTURE done?
MAPPING
yes
no
PLANNING
this figure done?
yes
no
LINES
this development done?
yes
no
SECTORS
this LINE done?
yes
no
AVOIDANCE
CURVES
this SECTOR done?
collision
LOOKAHEAD
clear
MOVEMENT CONTROL

figure 5.

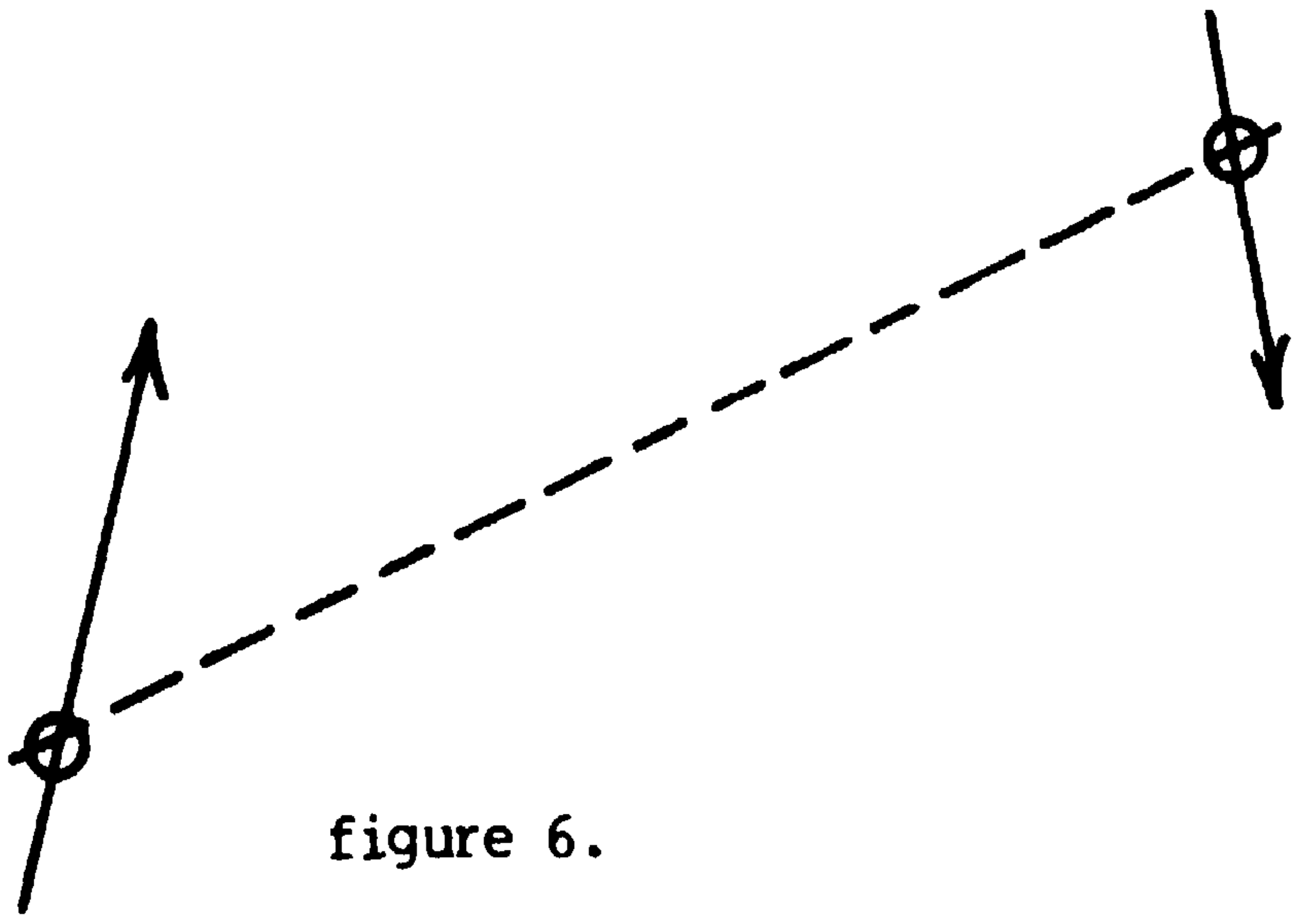


figure 6.

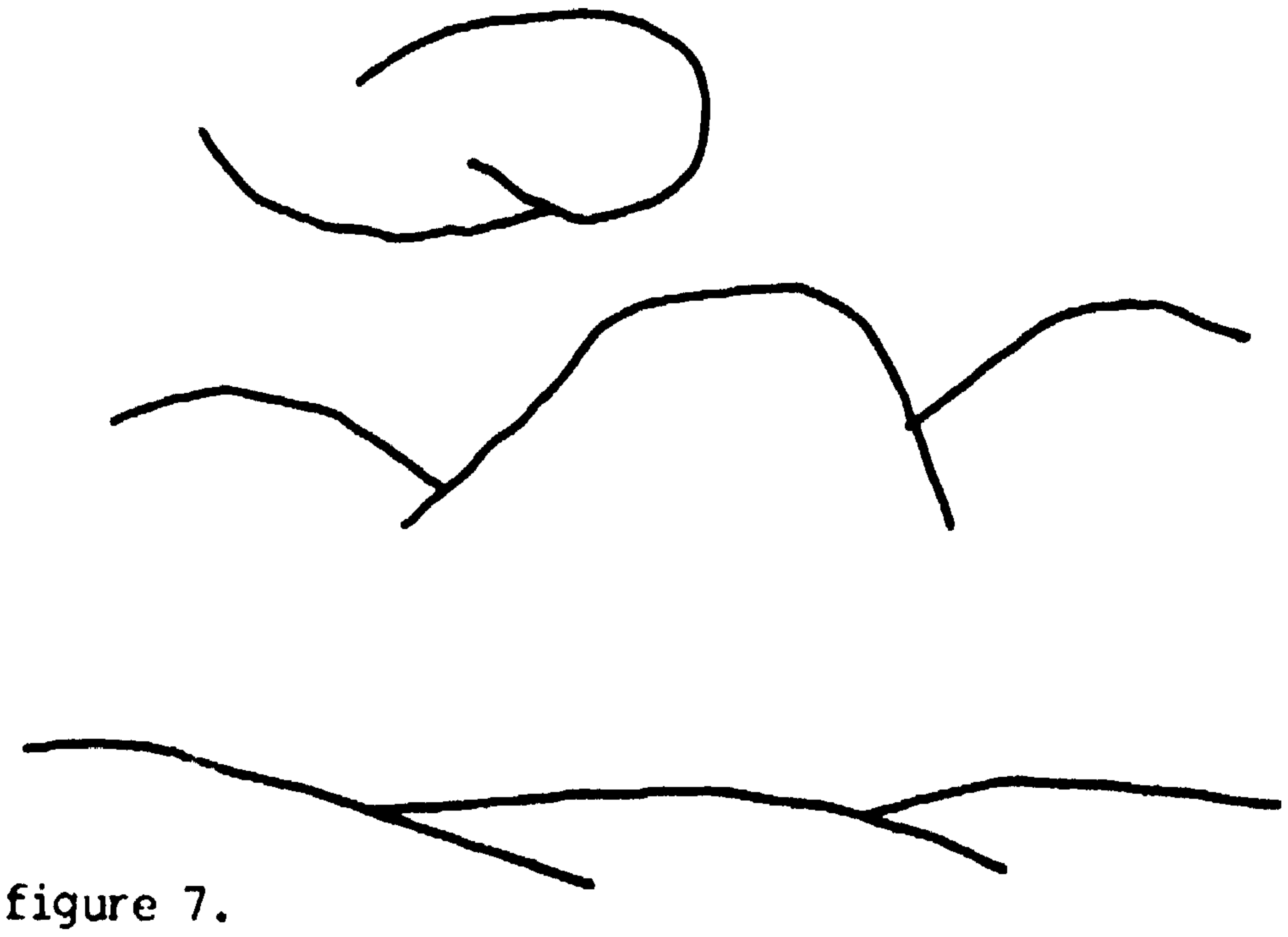


figure 7.

LINES AND SECTORS

Below this level the heirarchical structure of the system is fairly straightforward. Each figure is the result of (potentially) several developments, each provided by a return of control to PLANNING. Each of these developments may consist of several lines, and for each of the successive lines of each development of any figure LINES must generate a starting point and an ending point, each having a direction associated with it (fig 6). It also generates a number of parameters on the basis nf specifications drawn Up in PLANNING which determine how the line is to be drawn: whether reasonably straight, wiggly, or strongly curved, and, if various overlapping modes are called for (fig 7), how they are to be handled.

As I have indicated, lines are not drawn as the result of a single production. When LINES passes control to SECTORS the program does not know exactly where the line will go, since the constraint that it must start and end facing specified directions does not specify a path: there are an indeterminate number of paths which would satisfy the constraint. The program does not choose one, it generates one. SECTORS produces a series of "imagined" partial destinations — signposts, as it were (fig 8) — each designed to bring the line closer to its final end-state. On setting up each of these signposts it passes control to CURVES, whose function is to generate a series of movements of the pen which will make it veer towards, rather than actually to reach, the current signpost. Control is passed back to SECTORS when the pen has gone far enough towards the current signpost that it is time to look further ahead, and it is passed back to LINES when the current line has been completed and a new one is demanded by the development still in progress.

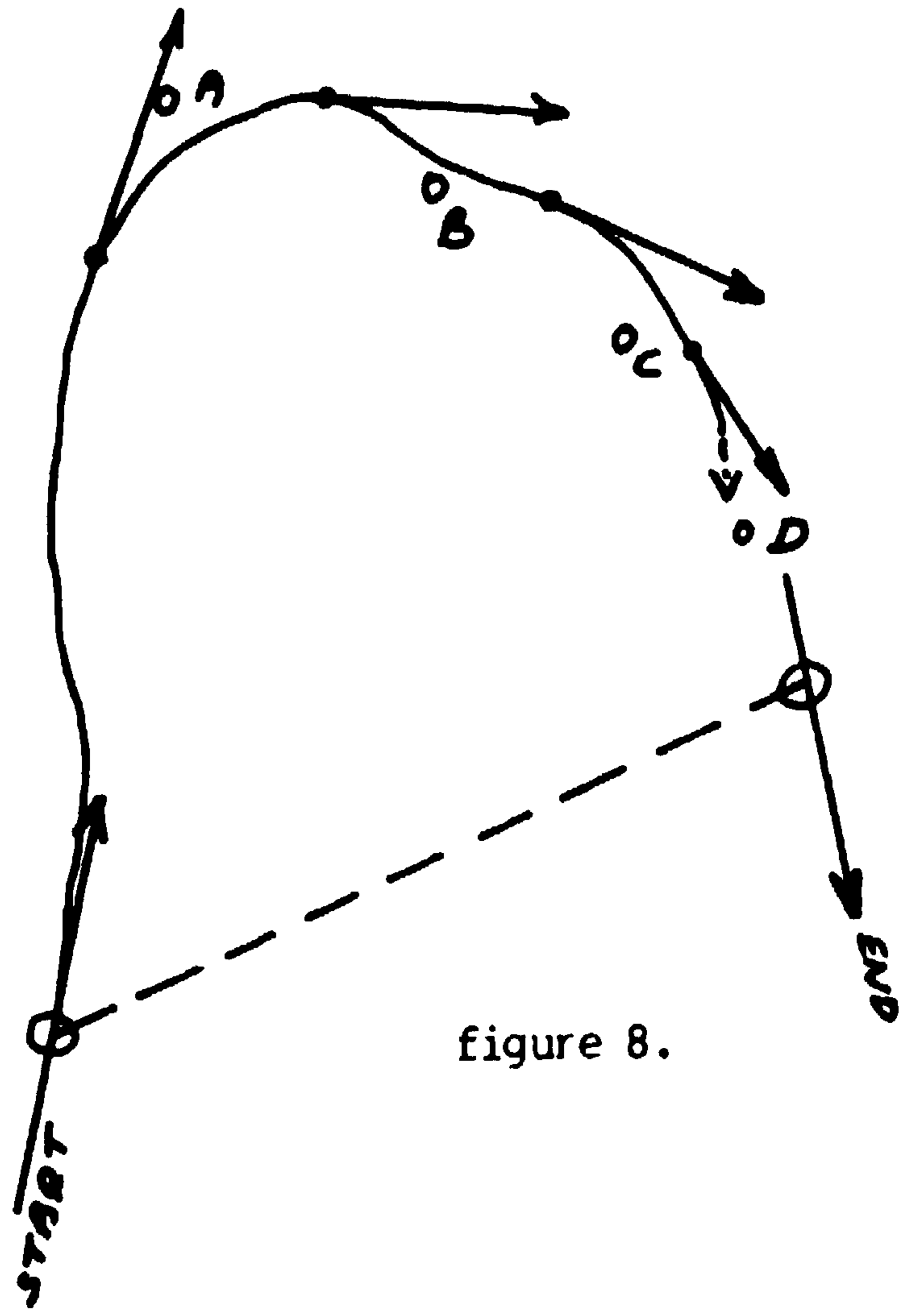


figure 8.

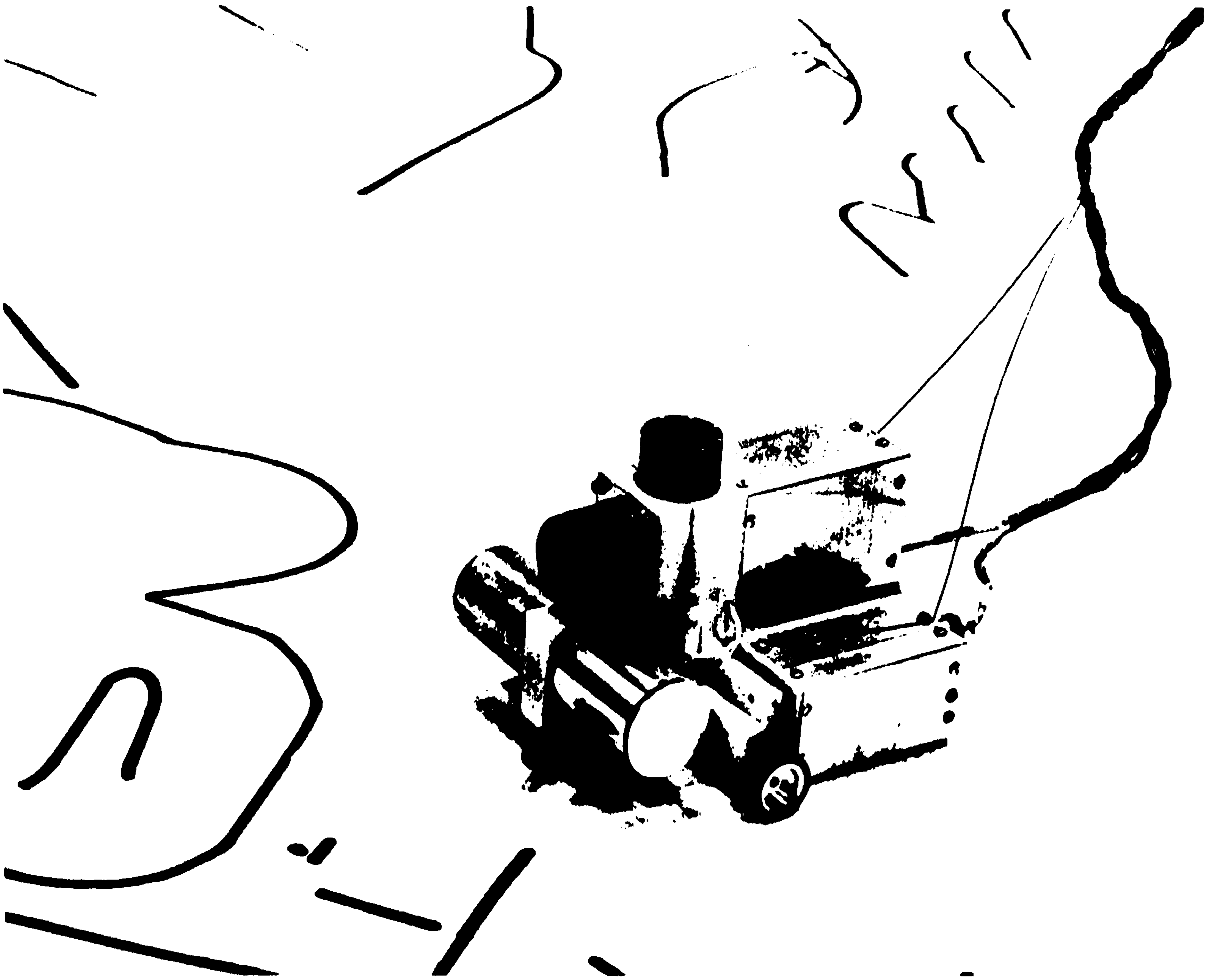
2.2 MOVEMENT CONTROL

We are now down to the lowest level of the program, and to the way in which the curves which make up the drawing are actually generated. This part is not discontinuous from the rest, of course. The flexibility of the program rests in large part upon the fact that the heirarchy of control extends downwards to the finest-grained decisions: no part of the control structure is considered to be so automatic that it should fall below the interface line. Thus, the story of how the pen gets moved around follows on from the description of how the intermediate signposts are set up.

Abstract Displays and Real Devices

In the earlier versions of the program all the development work was done exclusively on a graphic display, and the "pen" was handled as an abstract, dimensionless entity without real-world constraints upon its movements. Conceptually, however, I always thought of the problem of moving the pen from point A facing direction alpha, to point B facing direction beta, as being rather like the task of driving a car off a main road into a narrow driveway set at some known arbitrary angle to it. This is clearly not a dead-reckoning task for the human driver, but one which involves continuous feedback and a successive-approximation strategy.

It seemed quite reasonable, therefore, to be faced at some point with the problem of constructing an actual vehicle which would carry a real pen and make real drawings on real sheets of paper. That situation arose in the Fall of '76 when I was preparing to do the museum exhibitions which I mentioned earlier, and decided that if I wanted to make the drawing process visible to a large number of people simultaneously, I would need to use something a good deal bigger than the usual graphic display with its 20-inch screen.



The Turtle,

The answer turned out to be a small two-wheeled turtle (fig 9), each of its wheels independently driven by a stepping motor, so that the turtle could be steered by stepping the two motors at appropriate rates, it is thus capable of drawing arcs of circles whose radius depends upon the ratio of the two stepping rates.

Since the two wheels can be driven at the same speed in opposite directions, the turtle can be spun around on the spot and headed off in a straight line, so that this kind of device is capable of simulating a conventional x-y plotter. But it seemed entirely unreasonable to have built a device which could be driven like a car and then use it to simulate a plotter. In consequence the pen-driving procedures already in the program were re-written to generate the stepping rates for the motors directly — to stay as close as possible to the human model's performance — rather rather than calculating these rates as a function of decisions already made.

The advantage here was a conceptual one, with some practical bonus in the fact that the turtle does not spend a large part of its time spinning instead of drawing. It also turned out unexpectedly that the generating algorithm simplified enormously, and the quality of the freehand simulation improved noticeably.

Feedback.

The program does not now seek to any place — in cartesian terms — but concerns itself exclusively with steering: thus the turtle's cartesian position at the end of executing a single command is not known in advance. Nor is this calculation necessary when the turtle is operating in the real world. It was not designed as a precision drawing device, and since it cannot perform by dead-reckoning for long without accumulating errors, the principle of feedback operation was extended down into this real-world part of the program. The device makes use of a sonar navigation system (fig 10) by means of which the program keeps track of where it actually is. Instead of telling it to "go to x,y" as one would tell a conventional plotter, the program tells it "do the following and then say where you are".

A more detailed account of the turtle system, and it's effect upon the simulation of freehand drawing dynamics, is given in Appendix 1.

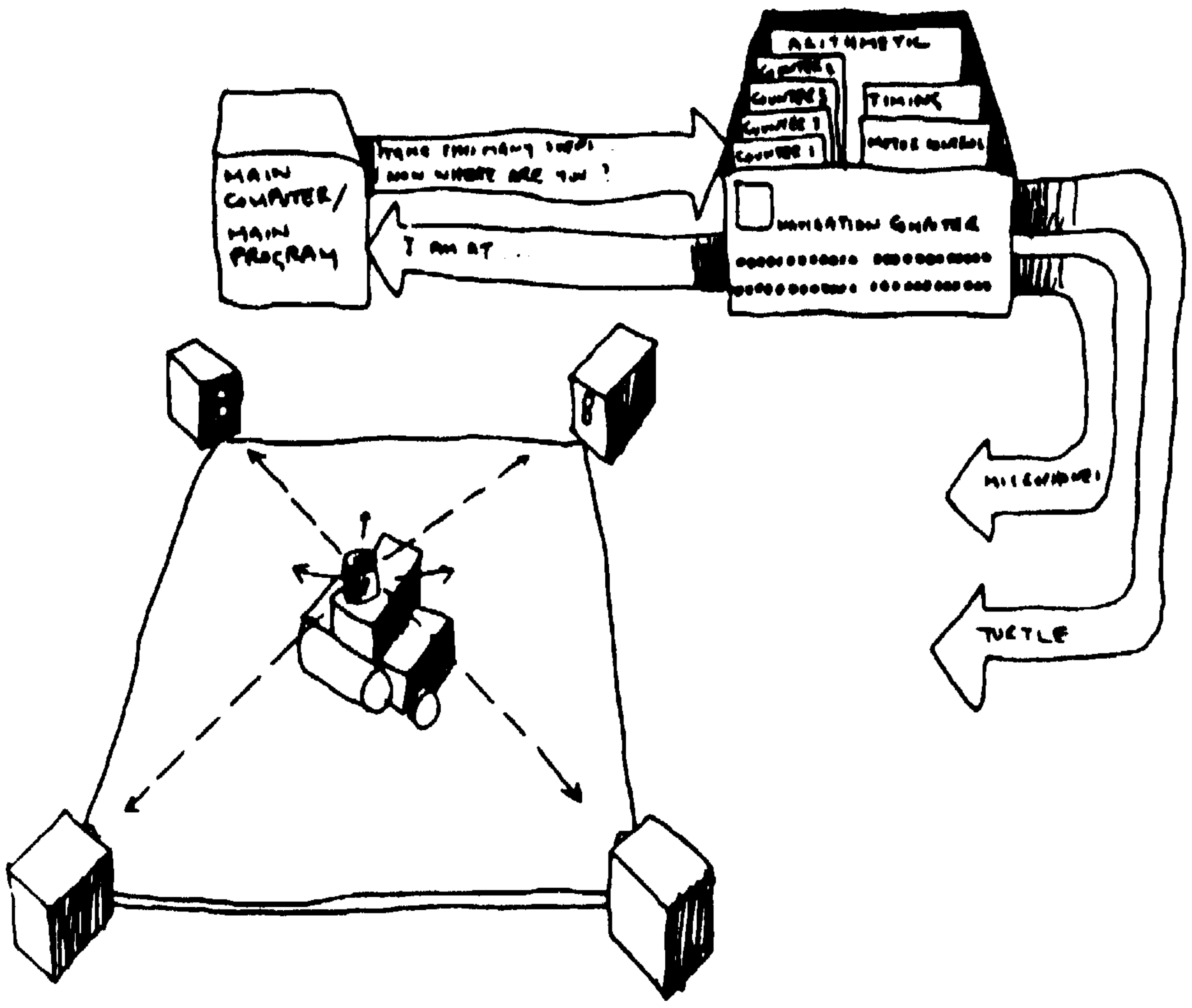


figure 10.

2.3 "PLANNING"

No single level of the program can be described adequately without reference to the other levels with which it interacts: it has already been mentioned, for example, that MAPPING may either precede PLANNING in determining what is to be done next, or it may serve PLANNING by finding a space specified there. Additionally, any development determined in PLANNING may be modified subsequently either as a result of an imminent collision with another figure or because provision exists in the program for "stacking" the current development in order to do something not originally envisaged (fig 11a,b). All the same, the drawing is conceived predominantly as an agglomeration of figures, and to that extent PLANNING, which is responsible for the development of individual figures, is of central importance.

Behavioral Protocols in Image-Making.

Of the entire program, it is also the part least obviously related to the effects which it accomplishes. While the formal results of its actions are clear enough — an action calling for the closure of a shape will cause it to close, for example — it is not at all clear why those actions result in the specifically evocative quality which the viewer experiences.

A rule-by rule account of this effect is not appropriate, because the individual rules do no more than implement conceptual entities — which I will call behavioral protocols — which are the fundamental units from which the program is built. These protocols are never explicitly stated in the program, but their existence is what authorises the rules. Thus, before describing in detail what is in PLANNING I should give an account of the thinking which preceeded the writing of the program, and try to make clear what I mean by a protocol.

Background.

It is a matter of fact that by far the greatest part of all the imagery to which we attach the name of "art" comes to us from cultures more or less remote from our own. It is also a matter of fact that within our own culture, and in relation to its recent past, our understanding of imagery rest to a great extent upon prior common understandings, prior cultural agreements, as to what is to stand for what — prior, that is to say, to the viewing of any particular image. It is unlikely that a Renaissance depiction of the Crucifixion ("of Christ" being understood here by means of just such an agreement!) would carry any great weight of meaning if we were not already familiar both with the story of Christ and with the established conventions for dealing with the various parts of the story. Indeed, we might be quite confused to find a depiction of a beardless, curly-headed youth on the cross unless we happened to possess the now-abstruse knowledge that Christ was depicted that way — attaching a new set of meanings to the old convention for the representation of Dionysus — until well into the 7th century. In general, we are no longer party to the agreements which make this form acceptable and understandable. We must evidently distinguish between what is understandable without abstruse knowledge — we can, indeed, recognise the figure on the cross as a figure — and what is understandable only by virtue of such knowledge.

In the most general sense, all cultural conditions are remote from us, and differ only in the degree of their remoteness. We cannot really comprehend why the Egyptians made sphinxes, what Michelangelo thought the ancient world had in common with Christianity, or how the internal combustion engine was viewed by the Italian Futurists seventy years ago who wanted to tear down the museums in its honor. What abstruse knowledge we can gain by reading Michelangelo's writings, or the Futurist Manifesto, does not place us into the cultural

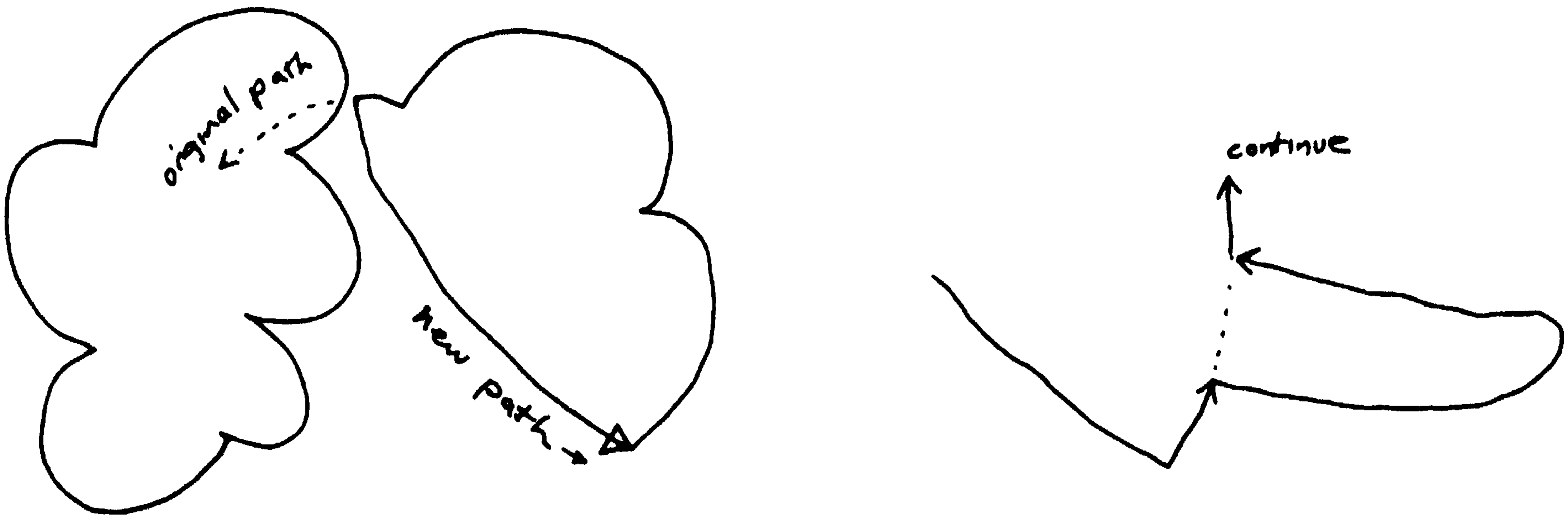


figure 11a,b.

environment in which the work is embedded. A culture is a contiuium, not a static event: its understandings and meanings shift constantly, and their survival may appear without close scrutiny to be largely arbitrary. In the extreme case, we find ourselves surrounded by the work of earlier peoples so utterly remote from us that we cannot pretend to know anything about the people themselves, much less about the meanings and purposes of their surviving images.

The Paradox of Insistent Meaningfulness.

There is an implicit paradox in the fact that we persist in regarding as meaningful — not on the basis of careful and scholarly detective work, but on a more directly confrontational basis — images whose original meanings we cannot possibly know, including many that bear no explicitly visual resemblance to the things in the world. Presumably this state of affairs arises in part from a fundamental cultural egocentrism — what, we ask, would we have intended by this image and the act of making it? — which is fundamentally distortive. There has also been a particular confusion in this century through the widespread acceptance of what we might call the telecommunications model of our transactions through imagery, particularly since in applying that model no differentiation has been observed between the culture we live in and the cultures of the remote past. In the view of this model, original meanings have been encoded in the image, and the appearance of the image in the world effects the transmission of the meanings. Allowing for noise in the system — the inevitability of which gives rise to the notion, in art theory, of "interpretation" — the reception and decoding of the image makes the original meanings available.

However useful the model is as a basis for examining real telecommunication-like situations, in which the intended meanings and their transformations can be known and tracked, it provides a general account of our transactions through images which is quite inadequate. The encoding and decoding of messages requires access to the same code-book by both the image-maker and the image-reader, and that code-book is precisely what is not carried across from one culture to another.

I think it is clear also that the paradox of insistent meaning fulness, as we might call it, constitutes the normal condition of image-mediated transactions, not an abnormal condition. It evidently extends below the level at which we can recognise the figure, but not what the figure stands for, since so much of the available imagery is not in any very obvious sense "representational" at all. The paradox is enacted every time we look at a few marks on a scrap of paper and proclaim them to be a face, when we know perfectly well that they are nothing of the sort.

Cognitive Bases for Image Structure.

In short, my tentative hypothesis in starting work on AARON was that all image-making and all image-reading is mediated by cognitive processes of a rather low-level kind, presumably processes by means of which we are able to cope also with the real world. In the absence of common cultural agreements these cognitive processes would still unite image-maker and image-viewer in a single transaction. On this level — but not on the more complex culture-bound level of specific iconological intentionality — the viewer's egocentricity might be justified, since he could correctly identify cognitive processes of a familiar kind in the making of the image. But let me detail this position with some care. I am not proposing that these processes make it possible for us to understand the intended meanings of some remotely-generated image: I am proposing that the intended meanings of the maker play only a relatively small part in the sense of meaningfulness. That sense of meaningfulness is generated for us by the structure of the image rather than by its content.

I hope I may be excused for dealing in so abbreviated a fashion with issues which are a good deal less than self-evident. The notion of non-enculturated behavior — and that notion lurks behind the last few paragraphs, obviously — is a suspect one, since all human behavior is enculturated to some degree: but my purpose was not to say what part of human behavior is dependent upon enculturating processes and what is not. It was simply to identify some of the determinants to a general image-structure which could be seen to be common to a wide range of enculturating patterns. The implication seemed strong — and still does — that the minimum condition for generating a sense of meaningfulness did not need to include the assumption of an intent to communicate: that the exercise of an appropriate set of these cognitive processes would itself be sufficient to generate a sense of meaningfulness.

Cognitive Skills,

The task then was to define a suitable set. I have no doubt that the options are wide, and that my own choices are not exclusive. I chose at the outset to include:

1. the ability to differentiate between figure and ground,

2. the ability to differentiate between open and closed forms, and

3. the ability to differentiate between insideness and outsideness (note 6).

AARON has developed a good deal from that starting point, and some of its current abilities clearly reflect highly enculturated patterns of behavior. For example, the program is now able to shade figures in a mode distinctly linked to Renaissance and post-Renaissance representational modes: other cultures have not concerned themselves with the fall of light on the surfaces of objects in the same way. Nevertheless, a large part of the program is involved still in demonstrating its awareness of the more primitive differentiations.

Protocols and Rules.

Against this background, I use the term protocol to mean the procedural instantiation of a formal awareness. This is clearly a defination which rests upon cognitive, rather than perceptual, modes, since it involves the awareness of relational structures. Thus, for example, the program's ability to differentiate between form and ground makes possible an awareness of the spatial relationships between forms, and generates finally a set of avoidance protocols, the function of which is to prohibit the program from ignoring the existence of one figure in drawing another one. The protocols themselves are not explicitly present in the program, and are manifested only through their enactment by the rules which describe what to do in particular circimstances where the overlapping of figures is threatened.

Figure Development

In keeping with the heirarchical structuring which informs the program as a whole, PLANNING considers a figure to be the result of a number of developments, each determined in part by what has gone before. The program enacts a number of repetition protocols, and a single development in the making of a figure can often involve the repetition of a single action (fig 12), rather than the agglomeration of different actions. The first productions to deal with the first development of any figure decide, on the basis of frequency considerations, that this figure will be closed, that it will be open, or that it will be, for the moment, "unconmitted" — that is, a line or a complex of lines will be drawn, but only at a later stage will it be decided whether or not to close. If the primary decision is for closure, then PLACING will decide between a number of options, mostly having to do with size and shape — MAPPING permitting — and with configuration. In some cases it will not actually draw the boundary of a closed form at all, and will leave the definition of the occupied space to await subsequent space-filling moves.

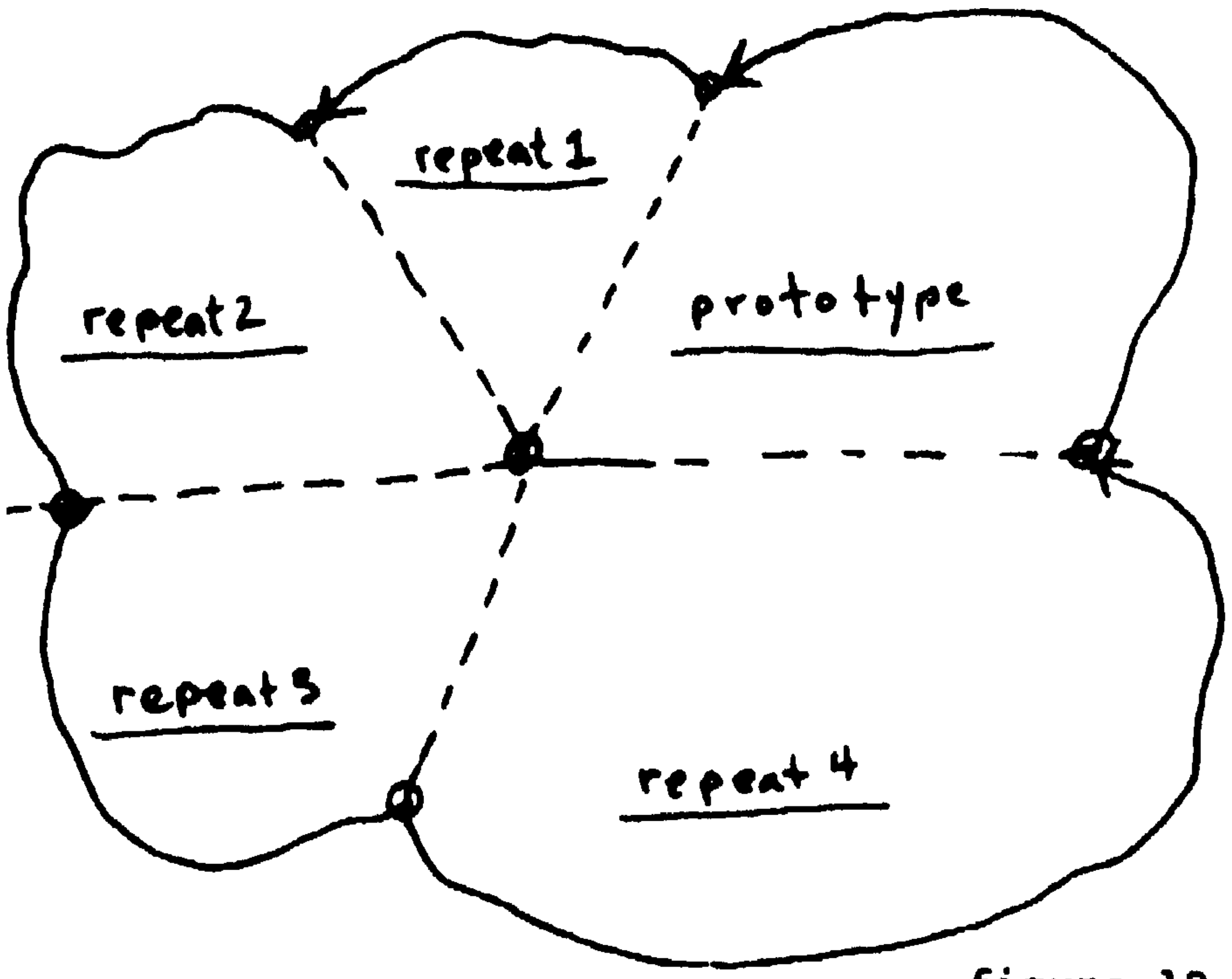


figure 12.

If the decision is for a non-closed form, then again a number of options are open. In both cases the available options are stated largely in terms of repetition protocols, the enactment of which determines the formal characteristics of the resulting configuration. These characteristics are not uniquely defining, however, and a number of different formal sub-groups may result from a single repetition protocol and its rules. For example, one such protocol, involving a single line in this case, requires the line to move a given distance (more-or-less) and then change direction, continuing this cycle a given number of times. All the figures marked in (fig 13) result from this: the details of implementation in the individual cases are responsive to their unique environmental conditions, and in any case may be changed at any point by the overriding avoidance protocol, which guarantees the territorial integrity of existing figures.

Thus the program will know at the beginning of each development what the current intention is, but will not know what shape will result. A closed form generated by a "go,turn, repeat" cycle may in fact turn out to be extremely long and narrow (fig 14), and a number of second developments associated with a closed-form first development will then be unavailable: there will be a limit, for example, upon what can be drawn inside it, though it may develop in other ways, as this one does.

Proliferation.

Even with constraints of this sort there is a significant proliferation in the number of productions associated with the second development of any figure. A typical first development might be initiated by:

```
It (this is a first development
   and the last figure was open
   and at least n figures have been done
   and at least q of them were open
   and at least t units of space are now
   available)

 Then
 This figure will be closed
         specifications for repetition
         specifications for configuration
         to move on from this point:
         If (this is a second development
            and the first was closed
            and its properties were
                 a. (size)
                 b. (proportions)
                 c. (complexity)
                 d. (proximity to ...)

Then either
         1. divide it
         ... specifications ...
         or
         2. shade it
         ... specifications...
         or
         3. add a closed form to it
         ... specifications...
         4. do a closed form inside
         ... specifications...
         or
         5. do an open form inside
         ... specifications...
```

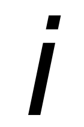


figure 13.

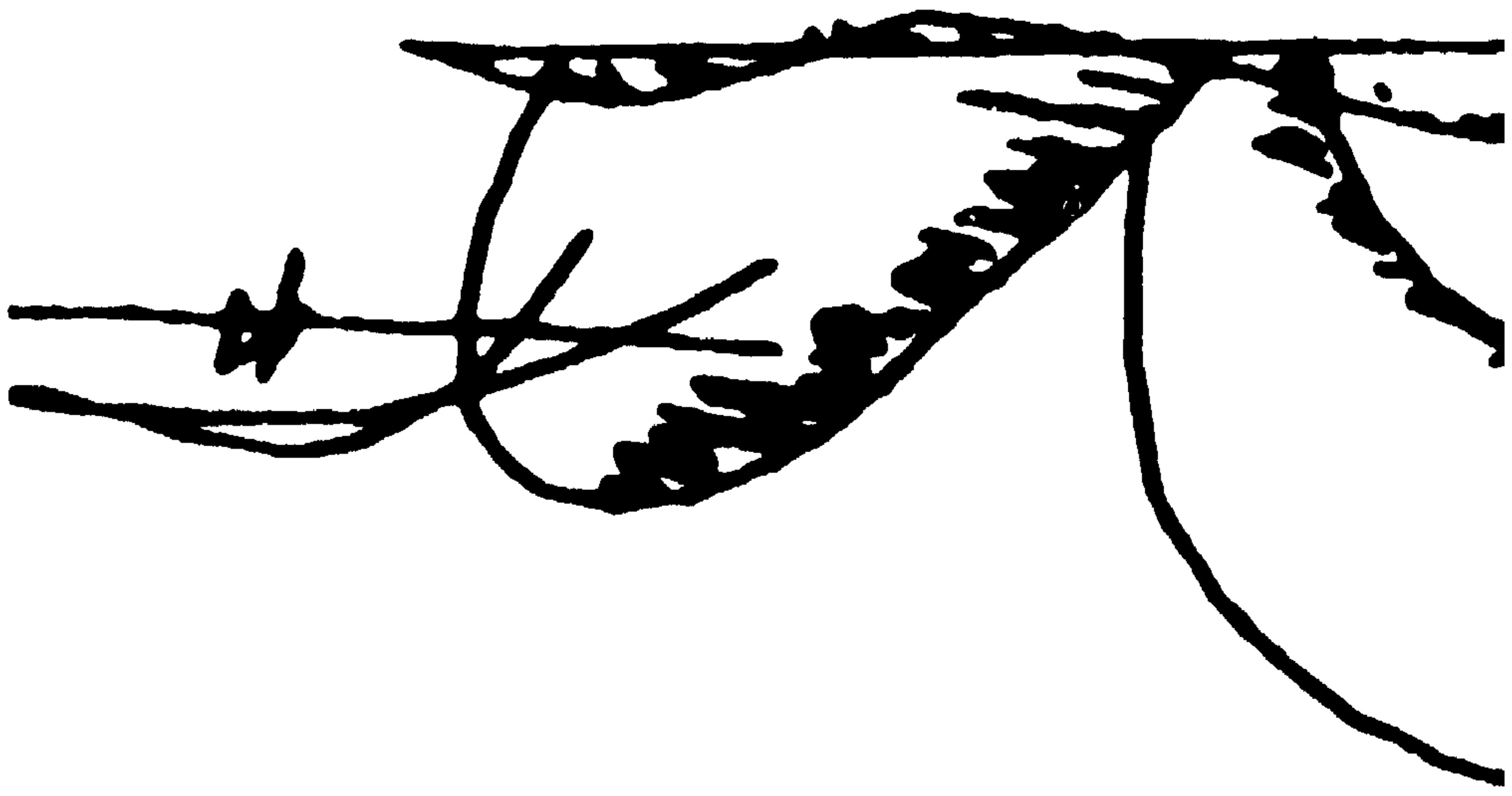


figure 14.

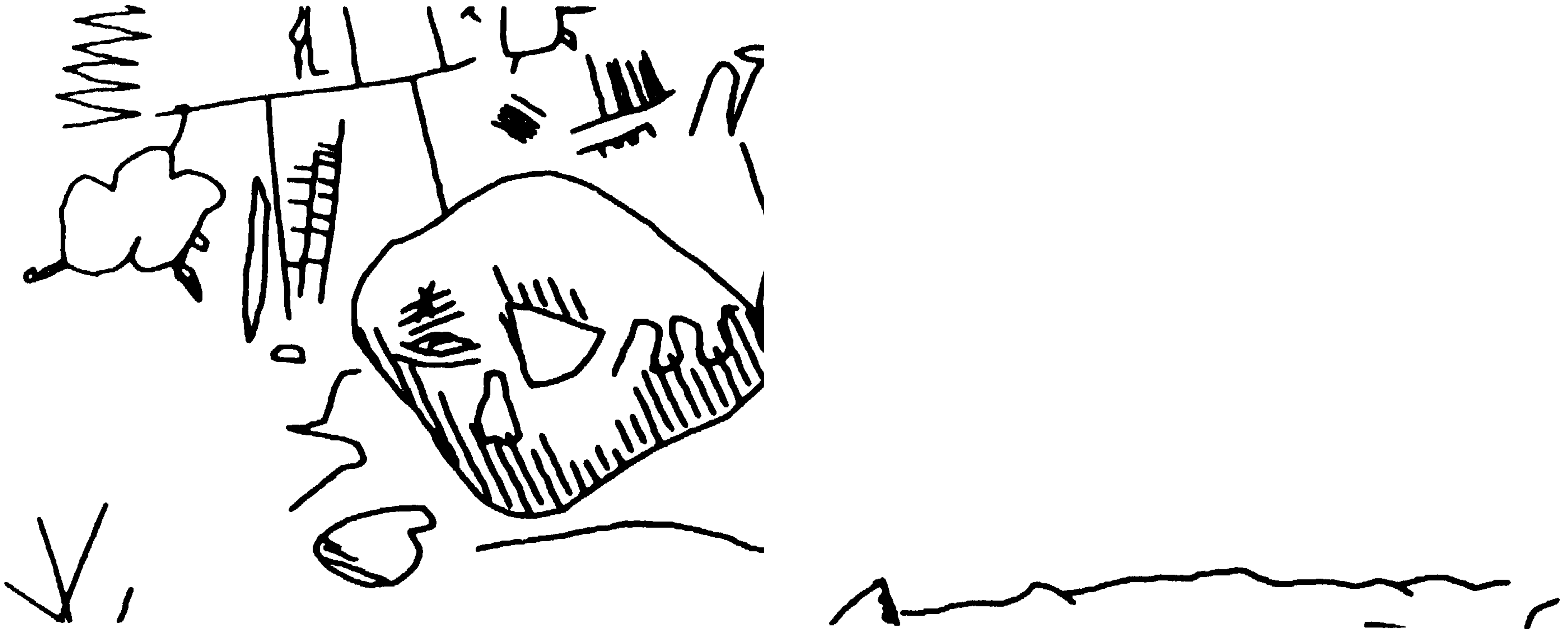


figure 15.

This is a prototype for an expanding class of productions, each responding to a different combination of properties in the first development. Similarly, continuation will require...

```
If (this is a third development
   and the first was a closed form
   ... properties...
   and the second was a closed form
   ... properties... )

 Then
 shade the entire figure:
         specification 1:
            a boulder with a hole in it
         or
         specification 2:
            a flat shape with a hole
         or
         specification 3:
            a penumbra.

If (this is a third development
   and the first was closed
   and the second was a series of
   parallel lines inside it ...
   and the remaining inside space is at
   least s... )

 Then
 do another series of lines:
         specification 1:
            perpendicular to first . .
         or
         specification 2:
            alongside the first...
         or
         specification 3:
            do a closed form in available
            space...
```

(note 7).

The Relationship of Closed Forms and Open Forms.

The same proliferation of options occurs for open-lined structures also, but not to the same degree. One of the interesting things to come out of this program is the fact that open-line structures appear to function quite differently when they are alone in an image than when they appear in the presence of closed forms. There seems to be no doubt that closed forms exert a special authority in an image — perhaps because they appear to refer to objects — and in their presence open-lined structures which in other circumstances might exert similar pressure on the viewer are relegated to a sort of spatial connective-tissue function. A similar context-dependancy is manifested when material is presented inside a closed form (fig 15): it is "adopted", and becomes either a detail of the form, or markings upon it. This seems to depend upon particular configurational issues, and especially the scale relationship between the "parent" form and the newly introduced material. This manifestation is important, I believe, in understanding why we are able to recognise as "faces" so wide a range of closed forms with an equally wide range of internal markings following only a very loose distribution specification.

Limits on Development.

At the present time no figure in the program goes beyond three developments, and few go that far, for a number of reasons. In the first place, most of the (formal) behavior patterns in the program were initially intended to model a quite primitive level of cognitive performance, and for most of these a single development is actually adequate. Once a zig-zag line has been generated, repetition, for example — as it is found in existing primitive models — seems limited to those shown in (fig 16).

It has remained quite difficult to come up with new material general enough for the purposes of the program. It is the generality of the protocols which guarantees the generality of the whole, and new material is initiated by the introduction of new protocols. On the level of the procedures which carry out the action parts of the subsequently-developed productions, the approach has been to avoid accumulation of special routines to do special things. There is only one single procedure adapting the protocols of repetition and reversal to the generation of a range of zigzag-like forms, for example (fig 13).

But there has been another, and equally significant reason, for the limitation upon permissible developments. It is the lack of adequate, and adequately important, differentiations in the existing figures. For the primitive model represented by the earlier states of the program it was almost enough to have a set of abilities called up by the most perfunctory consideration of the current state of the drawing: the stress was on the definition of a suitable set of abilities (as represented by the right-hand parts of the productions), and as it turned out it was quite difficult to exercise those abilities without generating moderately interesting results. But for a more sophisticated model it is clearly not enough merely to extend that set of abilities, and the problem of determining why the program should do this rather than that becomes more pressing.

The limitation here can be considered in two ways. One is that I had reached the point of exhausting temporarily my own insights into the image-building process. The other is that I had not made provision in the first versions of the program for being able to recognise the kind of differentiations I would want to deal with — since I could not know at the outset what they were going to be — and thus lacked a structure for developing new insights. This leads to a consideration of my next topic: how the program builds its own representation of what it has done up to any point in the making of the drawing.

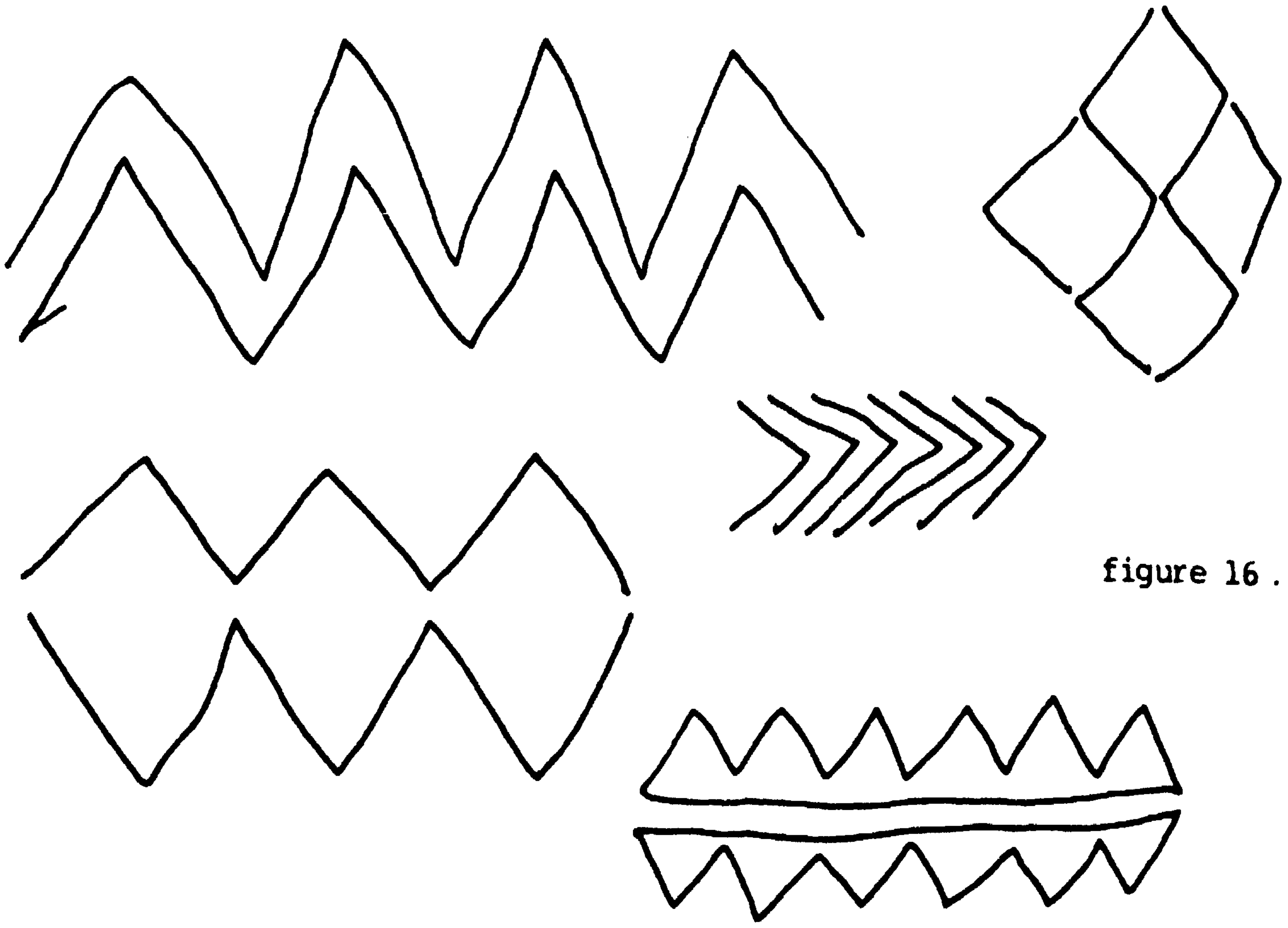


figure 16 .

2.4 INTERNAL REPRESENTATION

In the earlier stages of the development of the program, provision had been made for progressive access to the information stored in the data-structure, following the principal that it should not have to access more than it actually needed for the making of any particular decision. In practise, a great deal more was stored than was ever accessed. At the first level of detail the program made use of a quite coarse matrix representation, in each cell of which was stored an identifier for the figure which occupied it, and a number of codes which designated the various events which might have occurred in it: a line belonging to a closed form, a line belonging to an open form, a line junction, an unused space inside a closed figure, and so on. Obviously, it was not possible to record a great deal in this way, and data concerning the connectivity of the figure in particular required a second level of the structure.

This was an unpleasantly elaborate linked-list structure of an orthodox kind. By definition, the kind of drawing AARON makes is not merely a growing, but a continuously-changing, structure. What was a point on a line becomes a node when another line intersects it, and this change has to be recorded by updating the existing structure, which must now ideally show the four paths connecting this node to four adjacent nodes.

Both updating this structure and accessing the information contained within it proved to be quite tiresome, and the scheme was never general enough to admit of further development. As a result, it was used less and less, and decision -making has been based almost exclusively on the information contained in the matrix on the one hand, and in a third level of the structure, a simple property-list attaching to each figure, on the other. The most surprising thing about this simplistic and distinctly ad-hoc scheme is that it was actually quite adequate to the needs of the program.

<u>Explicit Data and Implicit Data</u>.

Human beings presumably get first-order information about a picture by looking at the picture. I have always found it quite frustrating that the program couldn't do the same thing: not because it made any difference to the program, but because it made it difficult for me to think about the kind of issues I believed to be significant. Part of the problem of using a linked-list structure to represent the connectivity of a figure, for example, derived from the fact that connectivity had to be <u>explicitly</u> recorded as it happened: it would have been much too difficult to traverse a structure of this kind post-hoc in order to <u>discover</u> facts about connectivity. If one could traverse the figure the way the eye does — loosely speaking! — it would not be necessary to give so much attention to recording explicitly all the data in the world without regard for whether it would ever be looked at again.

In short, the primary decision to be made was whether to accept the absolute non-similarity of picture and representation as given, devise a more sophisticated list-structure and drop the matrix representation altogether, or to drop the list-structure and develop the matrix representation to the point where it could be very easily traversed to generate information which was <u>implicit</u> within it. I opted for the latter. A description is included in Appendix 2, though at the time of writing (December '78) the implementation is not yet complete.

2.5 The <u>FUNCTION</u> Of <u>RANDOMNESS</u>.

This section does not deal with any single part of AARON: randomness is an active decision-making principle throughout the program, and I think it is important to say why that is the case. As a preface, it might be worth recording that beyond the limits of a mathematically sophisticated community most people evidently view randomness in a thoroughly absolutist fashion, and as the opposite to an equally absolute determinism. There is a firmly-held popular belief that a machine either does exactly what it has been programmed to do, or it acts "randomly". The fact that AARON produces non-random drawings, which its programmer has never seen, has given many people a good deal of trouble.

What I mean by "randomness" is the impossibility of predicting the outcome of a choice on the basis of previously-made choices. It follows, of course, that "randomness", in this sense, can never be absolute: if the domain of choice is the set of positive integers, one must be able to predict that the outcome will be a positive integer, not a cow or a color. In AARON the domain of choice is always a great deal more constrained than that, however. The corollary to the notion of randomness as a decision-making principle is the precise delineation of the choice space: in practice, the introduction into the program of a new decision characteristically involves the setting of rather wide limits, which are then gradually brought in until the range is quite small.

Randomness by Design and by Default.

AI researchers in more demonstrably goal-oriented fields of intellectual activity must obviously spend much time and effort in trying to bring to the surface performance rules which the expert must surely have, since he performs so well. I am not in a position to know to what extent "Let's try x" would constitute a powerful rule in other activities: I am convinced that it is a very powerful rule indeed in art-making, and more generally in what we call creative behavior, provided that "x" is a member of a rigorously constrained set.

A number of artists in this century — perhaps more in music than in the visual arts — have deliberately and consciously employed randomising procedures: tossing coins, rolling dice, disposing the parts of a sculpture by throwing them on the floor, and so on. But this simply derives a strategy from a principle, and examples of both can be found at almost any point in history. It is almost a truism in the trade that great colorists use dirty brushes. Leonardo recommended that the difficulty of starting a new painting on a clean panel — every painter knows how hard that first mark is to make — could be overcome by throwing a dirty sponge at it (note 8). But one suspects that Leonardo got to be pretty good with the sponge! An artist like Rubens would himself only paint the heads and hands in his figure compositions, leaving the clothing to one assistant, the landscape to another, and so on. All the assistants were highly-qualified artists in their own right, however. The process was not unlike the workings of a modern film crew: the delegation of responsibility reduces the director's direct control, and randomises the implementation of his intentions, while the expertise and commonly-held concerns of the crew provide the limits (note 9).

Randomising *in* the Program: Rules and Meta-rules.

For the hunan artist, then, randomising is not unconstrained, and therefore cannot be characterised by the rule "If you don't know what to do, do anything". Rather, one suspects the existence of a meta-rule which says, "precisely define a space within which any choice will do exactly as well as any other choice". In AARON, the implementation of the low-order rule has the following form:

```
(a and b and ...n)

    Then
    p% of the time do (x);
    q% of the time do (y);
    r% of the time do (z);
```

which fills out the description of the format discussed in PLANNING. The same frequency-controlled format is used within the action part of a production in determining specifications:

```
make a closed loop:
specification 1: number of sides
     50% of the time, 2 sides (simple loop)
     32% of the time, 3 sides
        . . .
specification 2: proportion
     50% of the time, between 1:4 and 1:6
     12% of the time, between 3:4 and 7:8
        . . .
specification 3.
```

AARON has only the simplest form of these meta-rules, which are used to determine the bounds of the choice space:

```
if(a) lowbound is La, highbound is Ha
if(b) lowbound is Lb, highbound is Hb
if(n) lowbound is Ln, highbound is Hn
specification taken randomly between
lowbound and higbound
```

where a,b,n are varying conditions in the state of the drawing. No consistent attempt has been made to develop more sophisticated meta-rules. In the final analysis, the existence of such rules implies a judgemental view of the task at hand, and they are consequently beyond the scope of a program like AARON, which is not a learning program and has no idea whether it is doing well or badly.

The Value of Randomness.

What does randomness do for the image-maker? Primarily, I believe its function is to produce proliferation of the decision space without requiring the artist to "invent" constantly. One result of that function is obviously the generation of a much greater nunber of discreet terminations than would otherwise be possible, and consequently the sense that the rule-set is a great deal more complex than is actually the case. A second result is that the artist faces himself constantly with unfamiliar situations rather than following the same path unendingly, and is obliged to pay more attention, to work harder to resolve unanticipated juxtapositions. It is a device for enforcing his own heightened participation in the generating process.

This last might seem less important in AARON: the program's attention is absolute, after all. But for the viewer the fact that AARON exercises the function is quite important. There is one level of our transactions with images on which we respond with some astuteness to what is actually there. The fact that AARON literally makes decisions every few microseconds — not binary decisions only, but also concerning quantitative specifications — shows clearly in the continuously changing direction of the line, in every nuance of shape, and succeeds in convincing the viewer that there is, indeed, an intelligent process at work behind the making of the drawings.

. . . .

3. CONCLUSIONS

AARON produces drawings of an evocative kind. It does so without user intervention; without recourse to user-provided data; and without the repertoire of transformational manipulations normal to "computer graphics". It remains now, if not to propose a coherent theory of image-making, at least to pull together those fragments of explanation already given into something resembling a plausible account of why AARON works.

This will be largely a matter of putting things in the right places.

Art-making and Image-making

First: no adequate justification has yet been given for the many references to art and art-making, as opposed to images and image-making, beyond saying that the first are a special case of the second. What makes them special?

Art is a bit like truth. Every culture has, and acts out, the conviction that truth and art exist, no two cultures will necessarily agree about what they are. There is no doubt, for example, that we use the word "art" to denote activities in other cultures quite unlike what our own artists do today, for the quite inadequate reason that those earlier acts have resulted in objects which we choose to regard as art objects. If it is surprisingly difficult to say what art is, it is not only because it is never the same for very long, but also because we evidently have no choice but to say what it is for us.

All the same, no justification is possible for making reference to it without attempting to say — once again! — what it is, and doing so in terms general enough to cover the greatest number of examples. Also, those terms should do something to account for the extraordinary persistence of the idea of art, which transcends all of its many examples.

Briefly, my view is that this persistence stems from a persistent and fundamental aspect of the mind itself. It would be stating the obvious here to propose that the mind may be regarded as a symbol processor of power and flexibility. I will propose, rather, to regard it as devoted primarily to establishing symbolic relationships: to attaching significance to events, and asserting that this stands for that. This is, surely, a large part of what we mean by understanding.

As for art: in its specifically cultural aspects art externalises specific assertions — the number three stands for the perfection of God, the racing car stands for the spirit of modern man, the swastika stands for the semi-mythical migrations of the Hopi people, or for a number of other things in a number of other cultures. But on a deeper level, art is an elaborate and sophisticated game played around the curious fact that within the mind things can stand for other things. it is almost always characterised by a deep preoccupation with the structures of standing-for-ness, and a fascination with the apparently endless diversity of which those structures are capable. What we see in the museums results from a complex interweaving of the highly individuated and the highly enculturated, and in consequence any single manifestation is bound firmly to the culture within which it was generated: or it is rehabilitated to serve new ends in a new culture. But ultimately, art itself, as opposed to its manifestations, is universal because it is a celebration of the human mind itself.

The Embeddedness of Knowledge

Second: much of what has come out of the writing of AARON has to be regarded simply as extensions to the body of knowledge which the program was intended to externalise. Writing it was not merely a demonstrative undertaking, and it is far from clear what has been raised to the surface and what newly discovered. I have regarded the program as an investigative tool, though for present purposes the distinction is not important.

It remains impossible to give an adequate account of this knowledge other than by reference to the program itself. There are several reasons for this. In the first place, this knowledge does not present itself initially as predominantly prescriptive. The first intuition of its existence comes in the form of an awareness that an issue — closure, repetition, spatial distribution — is significant: the program should be structured in terms of that issue, as well as in terms of all the other issues already contained. In this sense the left parts of the productions might eventually be taken together to represent the set of issues which AARON believes to be worth attending to in the making of an image. But this stage comes much later, and by this time an individual production functions as part of a fabric of issues, with so many threads tying it to so many knowledge sources, that a one-to-one account of how it achieves its effect is generally out of the question.

In fact, there is only a single example I can call to mind in which an effect can be ascribed with certainty to a single production: a particular class of junction in a meandering horizontal line will infallibly generate strong landscape reference, though only if the branching at the junction goes off on the lower side of the line (fig 17). This degree of specificity is certainly exceptional, but less powerful as an evocator rather than more so.

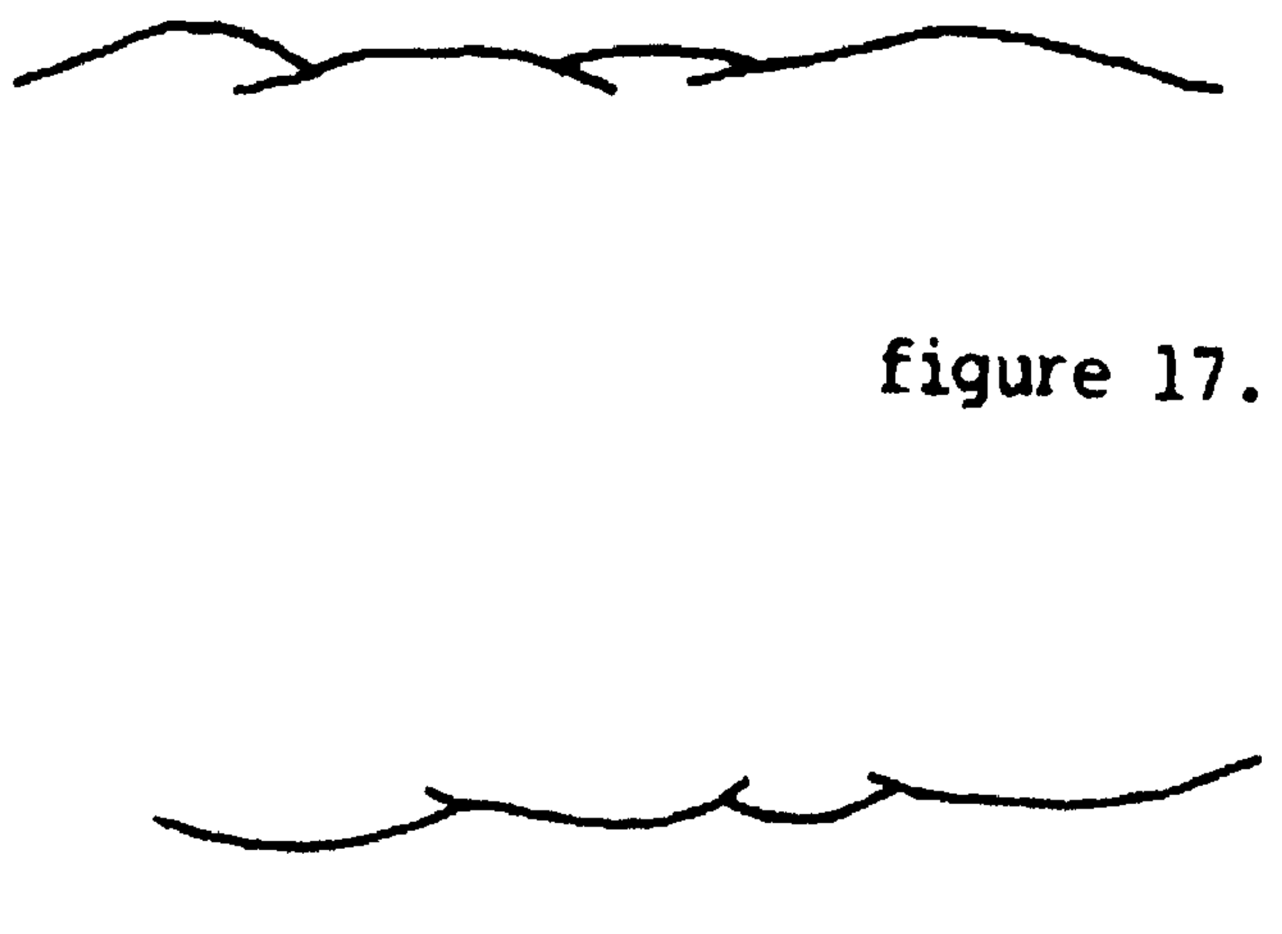


figure 17.

In general, this particular class of junction — it is more easily characterised visually than verbally — tends strongly to denote spatial overlap: but the specific effect is evidently quite context-dependant, and dependant also upon the precise configuration of the junction itself.

"Personality" as a Function of Complexity.

At the higher end of the scale of effects, the problem of saying what causes what becomes more difficult still. I have never been able to understand how there can be such general agreement about the "personality" which AARON'S drawings project, or why that "personality" appears to be like my own in a number of respects. Personality has never been an issue on the conscious level of writing code, and I know of nothing in the program to account for it. Tb put the problem another way, I would not know how to go about changing the program to project a different "personality".

I assume that the personality projected by an image is simply a part of a continuous spectrum of projection, not distinguishable in type from any other part. But I am forced now to the conclusion that these more elusive elements of evocation — personality is only one of them, presumably — are generated out of the complexity of the program as *a* whole, and not from the action of program pa7ts: that given an adequate level of complexity any program will develop a "personality". This "personality" may be more or less clear in individual cases, and may perhaps depend upon how many people have worked on the program — AARON is almost exclusively my own work — but it will in any case be a function of the program, and outside the willful control of the programmer. If this is the case it seems extremely unlikely that any complete causal account of the workings of a program would ever be possible.

The Continuousness of Image-making and and image reading

Third: I want to return to the question which lies at the root of this work. What constitutes a minimum condition under which a set of marks will function as an image?

The reader will have noted that much of what has been written here appears to bear as much upon the business of image-reading as it does upon image-making. There is no contradiction: the central issue being addressed is the image-mediated transaction itself, and image-making in particular has no meaningful, or examinable, existence outside of that transaction. Knowledge about image-making _is knowledge about image-reading: both rest upon the same cognitive processes. Thus the skilled artist does not need to enquire what the viewer sees in his work: the satisfaction of his own requirements guarantees it a reading in the world, and the explicit individual readings which it will have are irrelevant to him. The trainee artist, the student, on the other hand, frequently responds to his teacher's reading of his work by objecting, "You're not supposed to see it that way", evidently unaware that the reading does not yield to conscious control. Lack of skill in image-making more often than not involves a failure to discern the difference between what is in the image-maker's mind and what he has actually put on the canvas.

It is equally true, I believe, that image-reading has no meaningful existence outside the transactional context: not because the whole event is always present — it almost never is — but because every act of image-reading is initiated by the unspoken assertion "What I see is the result of a willful himan act". That is a part of what we mean by the word "image". However much we may amuse ourselves seeing dinosaurs in clouds or dragons in the fireplace, we have no difficulty in differentiating between marks and shapes made by man, and marks and shapes made by nature, and we do not hesitate to assign meaning in the one case where we deny it in the other: unless we belong to a culture with a more animistic attitude to nature than this one has.

In short, I believe that the first requirement of the condition in the question is the <u>undenied</u> assumption of human will (note 10).

The rest of the condition is given by the display of behavior which draws attention to a particular group of cognitive elements. In other words, evidence of cognitive <u>process</u> may be substituted for the <u>results</u> of an act of cognition. An actual desire to communicate — which may include the simple desire to record the appearance of the world — is not a necessary condition.

AARON's strength lies in the fact that it is designed to operate within, and feed into, the transactional context, not to reproduce the aesthetic qualities of existing art objects. It takes full advantage of the viewers' predispositions and does nothing to disabuse them: indeed, it might fairly be judged that some parts >f the program — the simulation of freehand dynamics, for example — are aimed primarily at <u>sustaining</u> an <u>illusion</u> (note 11). But the illusion can only be sustained fully by satisfying the conditions given above, and once that is accomplished the transactions which its drawings generate are real, not illusory. Like its human counterpart, AARON succeeds in delineating a meaning-space for the viewer, and as in any normal transaction not totally prescribed by prior cultural agreements, the viewer provides plausible meanings.

Standing-for-ness.

Fourthly: there is a multitude of ways in which something can stand for something else, and in adopting the general term "standing-for-ness" I intended for the moment to avoid the excess meanings which cling to words like "symbol", "referrent", "metaphor", "sign", and so on: words which abound in art theory and art history. An image, I have said, is something which stands for something else, and of course it is quite plain that I have been discussing only a very small subset of such things.

What are the defining characteristics of this subset?

Before attempting to answer that question, it should be noted that, while AARON's performance is based upon vision-specific cognitive modes (note 12), there are two closely related questions which cannot be asked about AARON at all.

Images of the World and its Objects.

The first of these has to do with the fact that in the real world people make images of <u>things</u>.

How do people decide what marks to make in relation to those things?

It is difficult to avoid the conclusion that image-making as a whole is vision-based, even though it bears directly on the issue of appearances only occasionally. It is my belief that even when an image is not purposively referential — as is the case with AARON — or when the artist seeks to refer to some element of experience which has no visual counterpart, it is his ability to echo the <u>structure</u> of visual experience which gives the image its plausibility (note 13).

<u>The Persistence</u> of <u>Motifs</u>

The second question has to do with the fact that actual image <u>elements</u>, motifs, have been used over and over again throughout human history, appearing in totally disconnected cultural settings, and bearing quite different

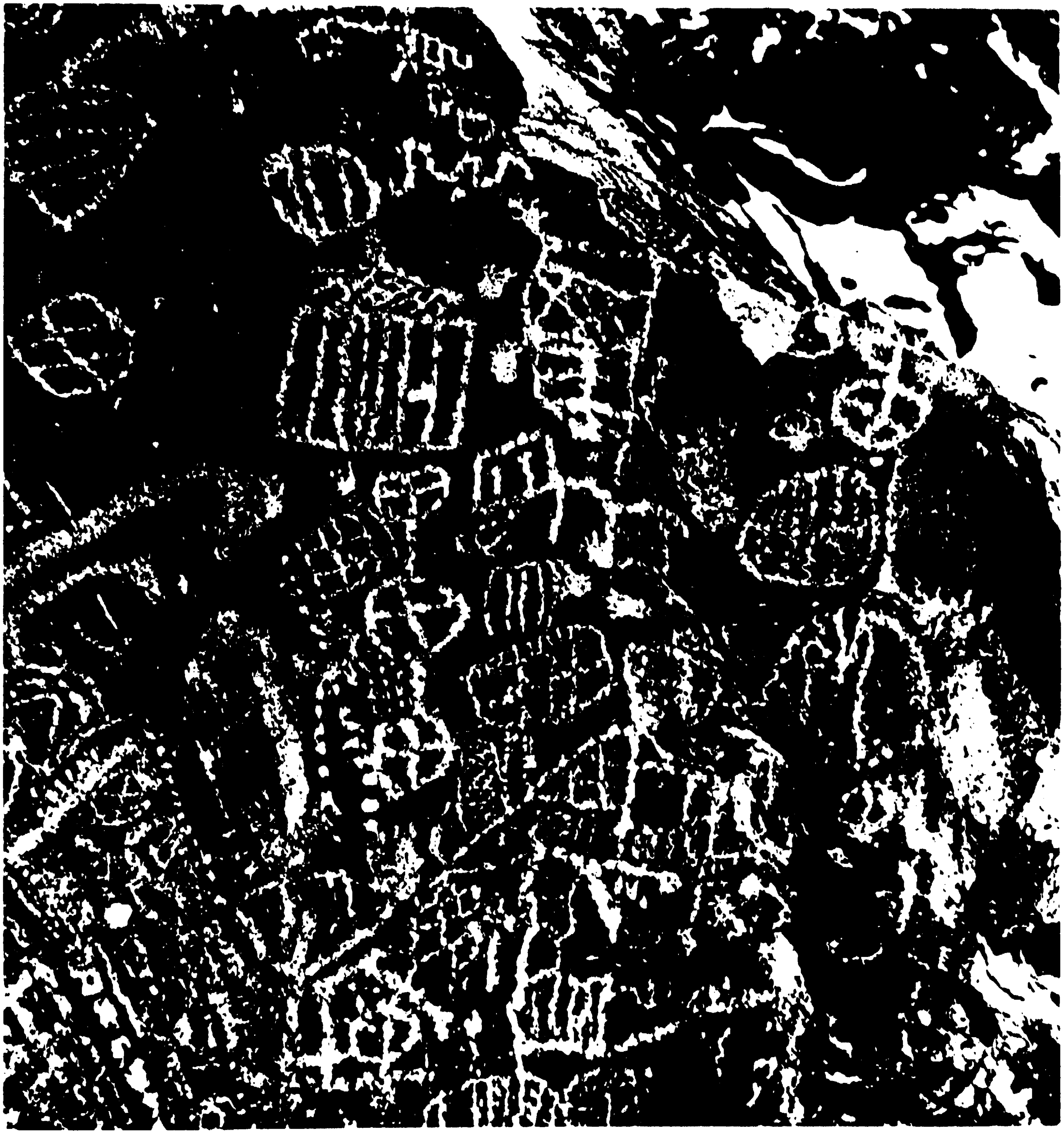


figure 18.

meanings as they do so. What is it that makes the zigzag, the cross, the swastika, squares, triangles, spirals, mandalas, parallel lines, combs (fig 18), ubiquitous, so desirable as imagistic raw material?

My own answer to this question is that the cognitive modes and their dependant behavioral protocols are absolutely ubiquitous, and that the recurring appearance of these motifs is hardly even surprising (note 14). In fact, we have only to start cataloguing the motifs to realize that most of them are simply formed through the combination of simple procedures. The swastika, for example, is both cross and zigzag, just as the mandala is cross and closed form, and the so-called diamond-backed rattlesnake motif of the Californian Indians is a symmetrically repeated zigzag (fig 19).

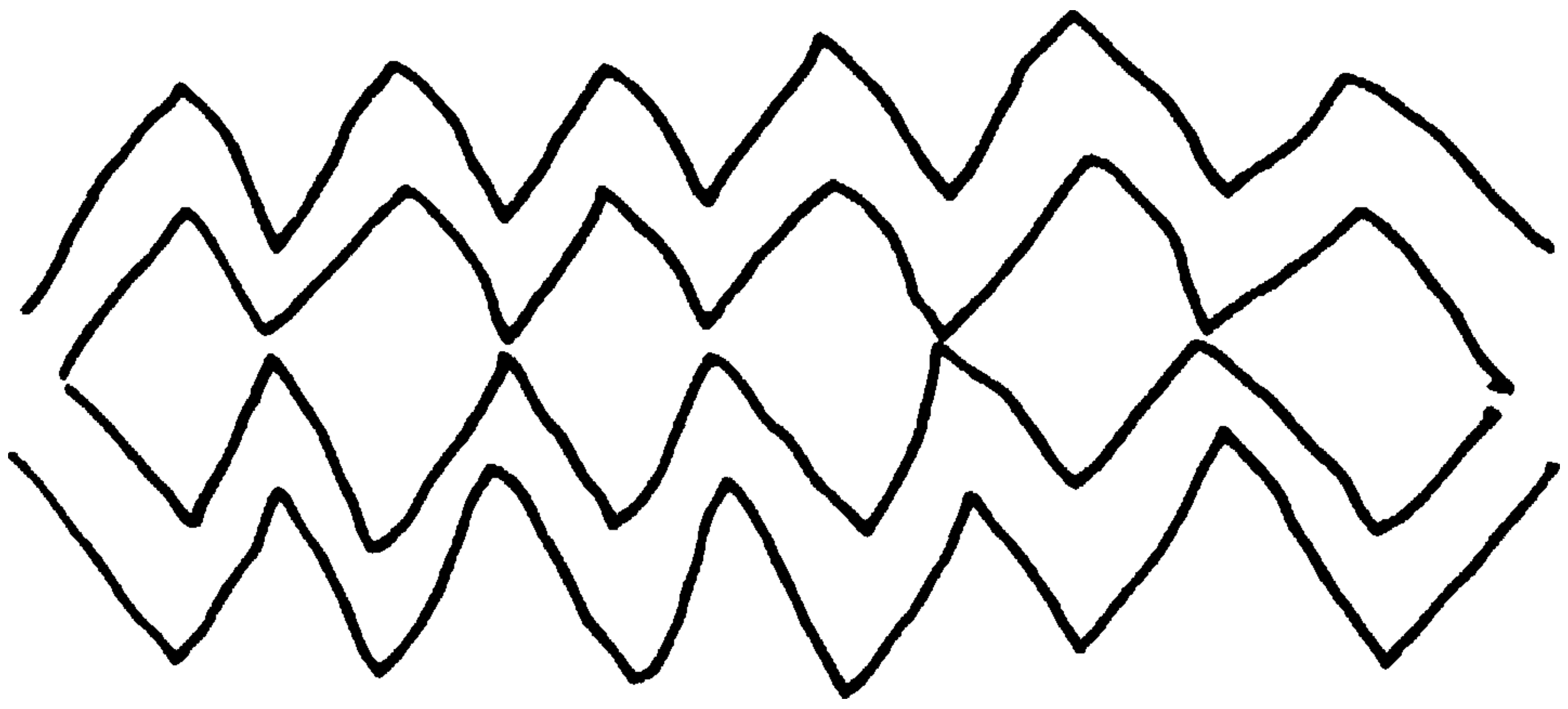


figure 19.

Taken together, these two questions point to the dualistic nature of image-making. If, as I believe to be the case, it can be shown that the representation of the world and its objects by means of images follows the same cognition-bound procedures as the simpler images I have been discussing, then it will be clear that the form of an image is a function both of what is presented to the eye and of the possession of appropriate modes.

Representation.

I said at the outset that my conclusions would bear upon the nature of visual representation, as distinct from what the AI/Cognitive Science community means by the word "representation". It is still the case that my specific concerns are with what people do when they make marks on flat surfaces to represent what they see, or think they see, in the world. All the same, some speculation is justified about possible correspondences between the two uses of the word.

It is important, for example, to note that the lines which the artist draws to represent the outline of an object do not actually correspond to its edges, in the sense that an edge-finding algorithm will replace an abrupt tonal discontinuity with a line. In fact, the edges of an object in the real world are almost never delineated by an unbroken string of abrupt tonal discontinuities. If the artist is unperturbed by the disappearance of the edge, it is likely to be because he isn't using that edge, rather than because he has some efficient algorithm for filling in the gaps. Similarly, most of the objects in the world are occluded by other objects, yet it would not normally occur to the artist that the shape of a face is the part left visible by an occluding hand (fig 20).

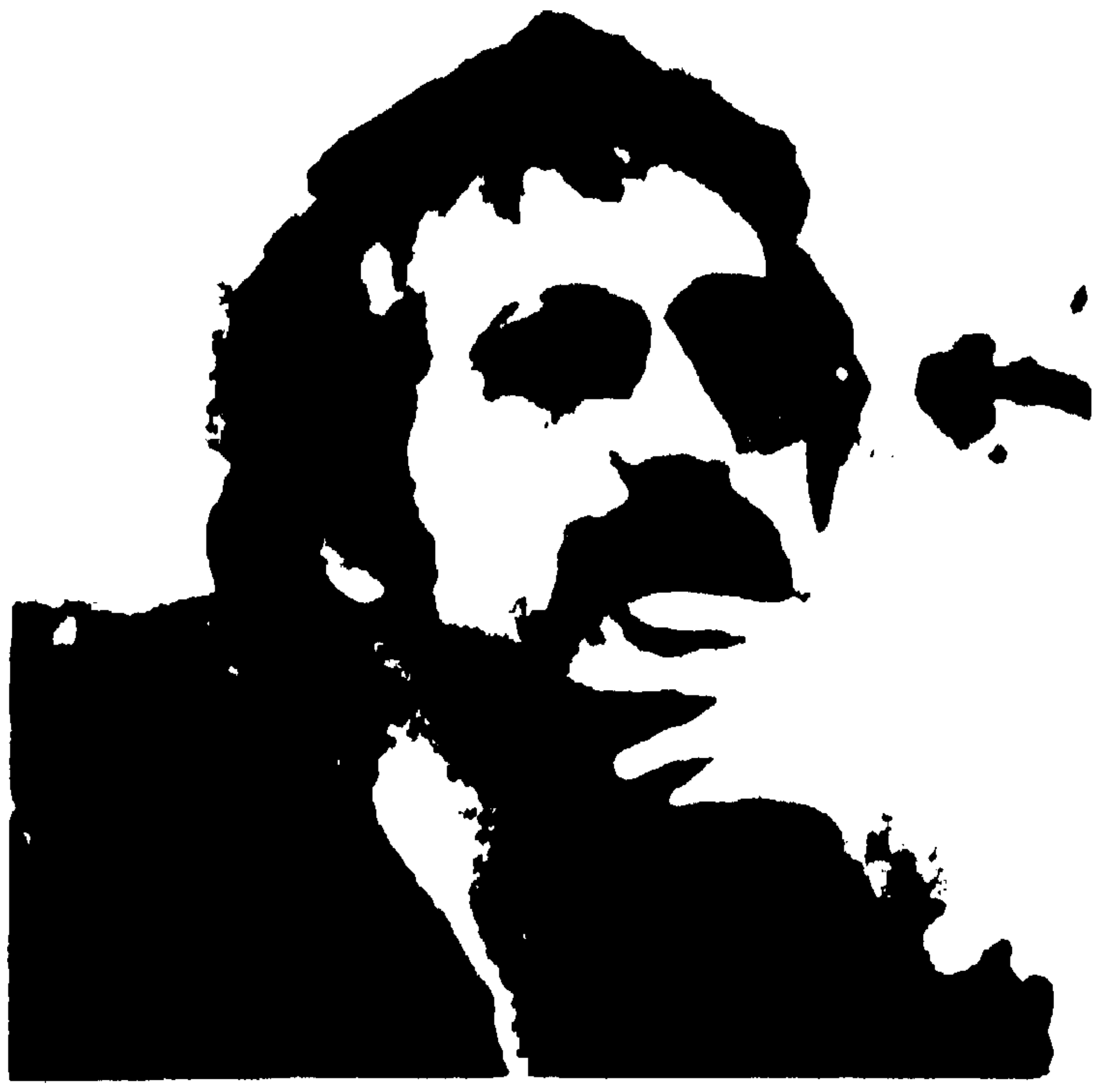


figure 20.

The face evidently exists for him as a cognitive unit, and will be recorded by means of whatever strategies are appropriate and available for the representation (note 15).

It is as true to your meaning of "representation" as to mine, not only that it rests upon the possession of appropriate and available strategies, but also that new strategies may be developed to fit particular concerns. Both are bound by entity-specific considerations, however: considerations, that is to say, which are independant of the particular event or object being represented and take their form from the underlying structures of the entity — the artist's cognitive modes on the one hand and the structural integrity of a computer program on the other.

What is a Representation "Like"?

It could not be seriously maintained that a computer program is "like" a human being in a general sense, and it should not be necessary to point out that a representation in my meaning of the word is not "like" the thing represented, other than in precisely defined senses of likeness. That may not be quite obvious, however, when we consider the idea that a portrait is "like" the sitter. Even though we may be careful enough to say that the portrait LOOKS like the sitter, or that a musical passage SOUNDS like the rustling of leaves, we tend to stop short of that level of detail at which it becomes clear that the appearance of a painted portrait and the appearance of a person actually have very little in common. A representation may be <u>about</u> appearance, but we never confuse the representation with the reality, no matter how "lifelike" it is. In fact, we might rather believe that all representations of a given class are more like each other than any of them is like the thing represented. Life follows its laws, representations follow theirs.

<u>What is an Image</u>?

The <u>purpose</u> of an act of representation is to draw attention to some particular aspect of the represented object, to differentiate that aspect from its context, not to reconstitute the object itself. To that degree we might regard a visual representation as constituting a partial theory of that object and its existence, just as we might regard a computer program as constituting a theory of the process it models. But neither the artist nor the program designer has any choice but to proceed in terms of the modes which are available or which they are capable of developing. In the case of the visual representation, the making of an image, I have tried to demonstrate the cognitive bases of those modes, and also, through my own program AARON, to demonstrate their raw power in the image-mediated transaction.

That, finally, defines my use of the word "image". An image is a reference to some aspect of the world which contains <u>within its own structure</u> and *in* <u>terms</u> of <u>its own structure</u> a reference to the act of cognition which generated it. It must say, not that the world is like this, but that it was <u>recognised</u> to have been like this by the image-maker, who leaves behind this record: not of the world, but of the act.

When the real turtle is not running, the program simulates its path, and calculates where it would have been in an error-free world after completing each command. In this case it substitutes a chord for the arc which the real turtle would have traced out. (The straight line segments which may just be visible in the illustrations here are due to the fact that they were photographed off the Tektronix 4014 display, not from an actual turtle drawing.)

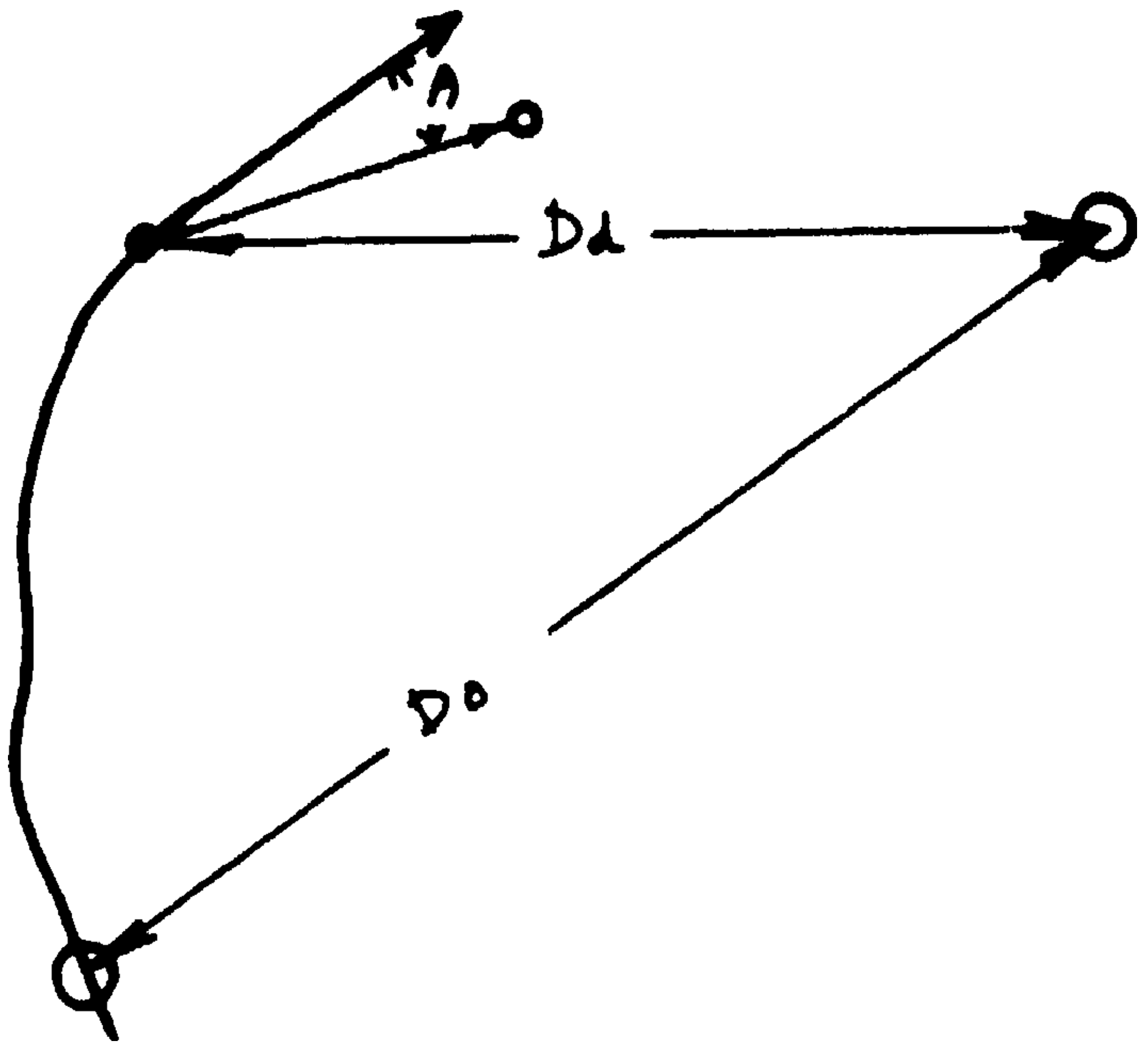
The Navigation System.

The navigation system is correct to about .2 inches: that is an absolute determined by the sonar operating frequency — about 40KHz — and does not change with the size of the drawing. Even with so coarse a resolution the feedback operation is efficient enough for the turtle to do everything on the floor that the program can do on the screen; indeed, if the turtle is picked up while it is drawing and put down in the wrong place it is able to find its way back to the right place and facing the correct direction.

<u>The Dynamics</u> of <u>Freehand Drawing</u>.

There are several complexities in this part of the program which are worth mentioning. One of them is that the program has to be able to accomplish dramatic shifts in scale in the drawing, to make small things which look like small examples of big things: smoothly-curved closed forms should not turn into polygons as they get smaller. This is required both on the issue of shifts in information density and also to maintain implied semantic relationships between forms.

A second complexity is that the movement of the line should convincingly reflect the dynamics of a freehand drawn line, and this should mean, roughly, that the "speed" of a line should be inversely related to the rate of change of curvature: the pen should be able to move further on a single command if it's path is not curving too radically. (The converse of this is that the amount of information needed to specify an arbitrary line should be a function of its rate of change of direction, with the straight line, specified by its two end points, as the limiting case.)



<u>Movement Scaling</u>.

Third, the pen should proceed more "carefully" when it is close to some final, critical position than when it has relatively far to go and plenty of time left to correct for carelessness. This, too, implies a scaling of movement in relation to the state of the local task. Finally, there is the practical problem that for any given number of cycles of a stepping pattern, the actual distance traversed by the pen will vary with the ratio of the turtle's two wheel speeds. Unfortunately, this relationship is not linear, and neither does it provide a useful simulation of freehand dynamics.

Briefly, the line-generating procedure concludes that, given the present position and direction of travel of the pen in relation to the current signpost and to the final destination, it will be appropriate to drive the two wheels at stepping rates r1 and £2, taking n steps on the faster of the two. In doing so""it takes account of all of the above considerations. *The* ratio determined for the two speeds is a function of two variables: the angle A between the current direction and the direction to the current signpost, and a scaling factor given by the remaining distance <u>Dd</u> to the final destination as a proportion of the original distance Do (fig 11). This speed ratio then becomes one of the two variables in a function which yields the number of steps to be taken — the distance to be travelled — by the fast wheel: the other variable being the relative size of the block of space allocated to the current figure.

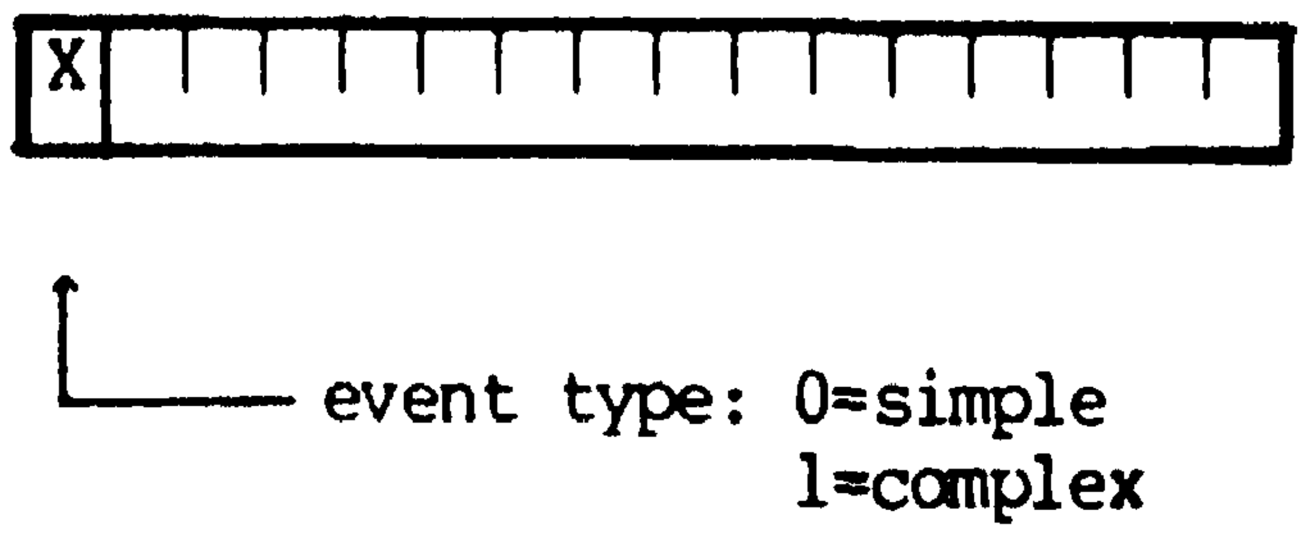
These functions have to be tuned with some care to be sure that each variable is correctly weighted, and to compensate for the turn-distance ratio of the turtle geometry itself. But none of this — or any other part of the program — involves any significant mathematical precision. There are only fifteen stepping rates available, synmetrically disposed between fast forward and fast reverse. The whole program, including extensive trigonometric operations, uses integer arithmetic — this for historical reasons as well as limitations of available hardware — and the geometry of the current turtle determines that it can only change direction in increments of about one sixth of a degree. (The turtle was not until recently interrupt-driven, and for design reasons this incremental direction-change factor was one degree in the earlier version.) Everything relies upon the feedback mode of operation to provide correction and to prevent error accumulation. The point is that a good car driver can drive a car with sloppy steering as well as a car with tight steering up to the point where feedback correction cannot be applied fast enough.

APPENDIX II — MATRIX REPRESENTATION.

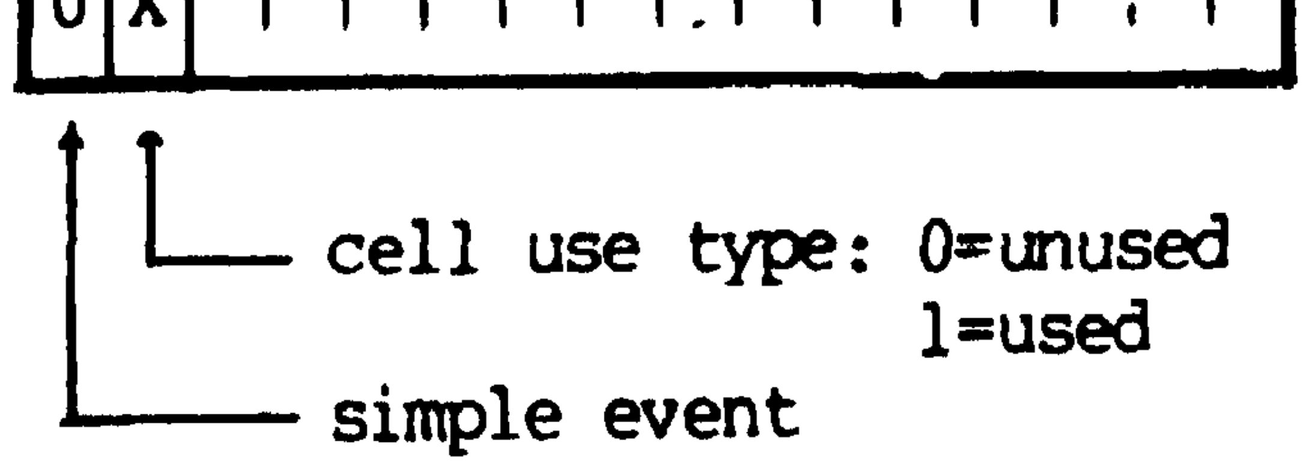
This description is given here primarily because it offers some insight into the kinds of considerations which the program believes to be important, and the way in which these considerations are accessed: not because there is anything particularly original from a data-structure point of view.

Much of the detail of the implementation is demanded by the word-length of the machine, and would go away in a larger machine. The intent is to make all the information relating to a particular part of the drawing effectively reside in a particular cell.

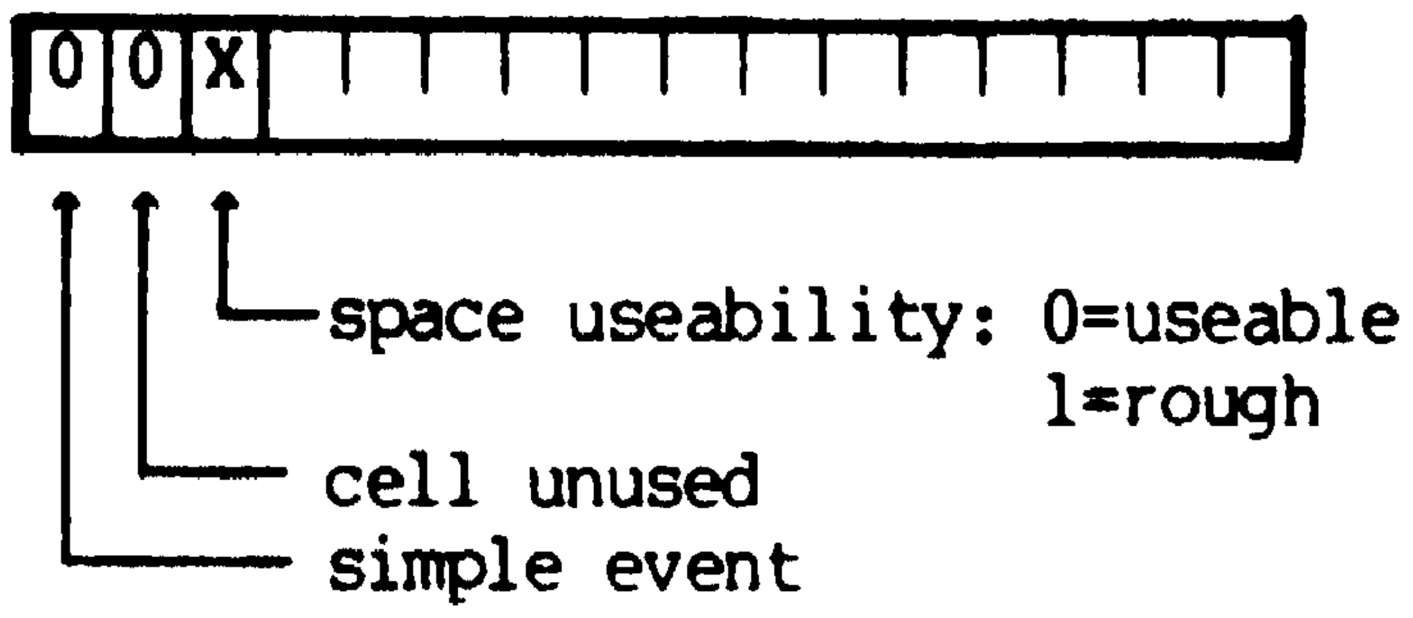
The program uses the single words representing matrix cells in different ways according to what is happening in the cells:-



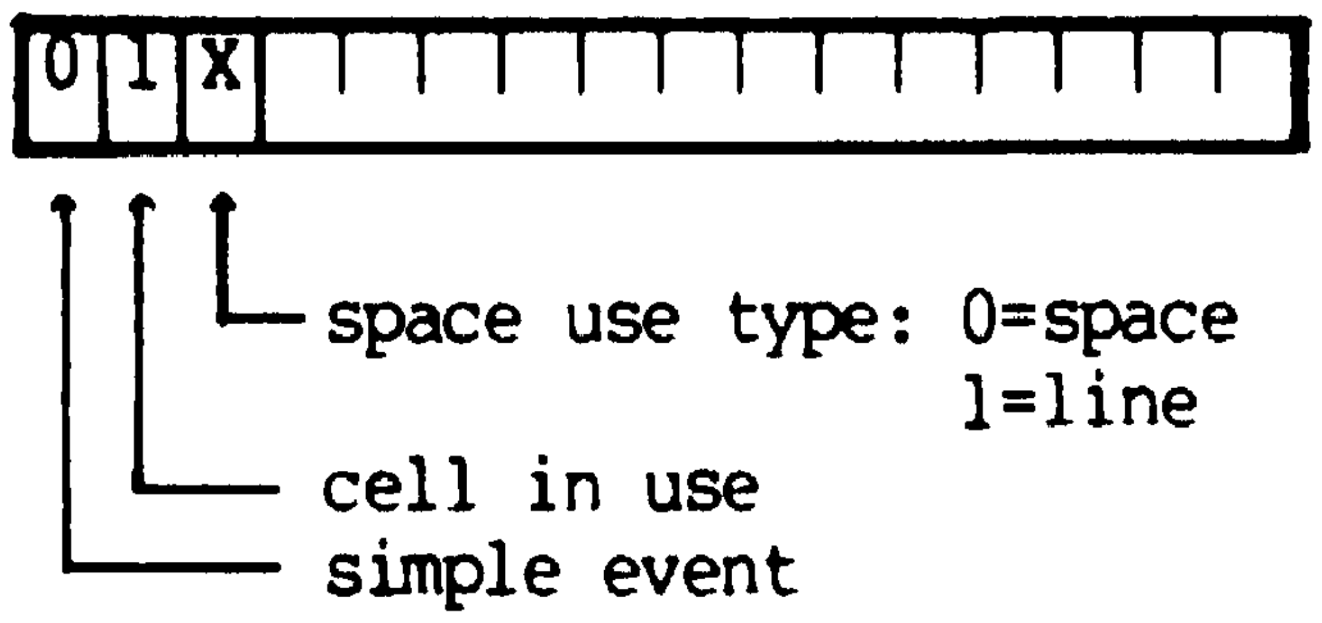
A "simple" event means, essentially, that all the data will be contained within this one word, although it will be seen that its simplicity relates to its use in a more meaningful sense:-



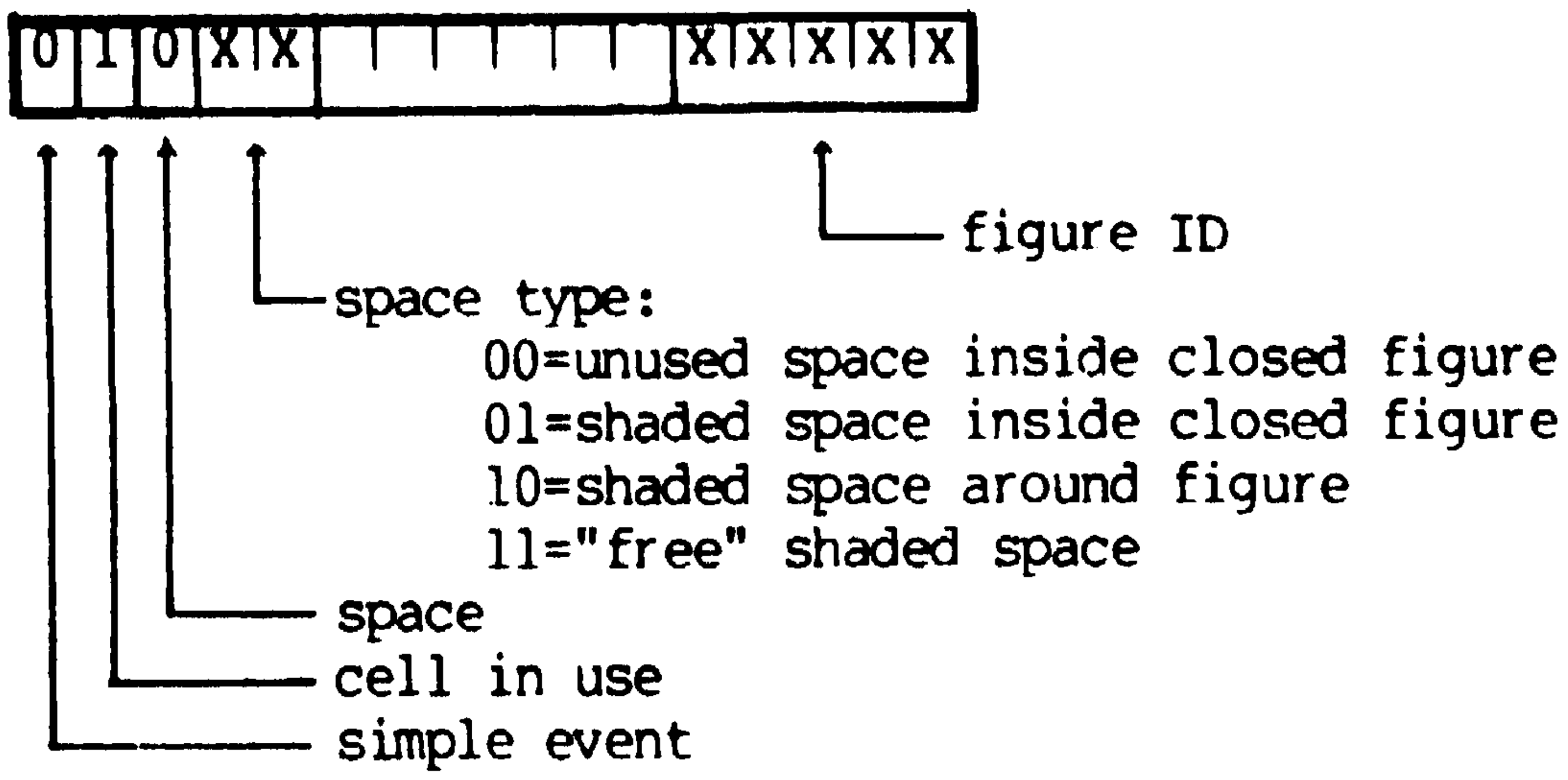
Before beginning work on the drawing, the program "roughens" the surface: that is, it declares some parts to be unuseable for the allocation of space to a new figure, although a developing figure may go into this "rough" space. This is done in order to maximise the rate of change of density across the image:-



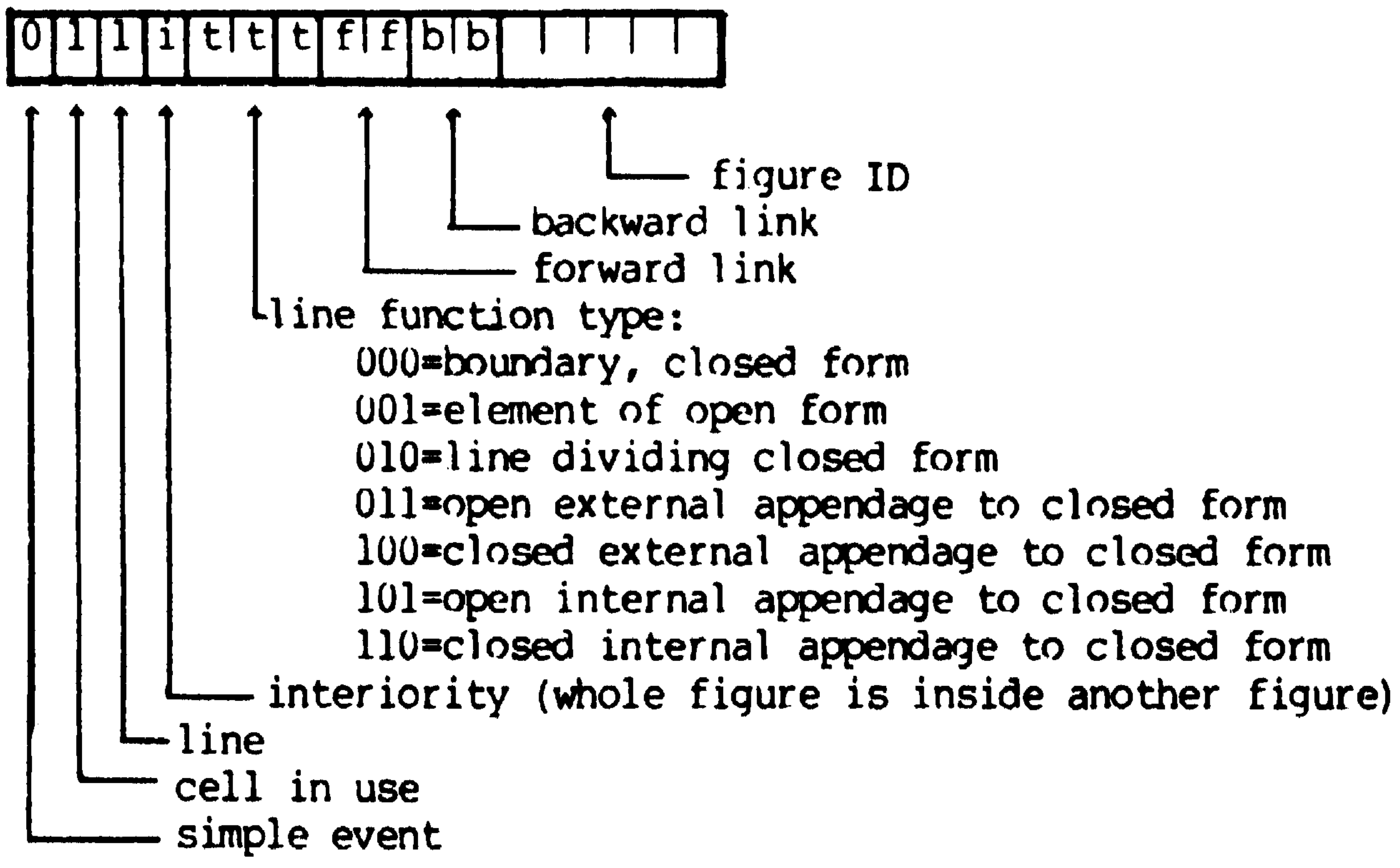
"Use" may involve either a line or some special spatial designation:=



In either case, the cell will now have a figure identifier associated with it. The new version of the program uses less figures than the earlier one, and develops them further: a maximum of 32 figures is permitted:-

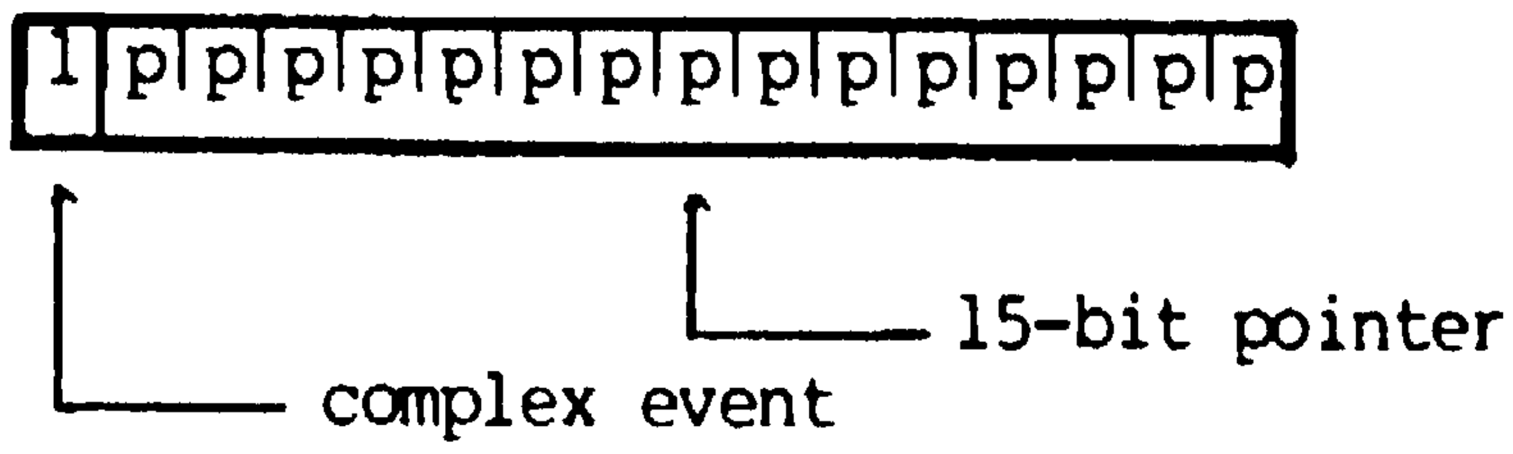

0 1 0 X X X X X X X
figure ID
space type:
00=unused space inside closed figure
01=shaded space inside closed figure
10=shaded space around figure
11="free" shaded space
space
cell in use
simple event

If the cell contains a line, then it can be dealt with as a simple event provided that it is not a line junction of a special kind. In this case the entry designates a line function type:-

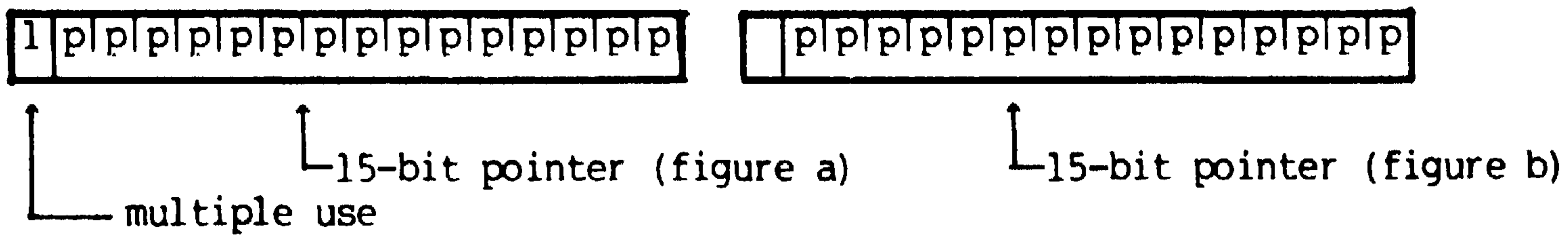

0 1 1 i t t t f f b b
figure ID
backward link
forward link
line function type:
000=boundary, closed form
001=element of open form
010=line dividing closed form
011=open external appendage to closed form
100=closed external appendage to closed form
101=open internal appendage to closed form
110=closed internal appendage to closed form
interiority (whole figure is inside another figure)
line
cell in use
simple event

V Linking.

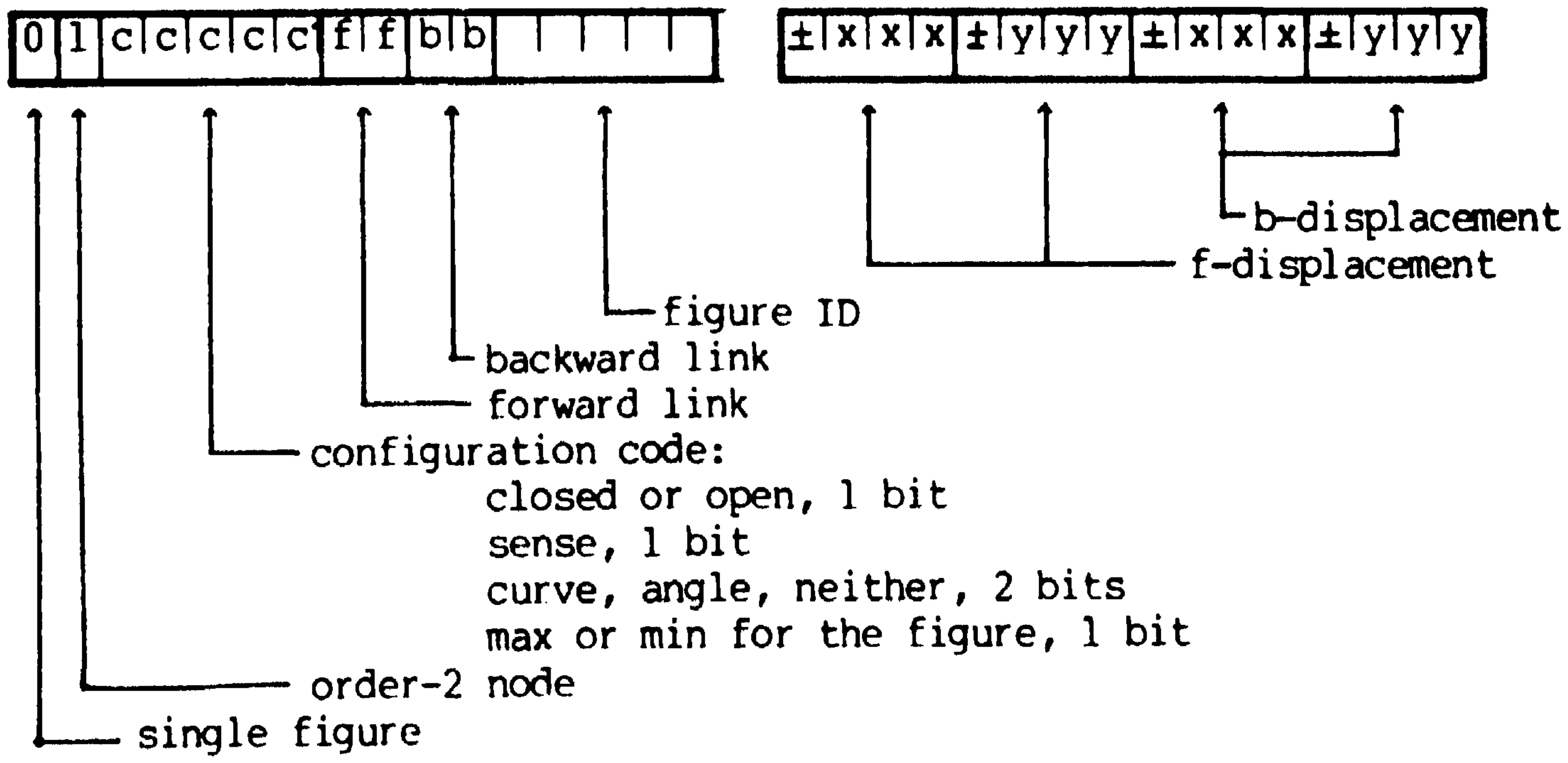
The forward and backward links are a very important device here. Lines are mapped onto the matrix as they are drawn, using an adapted form of Bresham's Algorithm to ensure that strings of cells never include corner-to-corner contiguity. This also means that for any given cell, the line it contains must have entered it from, and will subsequently leave it into, only one of four cells: thus the four-bit linking permits a complete traversal of any series of line segments not involving a complex event.



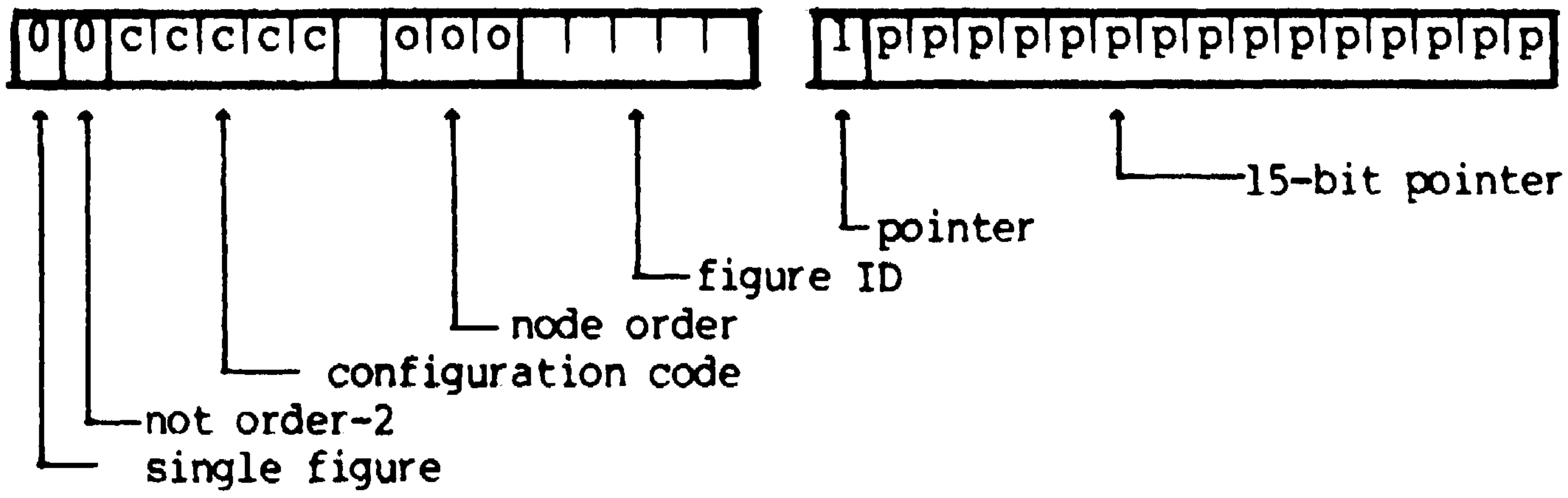
At this point, the single word is inadequate, and it is used as a pointer, words now being allocated in pairs from a freelist. Here again, one level down from the matrix, the words will be variously decoded. In particular, in the event that the cell is occupied by two figures, the two words are each used as pointers to new pairs of words, one for each figure:-

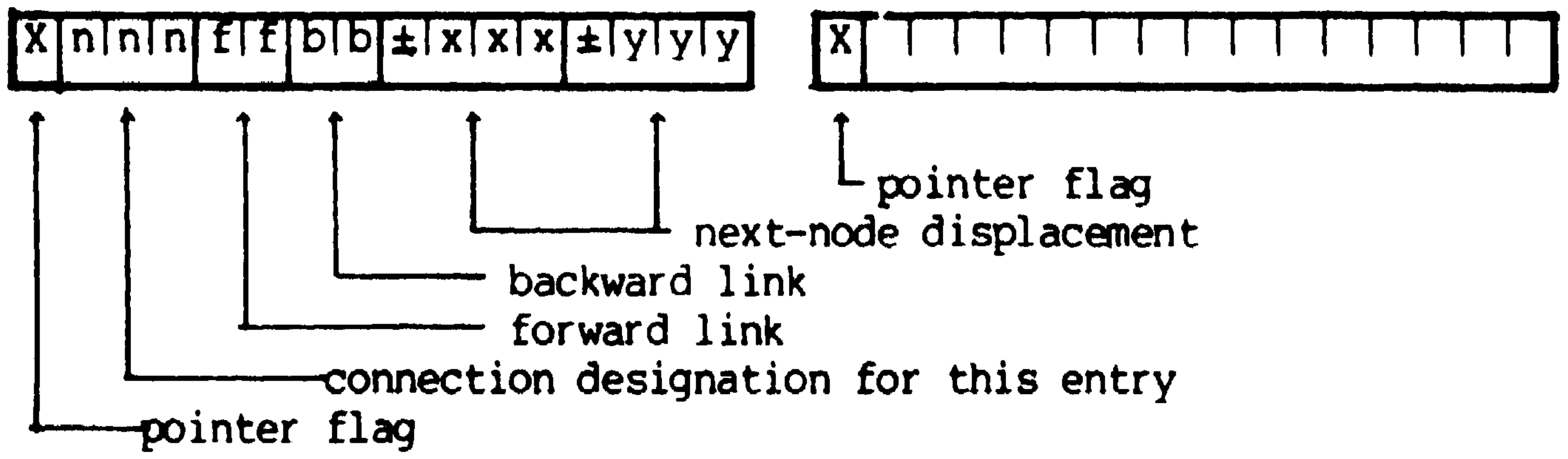

1 p p p p p p p p p p p p p p p
p p p p p p p p p p p p p p p
15-bit pointer (figure a)
15-bit pointer (figure b)
multiple use

A cell at this level may contain complex events from one or both of two classes: connective and configurational. Configurational events frequently involve order-2 nodes — nodes, that is, which fall on a continuous line — and include sharp angles, strong curvature, and so on. in practise, the program forces complex events so that they always occur within an 8-cell displacement in x and y from another cell, and the location of the next event can then be recorded rather cheaply:-



"Sense", here, means convex or concave if the line is the boundary of a closed figure, and up/right or down/left if it is not. If this is a figure node of any order other than two, one entry will be needed for each adjacent node:-





in addition to the displacements which chain this node to each of its connected nodes. This means that the traversal of the figure as it is represented by the matrix can continue from this point until the next node is reached.

Thus, the entire structure is contained essentially within the matrix, and the short lists which may be tacked onto any single cell serve merely to extend the effective capacity of that cell.

Ideally, this matrix should be as fine as possible: since the resolution of high-grade video is only 1024x1024, a matrix of this size would obviously constitute an extremely good representation. However, there are two considerations which make so fine a grain unnecessary. The first is that the program keeps a full list of all the actual coordinate pairs for each figure as it is drawing it, and can access it should some very precise intersection be required. The second is that the program is designed to simulate freehand drawing, not to do mechanical drawings, and once a figure is completed some approximation to it for purposes of avoidance or even intersection is unobjectionable. The maximum error induced by assuming a point to be at the center of a cell in a matrix of 90x160 will be about 7/8th of an inch in a sixteen-foot drawing: only three times the thickness of the line.

NOTES ON TOE TEXT

note 1. The word "representation" is used here in a more general sense than it now carries within the A.I. community: the problem of formulating an internal (machine) representation of some set of knowledge differs from the more general problem primarily in its technological aspects.

note 2. "The Art of Artificial Intelligence: 1, Teams and Case Studies of Knowledge Engineering," Ed Feigenbaum, Proceedings of IJCAI5, 1977; pp.1014-1029.

note 3. In the decade before I became involved in my present concerns my work was exhibited at all of the most serious international shows, and I represented my country at many of them, including the Venice Biennale: as well as in some fifty one-man shows in London, New York and other major cities.

note 4. Different from each other, loosely speaking, in the way one might expect a human artist's drawings to differ one from another over a short period of time.

note 5. Written in "C", under the UNIX operating sustem.

note 6. I am referring here to differentiations performed in relation to the image, not in relation to the real world, with which the program has had no visual contact.

note 7. The program does not attach semantic descriptors to the things it draws: the terms "penumbra", "boulder" and so on are my own descriptions, and are used here for the sake of simplicity.

note 8. Significantly, from the point of view of my argument here, the dirty marks were intended to "suggest" the elements of a composition.

note 9. The one unconstrained randomising agent in this scenario, the final cutting of the film by the producer rather than the director, has also demonstrated itself to be devastatingly non-creative.

note 10. "Undenied" is stressed here because there exists an odd case in which the will of the artist is to produce objects which demand the contemplation of their own qualities for their own sake — what they are rather than what they stand for — and which thus seek to deny the viewer his normal assumptions. To the degree that this aim can actually be achieved the resulting object could not properly be called an image, and I doubt whether aesthetic contemplation could properly be called reading. Thus much of XXth Century abstract art falls outside this discussion.

note 11. It is worth noting, though, that AARON did mechanical straight-line shading for about two years — it ran faster that way — and in that time only two people ever remarked on the inconsistency.

note 12. I will leave aside the interesting question of whether there are not more general underlying structures which are common to all physical experience. It is presumably no accident that terms 1 ike "repetition", "closure", and others I have used in relation to visual cognition are freely used in relation to music, for example.

note 13. The control of the rate of change of information density across the surface of the image, to which I referred earlier, is the most powerful example I know in this regard. The eye is capable of handling units as small as a speck of dust and as large as the sky, but the processes which drive the eye seem always to adjust some threshold to yield a preferred distribution spanning only a few octaves.

note 14. In fact, the more theatrical explanations which range from world-wide migrations to the influence of extra-terrestrial voyagers are not even necessary.

note 15. He is unlikely to treat the boundary between face and hand as part of the face, but as part of the hand, and may very well indicate the full boundary of the face as if he could actually see it.

FLATS, A MACHINE FOR NUMERICAL, SYMBOLIC AND ASSOCIATIVE COMPUTING

Eiichi Goto*'**, Tetsuo Ida , Kei Hiraki , Masayuki Suzuki and Nobuyuki Inada

* Institute of Physical and Chemical Research, 2-1, Hirosawa, Wako-shi,
Saitama 351 Japan
** Department of Information Science, Faculty of Science,
University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo 113 Japan

Abstract

Functional aspects of a machine called FLATS, presently (June 1979) in the design stage, are described. FLATS aims to efficiently run both numerical and algebraic programs. Overflow free and variable precision arithmetic, table look-up computation, and associative computation based on single-hit content addressed tables are introduced for advanced numerical, algebraic and symbolic computing. Hashing hardware, tag mechanism and hardware list processing are used to realize these features.

1. Introduction

Numerical computation and symbolic formula manipulation are the major practices in scientific computation. While numerical computation has a long history, computer-aided algebra or large scale formula manipulation has become practical only recently.

In this paper, we describe the architecture of FLATS, a machine which aims to efficiently run scientific computations. We consider Fortran (F) and Lisp (L) as major scientific programming languages for reasons:

(i) Scientific numerical programs are written mostly in Fortran.
(ii) Lisp is the major host language used for symbolic and algebraic systems such as REDUCE[1] and MACSYMA[2]. Moreover, Lisp has been used for writing a very large number of programs in the AI area.

Hence, high efficiency in running programs written in $^{f}F^{f}$ and ^{f}L* is considered a minimum requirement, which is symbolized in the first two letters of 'FLATS[1]. ^{f}A' in FLATS stands for associative capabilities; $^{?}T^{f}$ and $^{f}S^{f}$ stand for Table, Set and String which are the data types used in FLATS.

Many efforts have been made to increase the speed of computations by introducing parallelism[3'*4'5], in which we include the pipe line architecture as a specific implementation (of parallelism). Vectorized operations on floating and fixed point numbers of fixed bit length are recognized as a class computations suited for highly parallel machines. It should be noted, however, that highly parallel algorithms are not known for or not applicable to certain computations. For example, take the Euclidian gcd algorithm. The gcd of two integers TQ and rj, say, is obtained by successively computing the remainder sequence $^{r}0$» ^{r}l» $^{r}2$» $^{r}3$'*«(^{r}.i *^{s} ^{tne} remainder of r - divided by r) until it reaches the first zero remainder r_n 0 with r . giving the gcd.

Parallel schemes faster than the Euchidian algorithm are not known. Hence, for gcd, a machine equipped with a single sophisticated high speed remainder unit would do better than parallel machines with hundreds of slower units. List processing, arbitrary precision integer arithmeter, and variable precision floating point arithmetic are other examples of operations which are not suited for contemporary highly parallel machines. The objective of FLATS is to efficiently run those operations which are not suited for highly parallel machines.

FLATS is basically a single instruction stream machine and parallelism shall be pursued only to a limited extent such as advance control, parallel hashing and parallel interpolation but it shall not incorporate parallel processor arrays.

FLATS may hence be regarded complementary to the highly parallel machine approach.

The numerical support to algebraic manipulation systems is indispensable since the process of algebraic manipulation requires exact calculation of intermediate (and final) numerical coefficients as opposed to approximations done by conventional fixed precision floating arithmetic.

Furthermore, the results of the algebraic computation must often be given to numerical computaion systems.

Functional requirement for FLATS is therefore to incorporate the different characteristics of the two languages (F and L), e.g. compile-time vs. run-time data type checks, and static vs. dynamic storage allocation, into a single architecture, without impairing the efficiency of either one.

Besides the advanced numerical features, associative capabilities have been found to be very powerful[6] in efficiently carrying out basic opera tions in formula manipulation.

We first give in section 2 functional specifications of FLATS from users' point of view, and in section 3 consider the implementation from architectural point of view.

2. FLATS from users[1] point of view

2.1 Overflow free and variable precision computing

We observe that most contemporary computer systems neither provide adequate means for handling (1) overflows, (2) precision higher than builtin standard precisions, e.g. double or quadruple, nor (3) variable precision arithmetic. The design and implementations of some important class of algorithms[7,8,9] are hampered by the lack of the systematic support of these three features.

To remedy such defects, FLATS is provided with following means:

- Integers are treated with automatic multiple-precision.
- Exponents and mantissas of floating numbers are expressed in integers as stated above, hence practically no overflow would occur.
- A variable precision scheme[10] is adopted, in which precision is specified by a status word called an ASW (Arithmetic Status Word), which is part of the PSW (Program Status Word).

These features are made available to users in an extended FORTRAN called BFORT (Bignum Fortran).

2.2 Tabulative computing

Before high speed computers came into extensive use, look-up of mathematical tables was an essential process of numerical calculation. This fact is in quite contrast to 'on-demand' computing prevalent today. To pursue further the speed-up of the present numerical computation, table look-up is becoming again a practical method, owing to the recent advances in memory fabrication technology. In FLATS, we provide following means for tabulative computing[11];

(i) Tabulative computation of numerical functions with a floating number argument such as x^{-1}, $\sqrt{x}$, e^x, sin x, log x and atan x.

(ii) Associative tabulation for (sparse) integer and/or symbolic arguments.

Linear (or quadratic) interpolation shall be used for 24 bit (single length) mantissa and cubic interpolation for 48 bit (double length) mantissa. Namely $f(x)$ is to be computed by $f(x) = f(x_0) + f'(x_0)\Delta x + (1/2)f''(x_0)\Delta x^2 + (1/6)f'''(x_0)\Delta x^3$ where $x = x_0 + \Delta x$ and x_0 is the most significant 12 bit part of the 48 bit mantissa x and each term is assumed to be less than the previous term by a factor of 2^{-12}. The coefficients f, f', f'' and f''' are tabulated (in ROM or RAM). This cubic interpolation can be made within time of two stages of multiplications by introducing parallelism:
Stage 1. Read the coefficient table and the table of cubes of the 12 bit (in precision) Δx^3 and compute the 24 bit Δx^2 in parallel. (Table look up is assumed to be no slower than multiplication.)
Stage 2. Multiply and add in parallel.

For computing the 48 bit inverse $y = x^{-1}$, this scheme is faster than Newton iteration, which would consist of five stages of successive operations:
Stage 1. Read the 12 bit approximate inverse $y_0 = x^{-1}$ from table.
Stage 2,3. Two multiplications to get the 24 bit inverse $y_1 = (2-xy_0)y_0$.
Stage 4,5. Two multiplications to get the full 48 bit inverse $y = (2-xy_1)y_1$.

When a function, say g with (sparse) integer or symbolic arguments, is computed in tabulation mode, the value $v = g(i_1, \ldots, i_n)$ is tabulated in a content addressed table (CAT), using the tuple $(g, i_1, \ldots, i_n)$ as a search key in the CAT. In the subsequent computations of g with the same values of the arguments, the value v is fetched immediately from the CAT. Associative tabulation effectively and automatically speeds up recursions by avoiding repetitive computation. For example, the notoriously slow (exponential time) recursive computation of the Fibonacci numbers:

```
procedure Fib[n] = [n<1 -> 1;
                    T -> Fib[n-1] + Fib[n-2] ]
```

is turned into a fast algorithm (linear time for the first evaluation and O(1) time for the subsequent evaluations).

2.3 Associative capabilities and built-in data types

Starting from the pioneering associative language

LEAP[12], a variety of associative language and architectures for the associative processing have been proposed and implemented.

We consider an AMT (Address mapping table) and an h-op hash table (Fig. 1) as the two basic entities for building associative data structures[13]. We define an AMT as a mapping from bit patterns of fixed sizes (actually 32 bits/64 bits) into addresses (consecutive integers). An AMT may be regarded as a single-hit associative memory. A multi-hit associative scheme such as found in LEAP is to be dealt with by software, making use of arrays, linked-lists and AMT's. Table 1 gives the built-in data types in FLATS. The data types STR, INT, FLOAT, VECT (one dimensional array) are the same as in most other languages except that INT and FLOAT are of arbitrarily high precision as stated in 2.1. PAIR is the same as in Standard Lisp[14].

The data types prefixed by H denote those types obtained by hashing (h-op). h-type objects are guaranteed to be unique by virtue of hashing. The Lisp concept of 'intern' corresponds to that of h-op with the following differences; that h-op does not allow remob15 and that h-op can apply to all data types in FLATS, whereas Intern op in Lisp 1.5 is applicable only to atomic symbols, h-type objects must be read-only so as for the hashing search mechanism to work properly. Hence, HSTR objects are the same as atoms of Lisp; i.e. every different string of characters is represented as a unique pointer to a table called obllst in Lisp 1.5, or to name tables in assemblers and compilers. HINT and HPAIR objects are multi-precision integers and hashed lists, respectively. Equality check of two long integers of type HINT or two lists of type HPAIR is reduced to that of two pointers. HPAIR and HINT can be made by the shared linked list method as shown in Fig. 1.

Insertion, deletion and membership test of an AMT entry correspond to the same operations on a set. A 'set' created by h-op (represented by an AMT) is given[6,16] and shown in Fig, 1. In FLATS the hashed AMT (HAMT) is a synonym to 'set[1].

A CAT consists of an AMT and RAM (random access memory, i.e. hardware term for VECT sharing the same addresses)• A CAT can be hashed similarly to an AMT and the resultant data type is called HCAT.

These data types, especially the hashed types, have been found to be very useful in polynomial manipulation. For example, take a polynomial $P = 3AX^2 + 4\ BY^3$, where AX^2 and BY^3 are term identifiers; and 3 and 4 are coefficients. There are many (non-unique) ways to represent a term identifier in list forms owing to the commutative nature of multiplication. AX^2, for example, is represented in a list form ((A 1) (X 2)) and ((X 2) (A 1)) corresponding to AX^2 and X^2A respectively. The HCAT representation {A:1, X:2} is unique (AX^2 and X^2A are represented by a pointer to a unique HCAT structure). In this case, P is represented uniquely by nested HCAT's as
P = {{A:1,X:2}:3, {B:1,Y:3}:4}.

A FLATS FORTRAN program for multiplying polynomials in the HCAT normal form can be written in 9 lines:

```
      FUNCTION CATMULT (P,Q)
      C = MKCAT(0)
      SWEEP 1, P, TP=KEY, FP=VAL
      SWEEP 1, Q, TQ=KEY, FQ=VAL
      T = H(CATADD(TP, TQ))
    1 C(T) = C(T) + FP*FQ
      CATMULT = H(C)
      RETURN
      END
```

Here SWEEP is similar to DO in FORTRAN; SWEEP scans all the occurrences of keys in the specified CAT's P and Q. Function H corresponds to *h*-op above mentioned. This program would give the best known time complexity for multiplying multi-variate polynomials[6].

3. FLATS from architects' point of view

3.1 Tagged architecture

A practical solution for efficient detection of the built-in data types given in 2.3 is a tagged architecture [10,17] Table 2 gives tags to be used in FLATS.

Take numerical computation for example. Arithmetic operations on small numbers tagged as "sint" (short integer) or "afloat" (short floating number) are to be made with standard hardware at high speed, while those on tagged "big" numbers are to be trapped and handled by microprogramming. Since "sint" or "sfloat" numbers are likely to appear with high probability in most programs, the slow down factor due to the "big" numbers is very small.

Bit-loss, a common objection against tagblt(s), can be remedied by using a specific exponent value as a 'trapping tag*. In the case of 7-bit exponent -64 <. p < 63 FLOAT representation, p - -64, +63, +62 are used as the tags to denote data types other than "sfloat" with the mantissa part having different meanings given in table 2. The effective bit loss is only $\log_2(123/128)$ - -0.06 bits in this scheme.

3.2 Hashing Hardware

AMT's can be implemented either by using CAM (content addressable memory) chips or by hashing. In a previous paper[18], we presented a hardware hashing scheme which makes use of parallel read-out mechanism of memory. A parallel hashing hardward consists of multiple RAM banks, hash address sequence generators and a hashing control unit. A single comparator is attached to every RAM bank (cf. Fig. 2) in the parallel hashing hardware, whereas in CAM chip such as Intel 3104 a comparator is attached to every memory cell. Even at a chip level, a CAM requires several times more gates than a RAM chip. Hence total number of gates required for realization of CAM is greater by at least one order of magnitude than that for the hash table, with no significant speed gain over parallel hashing: The items in the hash table are searched in average 0(1) memory cycle (a quantity independent of the number of items entered in the hash table), in comparison with strict 0(1) used to search items entered in CAM. Since we are interested in the overall efficiency in our intended applications, it is more advantageous to invest the resource on enlargement of RAM memory than on still expensive CAM chips. Moreover, h-op of complex types such as AMT, CAT and VECT cannot be performed with CAM's.

Figure 2 shows the schematic diagram of the parallel bank hashing hardware of FLATS. Hashing is essentially the cycle (hash probe cycle) of (i) hash address generations, (ii) memory accesses, (iii) key comparisons and (iv) judgement of termination based on the comparisons. The hash tables, (i.e. AMT's and CAT's and an h-op table), are taken in main memory consisting of *J* banks. In a simple parallel hashing scheme with *J* banks, a single hash address generator and *J* comparators are used together with some auxiliary hash sequence control circuits. To handle key deletion, detectors of reserved words for denoting empty and deleted states are provided in each bank.

Assuming that the operations (i) - (iv) are performed in a single memory cycle, the performance of parallel hashing is given in number of memory cycles used until the completion of the probe cycles. Figure 3 shows that the average number of memory cycles in a successful search , (finding a key in the table) in the worst case[19] We presume the worst average performance in the performance evaluation when key deletion and insertion are repeated alternately. It shows that the basic hash operations can be performed in a time comparable to an indirect addressing operations, e.g. for *J* = 16 and the load factor of a hash table *a* * 0.9. Note that the hashing hardware handles a fixed length key at a time. For handling variable length strings, arbitrarily long integers and lists, a shared linked list above mentioned is used with the parallel hashing hardware.

3.3 <u>Hardware support for list processing</u>

The hardware mechanisms essential to the speed-up of Lisp are following;

(1) run time data type checking,
(2) hardware stack,
(3) high memory bandwidth
(4) segmented memory space with automatic boundary checking for arrays etc., and
(5) instructions for Lisp primitives, such as car, cdr, cons and atom.

Note that the tagged architecture given in 3.1 is a solution to point (1).

As for point (2), we incorporate in FLATS both general (global) register and stack frame machine architecture. The basic op codes consist of four 8 bit fields as (op, r_1 r_2, r_3), where r_1, r_2, r_3 are register addresses each specifying either one of 128 global registers or one of 128 registers on the current stack frame. While the global registers are used to hold global entities, the stack registers are used for local entries and for (recursive) subroutine linkage.

In contrast to the two register address operations, r_1: ■ op(r_1, r_2) notably used in IBM 360/370, three register address operations r_3: = op(r_1, r_2) are used in FLATS. The three register scheme would greatly reduce the number of register to register transfer operations. For example, while no register to register transfers are needed in the three register scheme for the following codes

```
    a:=add(x,y);
    b:=sub(x,y);
    c:=raul(x,y);
    d:-div(x,y);  ,
```

50% of the corresponding codes would be register to register transfers in the two register scheme as in

```
    a=x;  a;=add(a,y);
    b:=x;  b:-sub(b,y);
    c:-x;  c:-mul(cfy);
    d:-x;  d:-div(d,y);.
```

The unit of memory banking used in hashing is actually taken to be the bit width of a single list cell (64 bits). Expansion of memory to increase the degree of parallelism (i.e. to increase *J)* will speed up hashing operations as shown in Fig. 3; this does not imply the requirement of higher memory bandwidth since the comparators are provided in the memory units.

Need for segmentation is clear since the main memory is conceptually divided into several segments; AMT's, CAT's, arrays, list areas, program space and so on.

Fundamental Lisp primitives given above are executed in a single memory cycle. In fact, when run-time data type check is made by hardware, car and cdr instructions are equivalent to simple load operations. Complex Lisp primitives such as member and reverse are microprogrammed.

Furthermore, by virtue of the hashing hardware, operations on property lists and the oblist which rely on linear search in most Lisp systems are performed more efficiently in FLATS.

3.4 Hardware garbage collector

Dynamic storage allocation of variable length segments of memory in FLATS requires a compactifying garbage collection scheme. The most time-consuming process in compactifying garbage collection schemes[20,21] is pointer adjustment. FLATS is to incorporate the pointer adjustment hardware given in Fig. 5.

The pointer q in Q register is to be adjusted to be $q' = q - \Delta q$, where Δq is the total number of inactive cells in the memory block between base q_0 and q. A bit table of 2^n width is set up in the memory local to the hardware. The bit table is marked by a garbage collector: '1' in the m -th bit position in column ω corresponds to the inactive cell at $q_0 + \omega 2^n + m$ in the main memory. Therefore Δq is obtained by counting 1's between the base q_0 and q. The column $(q - q_0)/2^n$ of the offset table contains the cumulative bit count Σ. The adjustment can be done in a main memory cycle when fast memory for the offset table and the bit table is used.

4. Concluding Remarks

At a users[1] level, FLATS is a machine both for numerical processing and for formula manipulation. Instead of devising yet another language, we provide the facilities in the framework of existing languages, Fortran and Lisp (REDUCE is to be used for the user interface language). New compilers for Fortran and Lisp are under development, which incorporate the facilities for associative processing and overflow free and variable precision arithmetic, at the same time to accept programs written in Lisp and Fortran, which have been accumulated in libraries of scientific application packages. The underlying techniques include hardware hashing, tag mechanism, and hardware list processing (including hardware garbage collection). Concurrent garbage collection is not considered, since FLATS is not intended to be used for real time applications. FLAT is to be used as a back-end processor to a conventional computer system; hence the average performance, i.e. throughput, is a more important factor than responsiveness. For the same reason, hashing is best suited for the associative processing discussed.

The associative capabilities of FLATS would also be suited for some operations for data base management.

Implementation of unified memory management of secondary and main memories using virtual tapes[22] is under consideration.

Substantial amount of software for FLATS has already been written and has been running under a FLATS simulator. Architectural design of FLATS is an outcome of the experiences with the development of the software system HLISPf'[16] REDUCE is now running under HLISP system and are used for various applications. A version of BFORT mentioned before which translates the extended Fortran into HLISP has already been running.

References

[1 Hearn, A.C. REDUCE2 user's manual, Utah Computational Physics UCP-19 (March 1973)

[2 The MATHLAB Group, MACSYMA reference manual, Laboratory for computer science, MIT (1974)

[3 Watson, W.J. The TI ASC - A Highly Modular and Flexible Super Computer Architecture, AFIPS 1972 FJCC, AFIPS Press

[4 Russell, R.M. The CRAY-1 Computer System, CACM, Vol. 21, No. 1 (1978)

[5 Barnes, G.H., Brown, R.M., Kato, M., Kuck, D.J., Slotnick, D.L. and Stokes, R.A. The ILLIAC IV Computer, IEEE Trans. Computers, Vol. C-47, No. 8 (1968)

[6 Goto, E. and Kanada, Y. Hashing Lemmas on Time Complexity with Application to Formula Manipulation, Proc. ACM-SYMSAC, (1976)

[7 Takahasi, H. and Mori, M. "Double Exponential Formulas for Numerical Integration", Pub. Research Institute for Math. Science, Kyoto University, 9 (1974), 721-741

[8 Grau, A.A. "Algorithm 256, Modified Graeffe Method", Collected Algorithms from ACM, 256-P1-0

[9 Brent, R.P. "Fast Multiple-Precision Evaluation of Elementary Functions", JACM, Vol. 23, No. 2, (April 1976), 242-251

[10] Ida, T., and Goto, E. Overflow Free and Variable Precision Computing in FLATS, Journal of Information Processing Vol. 1, No. 3 (1978)

[11] Goto, E. and Terashima, M. MTAC - mathematical Tabulatlve Automatic Computing, Technical Report of Information Science, 77-03, Faculty of Science, University of Tokyo, (1977)

[12] Feldraan, J.A. and Rovner, P.D. An Algol-Based Associative Languages, CACM, Vol. 12, No. 8 (1969)

[13] Goto, E., Sassa, M. and kanada, Y. Algor-

ithms and Programming with CAM's (Content Addressable Associative Memories) Technical Report of Information Science Department, 78-04, Faculty of Science, University of Tokyo, (1978) submitted for publication

[14] Marti, J.B., Hearn, A.C., Griss, M.L., and Griss, C. Standard Lisp Report, UUCS-78-101, University of Utah, 1978

[15] McCarthy, J., et. al., Lisp 1.5 Programmers Manual, MIT Press, Cambridge, Mass., 1962

[16] Sassa, M. and Goto, E. A Hashing Method for Fast Set Operations, Information Processing Letters, Vol. 5, No. 2, (1976)

[17] Feustel, E.A. On the advantage of tagged architecture, IEEE Trans. Computers, Vol. C-21, No. 9 (1972)

[18] Ida, T. and Goto, E., Performance of a Parallel Hash Hardware with Key Deletion, Proc. IFIP Congress 77, North-Holland (1977)

[19] Ida, T. and Goto, E. Analysis of Parallel Hashing Algorithms with Key Deletion, Journal of Information Processing, Vol. 1, No. 1 (1978)

[20] Wegbreit, B. A Generalized Corapactifying Garbage Collector. The Computer Journal, Vol. 15, No. 3, (1972)

[21] Terashima, M. and Goto, E. Genetic Order and Compactifying Garbage Collectors, Information Processing Letters, Vol. 7, No. 1

[22] Sassa, M. and Goto, E. "V-tape", a Virtual Memory Oriented Data Type, and its Resource Requirements, Information Processing Letters, Vol. 6, No. 2, (1977)

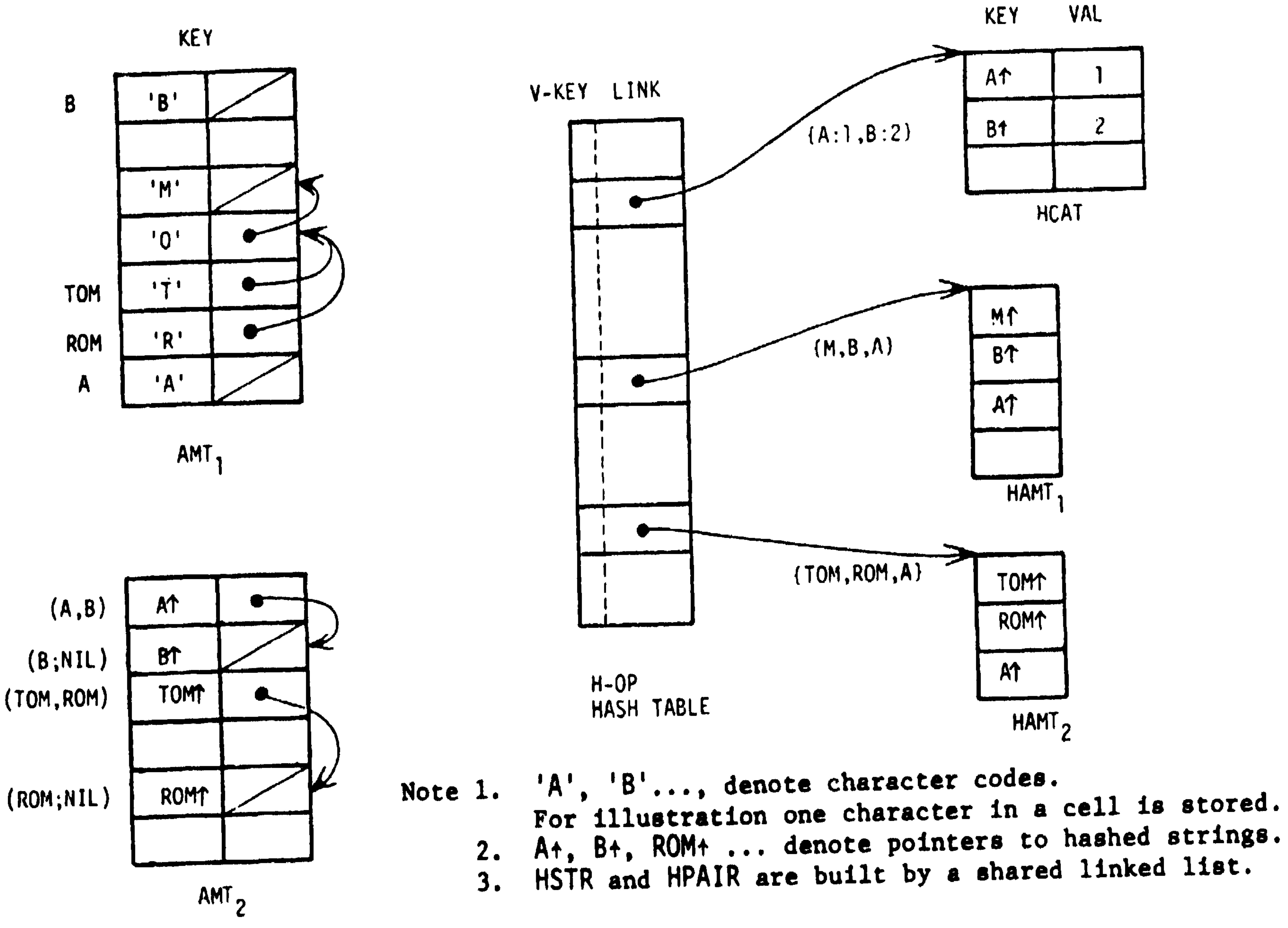
[23] This paper is the updated version of the paper by the same authors with same title given at ACM IEEE Symposium on Computer Architecture, May 1979, Philadelphia, PA.

main types	examples	hashed types	examples
ATOM			
STRing	3STOM	HSTR	3HTOM, 'TOM', TOM (three optional representations)
INT	-0137	HINT	-137
	+137		137
FLOAT	3.1415	HFLOAT	3.1415
PAIR		HPAIR	
DLIST	(A.B)	HDLIST	(A;B)
LIST	((A 1) (B 2))	HLIST	((A,1), (B,2))
FUNC	Fib		
VECT	[3 5 6 7]	HVEC	[3,5,6,7]
AMT	{A B}	HAMT	{A,B}
CAT	{A:1 B:2}	HCAT	{A:1,B:2}

Table 1. Built-in data types in FLATS

value of exponent, *p*	hardware data type	meaning of a mantissa
63	address	a pointer to some data structure
62	character	upto three characters
61 ~ -61	"sfloat", exponent of short floating number	mantissa of short floating number
-62	"sint", short integer	value of short integer
-63	software use	used as a trap code which may denote software (and microprogram) identified data types, such as "big" integer and "big" floating numbers
-64	reserved bit patterns	reserved values denoting nil, empty, deleted etc.

Table 2. Hardware tags used in FLATS



Note 1. 'A', 'B'..., denote character codes.
For illustration one character in a cell is stored.
2. A↑, B↑, ROM↑ ... denote pointers to hashed strings.
3. HSTR and HPAIR are built by a shared linked list.

Fig. 1 Realization of Built-in Data Types

Fig. 2 Parallel Hashing Hardware

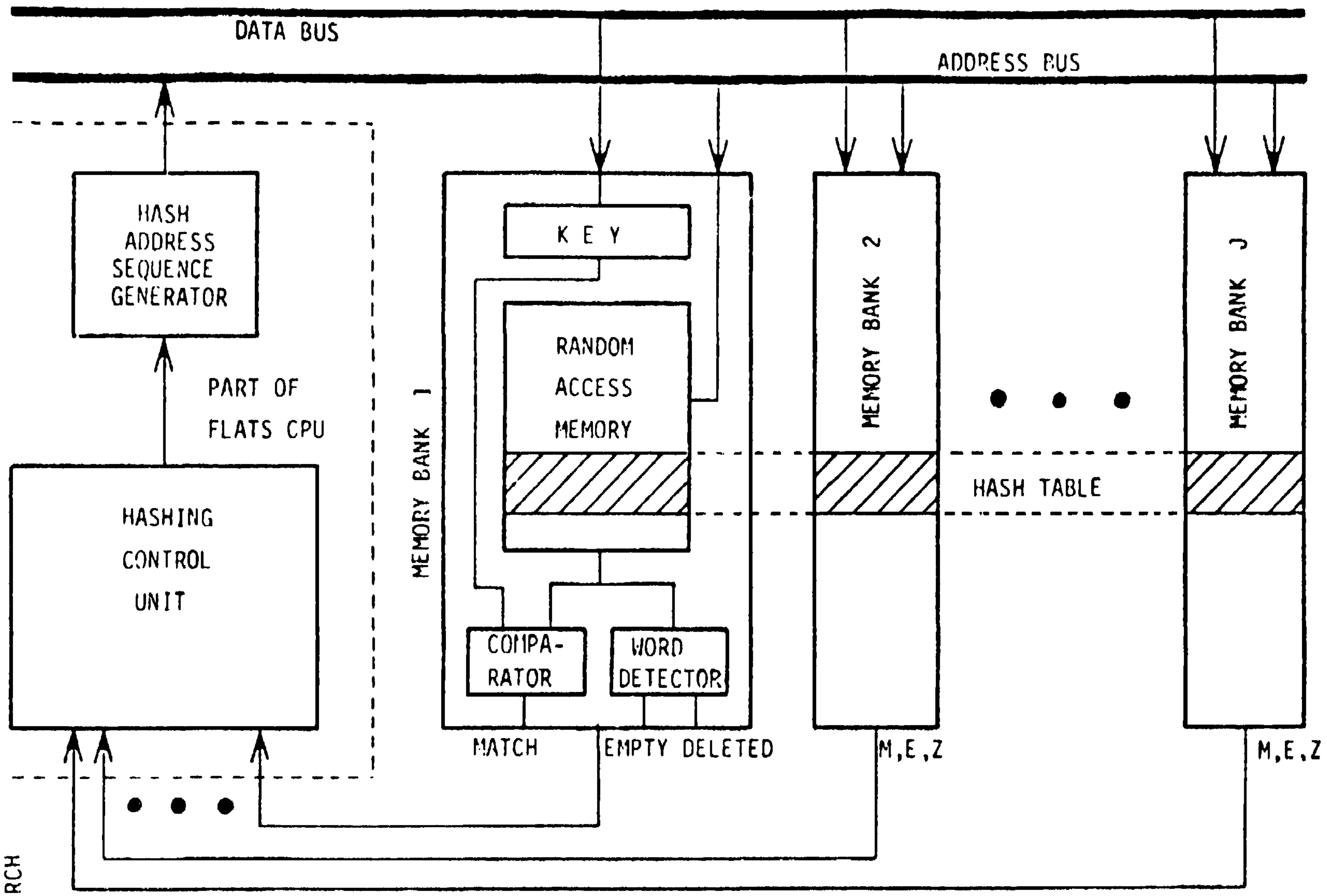
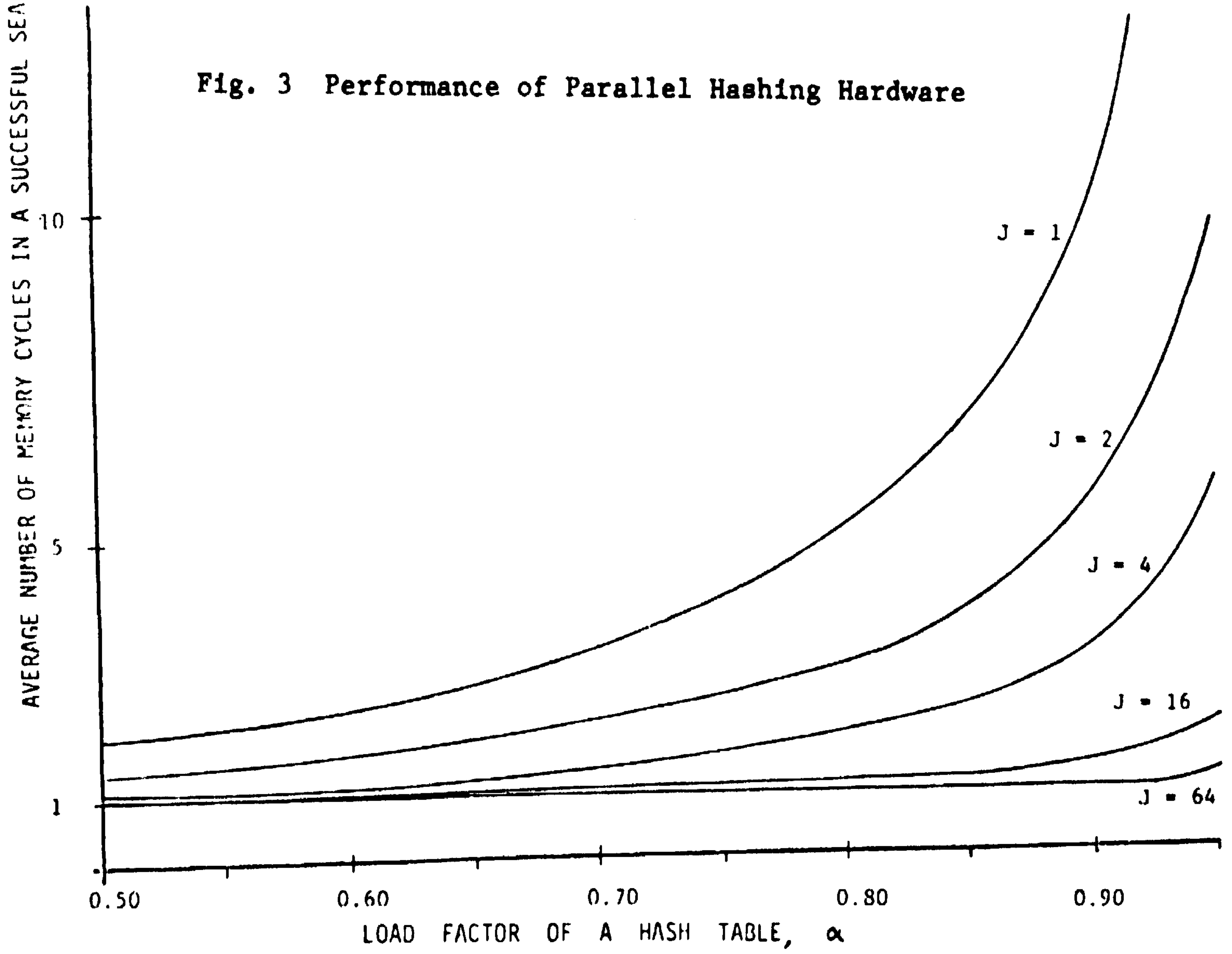


Fig. 3 Performance of Parallel Hashing Hardware



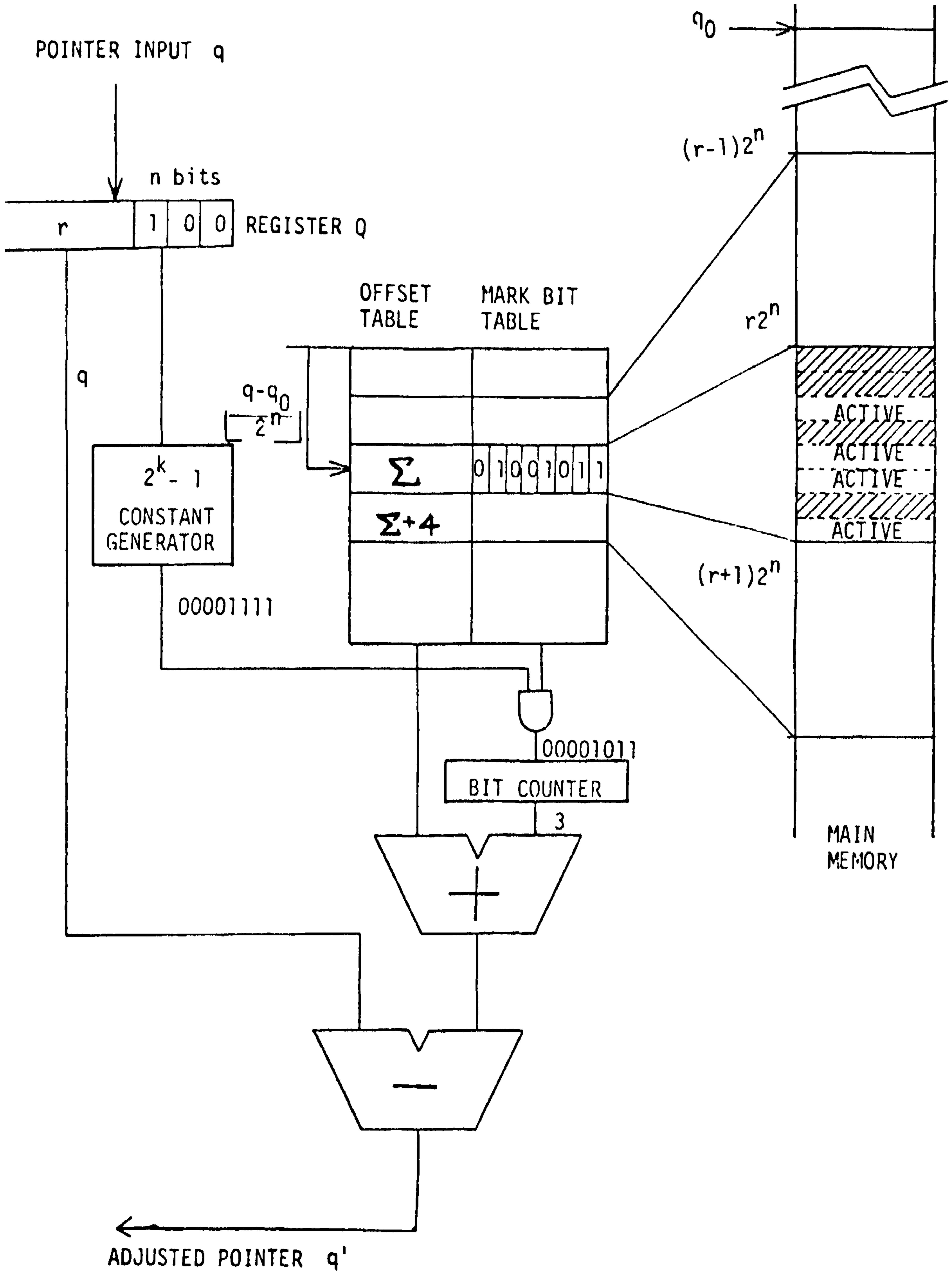


Fig. 4 Hardware for Pointer Adjustment

UTTERANCE AND OBJECTIVE:
ISSUES IN NATURAL LANGUAGE COMMUNICATION

Barbara J. Grosz
Artificial Intelligence Center
SRI International
Menlo Park, California 94025 USA

Communication in natural language requires a combination of language-specific and general common-sense reasoning capabilities, the ability to represent and reason about the beliefs, goals, and plans of multiple agents, and the recognition that utterances are multifaceted. This paper evaluates the capabilities of natural language processing systems against these requirements and identifies crucial areas for future research in language processing, common-sense reasoning, and their coordination.

1 INTRODUCTION

Two premises, reflected in the title, underlie the perspective from which I will consider research in natural language processing in this paper.[1] First, progress on building computer systems that process natural languages in any meaningful sense (i.e., systems that interact reasonably with people in natural language) requires considering language as part of a larger communicative situation. In this larger situation, the participants in a conversation and their states of mind are as important to the interpretation of an utterance as the linguistic expressions from which it is formed. A central concern when language is considered as communication is its function in building and using shared models of the world.2

Second, as the phrase "utterance and objective" suggests, regarding language as communication requires consideration of what is said (literally), what is intended, and the relationship between the two. Recently, the emphasis in research in natural language processing has begun to shift from an analysis of <u>utterances</u> as isolated linguistic phenomena to a consideration of how people use utterances to achieve certain <u>objectives</u>. But, in considering objectives, it is important not to ignore the utterances themselves. A consideration of a speaker's3 underlying goals and motivations is critical, but so is an analysis of the particular way in which that speaker expresses his thoughts. The choice of expression has implications for such things as what other entities may be discussed in the ensuing discourse, what the speaker's underlying beliefs (including his beliefs about the hearer) are, and what social relationship the speaker and hearer have. The reason for conjoining "utterance" and "objective" in the title of this paper is to emphasize the importance of considering both.

In the remainder of this paper I want to examine three consequences of these claims for the sorts of language processing theories we develop and the kinds of language processing systems we build.

* The emphasis in this paper will be on research concerned with the development of theoretical models of language use. Because of space limitations, I will not discuss a second major direction of current research in natural language processing, that concerned with the construction of natural language interfaces. The major difference between the two kinds of efforts is that research on interfaces has (to this point, though it need not) separated language processing from the rest of the system whereas one of the major concerns of research in the more theoretical direction is the interaction between language-specific and general knowledge and reasoning in the context of communication.

[2] Indeed, the notion of a shared model is inherent in the word "communicate," which is derived from the Latin <u>communicare</u>, "to make common".

I will use "speaker" and "hearer" to refer respectively to the producer of an utterance and the interpreter of that utterance. Although the particular communicative environment constrains the set of linguistic and nonlinguistic devices a speaker may use (Rubin, 1977), I will ignore the differences and concentrate on those problems that are common across environments.

* The similarity to <u>Word and Object</u> (Quine, 1960) is not entirely accidental. It is intended to highlight a major shift in the context in which questions about language and meaning should be considered. I believe the issues Quine raised can be addressed effectively nly in this larger context.

* Language processing requires a combination of language-specific mechanisms and general common-sense reasoning mechanisms* Specifying these mechanisms and their interactions constitutes a major research area.

* Because discourse involves multiple separate agents with differing conceptions of the world, language systems must be able to represent the beliefs and knowledge of multiple Individual agents. The reasoning procedures that operate on these representations must be able to handle such separate beliefs. Furthermore, they must be able to operate on Incomplete and sometimes inconsistent information.

* Utterances are multlfaceted; they must be viewed as having effects along multiple dimensions. As a result, common-sense reasoning (especially planning) procedures must be able to handle situations that involve actions having multiple effects.

2 MONKEYS, BANANAS, AND COMMUNICATION

To illustrate some of the current problems in natural language processing, I want to look at a variant of the "monkey and bananas" problem (McCarthy, 1968), the original version of which is substantially as follows: There is a monkey In a room that also contains a bunch of bananas hanging from the celling, out of reach of the monkey. There is also a box in one corner of the room. The monkey's problem is to figure out what sequence of actions will get him the bananas. For a while at least, this problem was a favorite test case for automatic problem solvers, and there are several descriptions of how it can be solved by machine (e.g., see NIlsson, 1971). The variation I want to discuss introduces a second monkey, the need for some communication to take place, and a change of scene to a tropical forest containing banana trees. To begin, I want to leave unspecified the relationship between the two monkeys and consider a short segment of hypothetical dialogue:

(1) monkeyl (looking longingly at bananas high
above him): I'm hungry.
(2) monkey2: There's a stick under the old
rubber tree.

If monkeyl Interprets monkey2's response as most Artificial Intelligence (AI) natural language processing systems would, he might respond with something like: "I can't eat a stick" or "I know, so what?" and, unless monkey2 helped him out, monkeyl would go hungry. Although there are a few systems now that might, with suitable tweaking, be able to get far enough for a response that indicates they have figured out that monkey2 intends for the stick to be used to knock down the bananas, there are no programs yet that would be able to understand most of the nuances of this response. For example, it implies not only that monkey2 has a plan for using the stick, but also that he expects monkeyl either to have a similar plan or to be able to figure one out once he has been told about the stick.

If we complicate the scenario just slightly, then our programs would all be stuck. In particular, suppose that the tree the stick is under is not a rubber tree, but rather a different sort of tree. Monkey2 might still use the phrase "the rubber tree", either by mistake or design, if he believes the phrase will suffice to enable monkeyl to identify the tree (cf. Donnellan, 1977). No current AI natural language processing system would be able to figure out where the stick is. Their responses, at best, would be like monkeyl saying, "Whaddayamean? There aren't any rubber trees in this forest." But referring expressions that do not accurately describe the entities they are intended to identify are typical of the sort of thing that occurs all the time in conversations between humans. The question is what It will take to get computer systems closer to being able to handle these sorts of phenomena.

In the remainder of this paper I want to look at some of the research Issues that need to be addressed to bring us closer to understanding why talking monkeys don't go hungry. I believe many critical language processing issues arise from our limited knowledge of how common-sense reasoning — which includes deduction, plausible reasoning, planning, and plan recognition — can be captured in a computational system. Consequently, research in natural language processing and common-sense reasoning must be tightly coordinated in the next few years. The root of the problem, I suspect, is the following discrepancy. Research in problem solving and deduction has focused almost exclusively on problems that a single agent could solve alone. The need for communication arises with those problems that require the resources of multiple agents, problems that a single agent has insufficient power to solve alone. As a result, language processing is typically an issue in

There Is an equal amount of sophisticated knowledge in monkeyl's using the statement "I am hungry" to ask for help from monkey2 in solving his problem.

just those contexts where the aid of another agent is essential. To obtain that aid, the first agent must take into account the knowledge, capabilities, and goals of the second. In exchange for not needing quite as much knowledge or capability in the problem domain, the agent must have additional communication capabilities. For such problems, the option of proceeding without considering the independence of other agents and the need to communicate with them is not feasible.

3 THE PROCESSES OF INTERPRETATION

To illustrate how language-specific processes combine with general cognitive processes (i.e., common-sense reasoning) in the interpretation of an utterance, I want to consider the monkeys and bananas example in more detail. It will be useful to view natural language interpretation as being divided into two major interacting levels. On the first, the linguistic analysis level, the form of an utterance is analyzed to determine its context-independent attributes. On the second, the assimilation level, common-sense reasoning processes operating in the context of the current cognitive state of the hearer —which includes such things as a focus of attention, a set of goals to be achieved or maintained and plans for achieving them, knowledge about the domain of discourse, knowledge about how language is used, and beliefs about the cognitive states of other agents, including other participants in the current conversation — use these attributes to update the cognitive state and to determine what response to the utterance is required, if any.

To illustrate these levels, let us return to the example and consider the interpretation of monkey2's response (2), "There's a stick under the old rubber tree," to monkey 1*8 indirect request (1).

3.1. Linguistic Analysis

At this level, the parsing process that assigns syntactic structure to the utterance also assigns attributes to the various syntactic subphrases in the utterance and to the utterance as a whole. Many of these attributes are of a semantic nature. For example, the attributes of the phrase "the old rubber tree" might include

The phrase is of syntactic class NP (noun phrase)

The phrase is definitely determined

The phrase describes a t such that TREE(t) and OLD(t), where OLD and TREE are predicate symbols[Q 0]

Attributes of utterance (2) as a whole include its syntactic structure and such properties as:^

The utterance presupposes that there exists a t such that OLD(t) and TREE(t), and that the description "OLD(t) & TREE(t)" should allow t to be determined uniquely in the current context.

The utterance asserts that there exists an s such that STICK(s).

The utterance asserts that UNDER(s t).

3.2. Assimilation

As attributes are extracted through the parsing process at the linguistic analysis level, common-sense reasoning processes begin to act on those attributes at the assimilation level. Two major activities are involved: completing the literal interpretation of an utterance in context, and drawing implications from that interpretation to discover the intended meaning.

I believe this option is becoming less feasible as well for problem solving and deduction components used for other purposes within AI. Situations in which multiple robots must cooperate introduce similar complexity even if the communication itself can be carried out in a formal language. Sacerdoti (1978) discusses the usefulness of research in natural language processing for the construction of distributed artificial intelligence systems. The issues being raised in this paper are central AI issues; they provide evidence of the interconnectedness of natural language processing research and other research in AI.

[7] This separation is useful for considering the kinds of processes involved in interpretation. The process of interpretation itself, of course, entails a great deal of interaction among the processes in the different levels. There are major research issues concerned with their coordination.

How much semantic specificity should be imposed at the linguistic level is an open research question. In particular, I have left open the question of what happens with the modifier "rubber"; suffice it to say, the question of how it modifies cannot be resolved solely at the linguistic level. Also, in a more complete analysis, the predicate OLD would indicate the set with respect to which age is evaluated.

[7] What an utterance presupposes and asserts are not necessarily components of the intended meaning, but the recognition of presuppositions and assertions is prerequisite to the assimilation level of processing.

For the example utterance (2), completing the literal interpretation in context involves the identification of the referent of the definite noun phrase, "the old rubber tree". The first attribute above indicates that a unique tree should be easily identified in context- Those objects currently in monkeyl's focus of attention are examined (perhaps requiring sophisticated common-sense reasoning) to determine whether there is such a tree among them* Assume that none is found* It may be that only two kinds of trees are present in this forest, and that one kind, say gumgum trees, resemble rubber trees, and that of all the trees near the two monkeys only one is a gumgum tree. Monekyl may tentatively assume that "rubber tree" matches "gumgum tree" closely enough to serve to identify this tree.

The sentence says there's a stick under the tree, so monkeyl might look under the tree and discover that, indeed, there is exactly one stick there. That stick must be the stick whose existence monkey2 was informing him of. The literal interpretation of the utterance is seen to be that the newly found stick is under the gumgum tree. °

Knowing that the sentence presupposed the existence of a rubber tree and asserted the existence of a stick, monkeyl may infer that monkey2 believes these presuppositions. Thus, monkeyl comes to believe several new things about monkey2's beliefs: in particular, that he believes these two entities exist, and that he thinks the gumgum tree is a rubber tree, or at least thinks that this description can be used to identify the tree. This fact may be important in further communications. Monkeyl may also infer that because monkey2 has just mentioned the stick and the tree, they are in his focus of attention and that he (monkey2), too, should pay special attention to these objects. The stick may be of particular importance because it was the subject of a there-insertion sentence (a syntactic position of prominence) and has been newly introduced into his focus of attention.

The second major process of assimilation is to use common-sense reasoning to determine how the utterance fits into the current set of plans and goals. In general, this is a highly complex process. For the particular example of interpreting utterance (2) in the context implied by utterance (1), monkeyl must determine what, "There's a stick under the rubber tree," has to do with his problem of getting something to eat. Briefly, he must see that the sentence emphasizes the stick and must know (or infer) that such sticks are often useful tools for getting things out of trees. He must infer that raonkey2 intends for him to use this stick in conjunction with a standard plan for knocking down things to acquire some bananas and accomplish his (implicitly stated) goal of not being hungry.

THE MULTIFACETED NATURE OF UTTERANCES

To determine what objective an utterance is intended to achieve requires determining where that utterance fits in the speaker's plans. That is, just as an agent may perform physical actions intended to alter the physical state of his environment, he may perform linguistic actions (utter sentences) intended to alter the cognitive state of the hearer. (Whatever effect an utterance eventually has on the outside world, its immediate effect is on the hearer's state.) But because a single utterance may be used to achieve multiple effects simultaneously, the problem is more complex than this analogy (or the preceding example) at first seems to suggest.[x]

The discussion so far has concentrated on a single dimension of effect: the use of an utterance to achieve what I will call a domain goal, that is, to convey information about the domain of discourse. In this section I want to discuss two other dimensions along which an utterance can have effects — the social and the discourse — and look at some of the problems in interpretation and generation that arise from the multifaceted nature of utterances.[3]

The social dimension includes those aspects of an utterance that concern the establishment and maintenance of interpersonal relationships. This dimension of utterance (1), "I'm hungry,"

[10] For more complex utterances, the process of completing the literal interpretation can involve determining the scopes of quantifiers and resolving various types of ambiguities.

[11] Cf. the analysis of a set of therapeutic interviews in Labov and Fanshel (1977).

[12] Physical actions may also have effects along •multiple dimensions although they are not usually thought of as doing so. For example, a ballet dancer in leaping (rather than walking slowly) not only changes position, but also conveys a particular state of mind for the character being portrayed and a particular level of capability.

[13] These dimensions parallel the three functions of language —ideational, interpersonal, and textual — in Halliday (1970), but the perspective I take on them is closer to that presented in Levy (1978).

is easily seen when it is compared with such choices as

(3) "How can I get some of those blasted bananas?"
(4) "Can you help me get some bananas down?"
(5) "Get me a banana."

Each of these achieves the same domain goal, informing monkey2 of monkeyl's desire to obtain some bananas, but utterance (1) does not convey the same familiarity as utterance (3) or the same level of frustration. Similarly, utterance (4) makes the same request as utterance (5) but does so indirectly and, therefore, can be used with social equals. The social dimension is present in every discourse[14] and prevails in some (e.g., Hobbs, 1979). It has been largely ignored in natural language processing research to date. The assumption has been that some sort of neutral stance is possible. But not choosing is choosing not to choose (cf. Goffman, 1978), and, although there are some serious philosophical issues raised by this dimension of utterances when considering communication between people and computers, I do not think we can continue to ignore it.

The discourse dimension includes those aspects of an utterance that derive from its participation in a coherent discourse — how the utterance relates to the utterances that preceded it and to what will follow. Typically the information a speaker wishes to convey requires several utterances. As a result the individual utterances must contain information that provides links to what went before and properly set the stage for what follows. Utterances that convey the same propositional content may differ widely in such things as the entities they indicate a speaker is focused on and hence may refer to later. As an extreme example, note that the propositional content of "Not every stick isn't under the rubber tree" is equivalent to that of utterance (2), but because it does not mention any individual stick, it does not allow whoever speaks next to make any reference to the stick that is under the gumgum tree.[15]

There are two characteristics of these dimensions and the multifaceted nature of utterances that introduce complications into natural language processing. First, as Halliday (1977) has pointed out, the units in which the information is conveyed along these other dimensions of meaning do not follow the constituent structure of sentences nearly so nicely as do the units conveying propositional content. In particular, the social implications of an utterance are typically reflected in choices scattered throughout it; for example, they are reflected in the choice of utterance type (a request vs. a command) and in the choice of lexical items.

Second, an utterance may relate to plans and goals along any number of these dimensions. It may be a comment on the preceding utterance itself, its social implications (or both, as is usually the case with "I shouldn't have said that"), or on some part of the domain content of the utterance. It is not simply a matter of determining where an utterance fits into a speaker's plan, but of determining which plan or plans —domain, social, or communicative — the utterance fits into. A one dimensional analysis of an utterance is insufficient to capture the different effects (cf. Goffman, 1978).

The multifaceted nature of utterances poses problems for language generation as well. A speaker typically must coordinate goals along each of these dimensions. He must design an utterance that conveys information linking it to the preceding discourse and maintains the social relationship he has with the hearer(s) (or establishes one) as well as conveying domain-specific information.[16] The speaker's task is further complicated because he has only incomplete knowledge of the intended hearer's goals, plans, and beliefs.

5 STATE OF THE ART

I will use our work in natural language processing at SRI International (Robinson, 1978; Walker, 1978) as an exemplar for discussing the current state of research in this area, both because I am most familiar with it and because I think the framework it provides is a useful one for seeing not only where the field stands, but also where the next several years effort might best be expended. The system coordinates multiple sources of language-specific knowledge and combines them with certain general knowledge and common-sense reasoning strategies in

[14] Pittenger et al. (1960) point out that "no matter what else human beings may be communicating about, or may think they are communicating about, they are always communicating about themselves, about one another, and about the immediate context of the communication."

[15] This example is based on one suggested by Barbara Partee for the Sloan Workshop at the University of Massachusetts, December, 1978. A discussion of her example is included in Grosz and Hendrix, 1978.

[16] Levy (1978) discusses how the multiple levels along which a speaker plans are reflected in what he says and the structure of his discourse.

arriving at a literal interpretation of an utterance in the context of an ongoing task-oriented dialogue* A major feature of the system is the tight coupling of syntactic form and semantic interpretation* In the interpretation of an utterance, it associates collections of attributes with each phrase. For example, noun phrases are annotated with values for the attribute 'definiteness', a property that is relevant for drawing inferences about focusing (Grosz, 1977a, 1977b, 1978) and about presuppositions of existence and mutual knowledge (Clark and Marshall, 1978).

Interpretation is performed in multiple stages under control of an executive and in accordance with the specifications of a language definition that coordinates multiple "knowledge sources" for interpreting each phrase. Two sorts of processes take part in the linguistic level of analysis* First, there are processes that interpret the input "bottom up" (i.e., words -> phrases -> larger phrases -> sentences). In the analysis of utterance (2), these processes would provide attributes specifying that the phrase "a stick" is indefinite and in the subject position of a there-initial sentence and the phrase "the rubber tree" is definite and presupposes the existence of a uniquely identifiable entity. Second, there are processes that refine the interpretation of a phrase in the context of the larger phrases that contain it, doing such things as establishing a relationship between syntactic units and <u>descriptions</u> of (sets of propositions about) objects in the domain model. For example, the structure for "the rubber tree" would Include predications of existence and treeness.

The assimilation level in the current system only goes so far as determining a literal interpretation in context. The major tasks performed here include delimiting the scope of quantifiers and associating references to objects with particular entities in the domain model, taking into account the overall dialogue and task context* In the case of our two monkeys, the system would determine whether there was a unique rubber tree in, or near, the focus of attention of the monkey (more on this shortly) and then posit, or check, the existence of a stick under it* The system does make some inferences based on the information explicitly contained in an utterance and its literal interpretation. To see the sorts of inferences it will make, consider the sequence:

(6) monkey 1: I'm going out to get some bananas.
Where is the stick?
(7) monkey2: It's under the gumgum tree.

If the system (playing the role of monkey2) knows of some plan for getting bananas from a tree that involves the use of a particular stick, it would be able to make sense of monkeyl's request, including identifying "the stick" as the one that is usually used to knock bananas out of trees. Furthermore, if the plan included steps prerequisite to getting a stick (e.g., getting a knapsack for carrying the relevant tools and the gathered bananas), the system would infer that they too had been performed.

Initial progress has been made in overcoming the limitations of literal interpretation and including a consideration of a speaker's plans and goals in the interpretation of an utterance. Recent research on the role of planning in language processing Includes that of Cohen (1978), Wilensky (1978), Carbonell (1979), and Allen (1979). Cohen (1978) views speech acts (Searle, 1969) as one kind of goal-oriented activity and describes a system that uses mechanisms previously used for planning nonlingulstic actions to plan individual speech acts (on the level of requesting and informing) intended to satisfy some goals involving the speaker's or hearer's knowledge. In Wilensky's work on story understanding (see also Schank and Abelson, 1977), the speaker's overall plans and goals, some of which are implicit, are inferred from substeps and intermediate or triggering states (e.g., inferring from "John was hungry. He got in his car." that John was going to get something to eat.) Carbonell (1979) describes a system constructed to investigate how two agents with different goals interpret an input differently; it is particularly concerned with the effect of conflicting plans on interpretation. Allen (1979) describes a system based on a model in which speech acts are

Several other systems are capable of fairly sophisticated analysis and processing at the level of coordinating different kinds of language-specific capabilities (e.g., Sager and Grishman, 1975; Landsbergen, 1976; Plath, 1976; Woods et al., 1976; Bobrow et al., 1977; Reddy et al. 1977) and of taking into account some of the ways in which context affects meaning through the application of limited action scenarios (Schank et al., 1975; Novak, 1977) or by considering (either independently or in conjunction with such scenarios) language-specific mechanisms that reference context (Hobbs, 1976; Rieger, 1975; Hayes, 1978; Mann et al., 1977; Sidner, 1979).

The system actually works on dialogues for assembling mechanical equipment. The plans it knows about are partially ordered (and not linear), and the structures It uses allow for describing plans at multiple levels of abstraction.

defined in terms of "the plan the hearer believes the speaker intended him to recognize[11] and has perhaps gone furthest in determining mechanisms by which a speaker's goals and plans can be taken into account in the interpretation of an utterance*

These efforts have demonstrated the feasibility of incorporating planning and plan recognition into the common-sense reasoning component of a natural language processing system, but their limitations highlight the need for more robust capabilities in order to achieve the integration of language-specific and general common-sense reasoning capabilities required for fluent communication in natural language. No system combines a consideration of multiple agents having different goals with a consideration of the problems that arise from multiple agents having separate beliefs and each having only incomplete knowledge about the others agent's plans and goals.[19] Furthermore, only simple sequences of actions have been considered, and no attempt has been made to treat hypothetical worlds.

One of the major weaknesses in current AI systems and theories (and the limitation of current systems that I find of most concern) is that they consider utterances as having a single meaning or effect. Analogously, a critical omission in work on planning and language is that it fails to consider the multiple dimensions on which an utterance can have effects. If utterances are considered operators (where "operator" is meant in the general sense of something that produces an effect), they must be viewed as conglomerate operators.

Although it does not yet go beyond literal interpretation (except by filling in unmentioned intermediate steps in the task being performed), the SRI language system does account for two kinds of effects of an utterance. In addition to determining the propositional content of an utterance (and what it literally conveys about the state of the world), the system determines whether the utterance indicates that the speaker's focus of attention has shifted (Grosz, 1977a,b, 1978; Sidner, 1979).[20]

To summarize then, one or more of the following crucial limitations is evident in every natural language processing system constructed to date (although most of these problems have been addressed to some extent in the research described above and elsewhere):

* Interpretation is literal (only propositional content is determined).

* The knowledge and beliefs of all participants in a discourse are assumed to be identical.

* The plans and goals of all participants are considered to be identical.

* The multifaceted nature of utterances is not considered.

To move beyond this state, the major problems to be faced at the level of linguistic analysis concern determining how different linguistic constructions are used to convey information about such things as the speaker's (implicit) assumptions about the hearer's beliefs, what entities the speaker is focusing on, and the speaker's attitude toward the hearer. The problems to be faced at the assimilation level are more fundamental. In particular, we need to determine common-sense reasoning mechanisms that can derive complex connections between plans and goals — connections that are not explicit either in the dialogue or in the plans and goals themselves — and to reason about these relationships in an environment where the problem solver's knowledge is necessarily incomplete. This is not just a matter of specifying more details of particular relationships, but of specifying new kinds of problem solving and reasoning structures and procedures that operate in the kind of environment in which natural language communication usually occurs.

COMMON-SENSE REASONING IN NATURAL LANGUAGE PROCESSING

The previous sections of this paper have suggested several complexities in the common-sense reasoning needs of natural language communication. A participant in a communicative situation typically has incomplete information about other participants. In particular he cannot assume that their beliefs, goals, or plans are identical. Communication is inherently interpersonal. Furthermore, the information a speaker conveys typically requires a sequence of utterances. As a result, interpretation requires recognition of different kinds of plans, and generation requires the ability to coordinate multiple kinds of actions to satisfy goals along multiple dimensions. Other complications are introduced by the

[19] Moore (1979) discusses problems of reasoning about knowledge and belief.

[20] Grosz and Hendrix (1978) discuss focusing as one of the elements of cognitive state crucial to the interpretation of both definite and indefinite referring expressions, and Grosz (1978) discusses several open problems in modeling the focusing process.

interactions among plans of different agents (Bruce and Newman, 1978; Hobbs and Robinson (1978) discuss some of the complexity of the relationship between an utterance and domain specific plans).

From this perspective, the current deduction and planning systems in AI are deficient in several areas critical for natural language processing. A review of the current state of the art in plan generation and recognition shows that the most advanced systems have one or another (but not both) of the following capabilities:*[1] plans for partially ordered sequences of actions can be generated detail (Sacerdoti, 1977) and recognized (Genesereth, 1978; Schmidt and Sridharan, 1977) at multiple levels of detail in a restricted subject area. However, these programs only consider single agents, assume the system's view of the world is "the correct" one, and plan for actions that produce a state change characterized by a single primary effect.

The most important directions in which these capabilities must be extended and integrated for use in the interpretation and generation of language are the following:

- It must be possible to plan in a dynamic environment that includes other active agents, given incomplete information.
- It must be possible to coordinate different types of actions and plan to achieve multiple primary effects simultaneously.
- It must be possible to recognize previously unanticipated plans.

7 CONCLUSIONS

Common-sense reasoning, especially planning, is a central issue in language research, not only within artificial intelligence, but also in linguistics (e.g., Chafe, 1978; Morgan 1978), sociolinguistics (e.g., Goffman, 1974; Labov and Fanshel, 1977), and philosophy (e.g., Kasher, 1978). The literal content of an utterance must be interpreted within the context of the beliefs, goals, and plans of the dialogue participants, so that a hearer can move beyond literal content to the intentions that lie behind the utterance. Furthermore, it is insufficient to consider an utterance as being addressed to a single purpose. Typically, an utterance serves multiple purposes: it highlights certain objects and relationships, conveys an attitude toward them, and provides links to previous utterances in addition to communicating some proposltional content.

Progress toward understanding the relationship between utterances and objectives and its effect on natural language communication will be best furthered by consideration of the fundamental linguistic, common-sense reasoning, and planning processes involved in language use and their interaction. A merger of research in common-sense reasoning and language processing is an important goal both for developing a computational theory of the communicative use of language and for constructing computer-based natural language processing systems. The next few years of research on language processing should be concerned to a large extent with issues that are at least as much issues of common-sense reasoning (especially planning issues). While common-sense reasoning research could continue without any regard for language, there is some evidence that the perspective of language processing will provide insights into fundamental issues in planning that confront AI more generally.

Finally, I want to emphasize the long-term nature of the problems that confront natural language processing research in AI. I believe we should start by adding communication capabilities to systems that have solid capabilities in solving some problem (constructing such systems first if necessary; cf. McDermott, 1976). Although it may initially take longer to create functioning systems, the systems that result will be useful, not toys. People will have a reason to communicate with such systems. Monkey2 can help monkeyl get something to eat only if he himself has a realistic conception of the complexities of monkeyl's world.

ACKNOWLEDGEMENTS

Preparation of this paper was supported by the National Science Foundation under Grant No. MCS76-22004, and the Defense Advanced Research Projects Agency under Contract N00039-79-C-0118 with the Naval Electronic Systems Command.

The Natural Language Research Group of the Artificial Intelligence Center at SRI International provides a stimulating environment in which to pursue research in language processing. Many of the ideas presented in this paper are the product of thought-provoking discussions, both oral and written, with members of this group, especially Douglas Appelt, Gary

21 Earl Sacerdoti provided me with this characterization of the current state of affairs in AI research on problem solving as well as with much useful information about problem solving issues in general.

Hendrix, Jerry Hobbs, Robert Moore, Nils Nilsson, Ann Robinson, Jane Robinson, Earl Sacerdoti, and Donald Walker. 1 would also like to thank David Levy and Mitch Marcus for many discussions about language and the insights they provided. Many of these people also provided helpful comments on this paper.

REFERENCES

Allen, J. 1979 A Plan-Based Approach to Speech Act Recognition. Technical Report No. 131/79, Department of Computer Science, University of Toronto.

Bobrow, D. G., et al. 1977. GUS, A Frame Driven Dialog System. Artificial Intelligence, 8, 155-173.

Bruce, B. and D. Newman. 1978. Interacting Plans. Cognitive Science, 2, 195-233.

Carbonell, J. G. 1979. Computer Models of Social and Political Reasoning. Ph. D. Thesis, Yale University, New Haven, Connecticut.

Chafe, W. 1978. The Flow of Thought and the Flow of Language. In T. Givon (ed.), Syntax and Semantics, Vol. 12. Academic Press, New York.

Clark, H. and C Marshall. 1978. Definite reference and mutual knowledge. Paper presented at the Sloan Workshop on Computational Aspects of Linguistic Structure and Discourse Setting. University of Pennsylvania, Philadelphia, Pennsylvania.

Cohen, P. 1978. On Knowing What to Say: Planning Speech Acts. Technical Report No. 118, Department of Computer Science, University of Toronto, Toronto, Canada.

Collins, A. 1978. Studies of Plausible Reasoning. Volume I. BBN Report No. 3810, Bolt Beranek and Newman, Cambridge, Massachusetts.

Donnellan, K. S. 1977. "Reference and Definite Descriptions". In S. P. Schwartz (Ed.), Naming, Necessity, and Natural Kind. Ithaca, New York: Cornell University Press. Pp. 42-65.

Genesereth, M. R. 1978. Automated Consultation for Complex Computer Systems. Ph. D. Thesis, Harvard University, Cambridge, Massachusetts.

Goffman, E. 1974. Frame analysis. New York: Harper.

Goffman, E. 1978. Response Cries. Language, Vol. 54, 787-815.

Grosz, B. 1977a. The Representation and Use of Focus in Dialogue Understanding. Technical Note 151, Stanford Research Institute, Menlo Park, California.

Grosz, B. J. 1977b. The Representation and Use of Focus in a System for Understanding Dialogs. Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Carnegie-Mellon University, Pittsburgh, Pennsylvania. Pp. 67-76.

Grosz, B. J. 1978. Focusing in Dialog. TINLAP-2: Theoretical Issues in Natural Language, Urbana, Illinois, 24-26 July, 1978.

Grosz, B. and G. Hendrix. 1978. A Computational Perspective on Indefinite Reference. Presented at Sloan Workshop on Indefinite Reference, University of Massachusetts, Amherst, Massachusetts, December 1978. Also Technical Note No. 181, Artificial Intelligence Center, SRI International, Menlo Park, California (in preparation).

Halliday, M.A.K. 1970. Language Structure and Language Function. In New Horizons in Linguistics, J. Lyons (ed.) (Penguin).

Halliday, M.A.K. 1977. Structure and Function in Language. Presented at Symposium on Discourse and Syntax, University of California at Los Angeles, Los Angeles, California, November 1977.

Hayes, P. J. 1978. Some Association-Based Techniques for Lexical Disambiguation by Machine. Doctoral Thesis, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland.

Hobbs, J. 1976. A Computational Approach to Discourse Analysis. Research Report 76-2, Department of Computer Sciences, City College, City University of New York, New York, New York.

Hobbs, J. 1979. Conversation as Planned Behavior. To appear in Proceedings of the Sixth International Joint Conference on Artificial Intelligence. Stanford University, Stanford, California.

Hobbs, J. and J. Robinson 1978. Why Ask? SRI Technical Note 169, SRI International, Menlo Park, California.

Kasher, A. 1978. Indefinite Reference: Indispen8ability of Pragmatics. Presented at Sloan Workshop on Indefinite Reference, University of Massachusetts, Amherst, Massachusetts, December 1978.

Labov, W. and D. Fanshel. 1977. Therapeutic Discourse. New York: Academic Press.

Landebergen, S. P. J., 1976. Syntax and Formal Semantics of English in PHLIQA1. In Coling 76, Preprints of the 6th International Conference on Computational Linguistics. Ottawa, Ontario, Canada, 28 June - 2 July 1976. No. 21.

Levy, D. 1978. Communicative goals and strategies: Between discourse and syntax. In T. Givon (ed.), Syntax and Semantics. Vol. 12* Academic Press, New York.

Mann, W., J. Moore, & J. Levin 1977. A comprehension model for human dialogue. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Carnegie-Mellon University, Pittsburgh, Pennsylvania. Pp. 77-87.

McCarthy, J. 1968. Programs with Common Sense. In M. Minsky (ed.) Semantic Information Processing, MIT Press, Cambridge, Massachusetts. Pp. 403-419.

McDermott, D. 1976. Artificial Intelligence Meets Natural Stupidity. SIGART Newsletter, no. 57, Pp. 4-9.

Moore, R. 1979. Reasoning about Knowledge and Actions. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Morgan, J. L. 1978. Two Types of Convention in Indirect Speech acts. In Peter Cole (ed.), Syntax and Semantics, vol. 9, Academic Press, Inc., New York, N.Y.

Nilsson, N. 1971. Problem Solving Methods in Artificial Intelligence. McGraw-Hill, New York.

Novak, G. 1977. Representations of knowledge in a program for solving physics problems. Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Carnegie-Mellon University, Pittsburgh, Pennsylvania. Pp. 286-291.

Pittenger, R., Hockett, C, and Danehy J., I960. The First Five Minutes* Paul Martineau, Ithaca, New York.

Plath, W. J. 1976. Request: A Natural Language Question-Answering System. IBM Journal of Research and Development, 20, 4, 326-335.

Quine. W. V. 0. 1960. Word and Object. MIT Press, Cambridge, Massachusetts.

Reddy, D. R., et al. 1977. Speech Understanding Systems: A Summary of Results of the Five-Year Research Effort. Department of Computer Science. Carnegie-Mellon University, Pittsburgh, Pennsylvania*

Rleger, C. 1975. Conceptual Overlays: A Mechanism for the Interpretation of Sentence Meaning in Context. Technical Report TR-354. Computer Science Department, University of Maryland, College Park, Maryland.

Robinson, A. 1978. Investigating the Process of Natural-Language Communication: A Status Report. Technical Note 165. SRI International, Menlo Park, California.

Rubin, A. D. 1978. A Theoretical Taxonomy of the Differences Between Oral and Written Language. In R. Spiro et al. (eds.) Theoretical Issues In Reading Comprehension. Lawrence Erlbaum, Hillsdale, New Jersey.

Sacerdoti, E. D. 1977, A Structure for Plans and Behavior. Elsevier, New York.

Sacerdoti, E. D. 1978. What Language Understanding Research Suggests about Distributed Artificial Intelligence. Proceedings of a workshop held at Carnegie-Mellon University, December 7-8, 1978.

Sager, N. and R. Grishman. 1975. The Restriction Language for Computer Grammars. Communications of the ACM, 18, 390-400.

Schank, R. C, and Yale A.I. Project 1975. SAM — A story understander. Research Report No. 43, Department of Computer Science, Yale University, New Haven, Connecticut.

Schank, R. and R. Abelson. 1977. Scripts, Plans, Goals, and Understanding. Laurence Erlbaum Associates, Hillsdale N.J.

Schmidt, C. F. and N. S. Sridharan. 1977. Plan Recognition Using a Hypothesis and Revise Paradigm: an Example. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Carnegie-Mellon University, Pittsburgh, Pennsylvania. Pp. 480-486.

Searle, J. 1969. Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, Cambridge, England.

Sidner, C. L. 1979. A Computational Model of Co-Reference Comprehension in English. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Walker, D. E. (ed.) 1978. Understanding Spoken Language. Elsevier North-Holland, New York:

Wilensky, R. 1978. Understanding Goal-Based Stories. Ph.D. Thesis. Yale University, New Haven, Connecticut.

Woods, W. A., et al. 1976. Speech Understanding Systems: Final Report. BBN Report No. 3438, Bolt Beranek and Newman, Cambridge, Massachusetts.

PROBLEM SOLVING TACTICS

Earl D. Sacerdoti
Artificial Intelligence Center
SRI International
Menlo Park, California 94025 USA

This paper describes the basic strategies of automatic problem solving, and then focuses on a variety of tactics for improving their efficiency. An attempt is made to provide some perspective on and structure to the set of tactics. Finally, some new directions for problem-solving research are discussed, and a personal perspective is provided on where the work is headed: toward greater flexibility of control and more intimate integration of plan generation, execution, and repair.

1. AUTOMATIC PROBLEM SOLVING

For intelligent computers to be able to interact with the real world, they must be able to aggregate individual actions into sequences to achieve desired goals. This process is referred to as automatic problem solving, sometimes more casually called automatic planning. The sequences of actions that are generated are called plans.

Early work in automatic problem solving focused on what Newell [1] has called "weak methods." While these problem-solving strategies are quite general and are formally tractable, they are insufficient in practice for solving problems of any significant complexity. During the last decade, a number of techniques have been developed for improving the efficiency of these strategies* Since these techniques operate within the context of the general strategies, they are termed here problem-solving tactics* The bulk of this paper consists of a description of the problem-solving strategies and a catalogue of tactics for improving their efficiency. This is followed by an attempt to provide some perspective on and structure to the set of tactics. Finally, some new directions in problem-solving research are discussed, and a personal perspective is provided on where the work is headed: toward greater flexibility of control and more intimate integration of plan generation, execution, and repair.

Because problem solving involves exploration of alternative hypothesized sequences of actions, a symbolic model of the real world, referred to as a world model, is used to enable simple simulations of the critical aspects of the situation to be run as the plans are evolved. As with all models, the world models used in problem solving are abstractions or oversimplifications of the world they model.

1.1. What is Needed to Generate Plans

The general function of an automatic problem solving system, then, is to construct a sequence of actions that transforms one world model into another. There are three basic capabilities that a problem solving system must have. These are:

a. Management of State Description Models - A state description model is a specification of the state of the world at some time. The facts or relations that are true at any particular time can be represented as some equivalent of predicate calculus formulas. (We shall refer, somewhat loosely, to these facts and relations as attributes of a state.) The critical aspect of representation for problem solving is the need to represent alternative and Jiypothetical situations, that is, to characterize the aggregate effects of alternative sequences of actions as the problem solver searches for a solution.

Preparation of this paper was supported by the Defense Advanced Research Projects Agency under contract N00039-79-C-0118 with the Naval Electronic Systems Command. Barbara Grosz, Peter Hart, Nils Nilsson, and Don Walker suggested helpful presentation tactics.

Three methods have typically been used for representing these alternatives. One method has been to include an explicit state specification in each literal or assertion (as suggested by McCarthy and Hayes [2] and implemented by Green [3]). Another alternative is to associate each literal with an implicit data context that can be explicitly referenced (as in QA4 [4]). A third choice is to have all the literals that describe the states explicitly tied up in the control structure of the problem solver (as, for example, in most problem solvers written in CONNIVER [5]).

b. Deductive Machinery - A state description model, then, contains all the information needed to characterize a particular state of the world. The information will not all be explicitly encoded, however, so a deductive engine of some sort must be provided to allow needed Information to be extracted from a model. The deductions are of two types: within a particular state (this is where traditional, "monotonic" deduction systems are used), and across states (that is, reasoning about the effects of prior actions in a sequence). The deductive machinery can be viewed as a question-answering system that allows the problem solver to retrieve information about a particular state of the world from the state description model.

c. Action Models - In addition to state description models and a means of querying them, a problem solver must have a way of modelling what changes when an action is applied in an arbitrary state. Thus, an action is described by a mapping from one state description to another. Such a mapping is usually referred to as an operator. The mapping may be specified either by procedures, as in the problem solvers based on so-called AI languages [6], or by declarative data structures. In any case, they must specify at least the expressions that the action will make true in the world model and the expressions that its execution will make untrue in the world model. Usually, to help guide the heuristic search for actions that are relevant to achieve particular goals, one of the expressions to be made true by each operator is designated in some way as its "primary effect'."

I-2* The Basic Control Strategy for Plan Generation

The process of generating a plan of action that achieves a desired goal state from a given initial state typically involves a extensive search among alternative sequences* A number of control strategies for tree search constitute the basic tools of all problem solving systems. Problem solving systems usually work backward from the goal state to find a sequence of actions that could lead to it from the initial state. This procedure generates a tree of action sequences, with the goal state at the root, instances of operators defining the branches, and intermediate states defining the nodes. A tree search process of some sort is used to find a path to a node that corresponds to the initial state. The path from initial state to goal then defines the plan. Two particular tree search strategies are discussed here since they are so commonly used.

The first of these is means-ends analysis, which was the central search algorithm used by GPS [7] and STRIPS [8]. This strategy works as follows. The "difference" between the initial and goal states is determined, and that instance of the particular operator that would most reduce the difference is chosen.

If this operator is applicable in the initial state, it is applied, creating a new intermediate state. If the goal is satisfied in the new state, the search is completed. Otherwise, the difference between the new state and the goal state is determined, an operator to most reduce the new difference is chosen, and the process continues.

If the chosen operator is not applicable, its preconditions are established as a new intermediate subgoal. An attempt is made, using the search strategy recursively, to find a sequence of operators to achieve the subgoal state. If this can be done, the chosen operator is now applicable and the search proceeds as described above. If the new subgoal cannot be achieved, a new instance of an operator to reduce the difference is chosen and the process continues as before.

A second important search strategy, used in simple problem solvers written in the so-called AI languages [6], is backtracking, which works in the following manner. If the goal is satisfied in the initial state, a trivial solution has been found. If not, an operator that, if applied, would achieve the goal Is selected. If it is applicable in the initial state, It is applied and a solution has been found. If the chosen operator is not applicable, operators that would achieve its preconditions are found, and the search proceeds as before to find plans to render them applicable. If the search falls, a different candidate operator la chosen and the process repeats.

This strategy follows a line of action out fully before rejecting it. It thus permits the search tree to be represented elegantly; all the active parts of the search tree can be encoded by the control stack of the search procedure itself, and all the inactive parts of the search tree need not be encoded at all. Because of the full search at each cycle of the process, it is critical that the correct operator be chosen first almost always. Otherwise, the simplicity of representation offered by this strategy will be amply repaid by the inefficiency of the search.

As was discussed above, these strategies are insufficient in practice for solving problems of any significant complexity. In particular, one of the most costly behaviors of the basic problem solving strategies is their inefficiency in dealing with goal descriptions that include conjunctions- Because there is usually no good reason for the problem solver to prefer to attack one conjunct before another, an incorrect ordering will often be chosen. This can lead to an extensive search for a sequence of actions to try to achieve subgoals in an unachievable order.

2. TACTICS FOR EFFICIENT PROBLEM SOLVING

2.1. Hierarchical Planning

The general strategies described above apply a uniform procedure to the action descriptions and state descriptions that they are given. Thus, they have no inherent ability to distinguish what is important from what is a detail. However, some aspects of almost any problem are significantly more important than others. By employing additional knowledge about the ranking in importance of aspects of the problem description, a problem solver can concentrate its efforts on the decisions that are critical while spending less effort on those that are relatively unimportant.

Information about importance can be used in several ways. First, the standard strategies can be modified to deal with the most important (and most difficult to achieve) subgoals first. The solution to the most important subgoals often leaves the world model in a state from which the less important subgoals are still achievable (if not, the weaker search strategies must be employed as a last resort). The less important subgoals could presumably be solved in many ways, some of which would be incompatible with the eventual solution to the important subgoals. Thus, this approach constrains the search where the constraints are important, and avoids overconstraining it by making premature choices about how to solve less important aspects of the problem. Siklossy and Dreussi [9] used an explicit ordering of types of subgoals to guide search.

Another way to focus on the critical decisions first is to abstract the descriptions of the actions, thereby creating a simpler problem. This abstracted problem can be solved, producing a sequence of abstracted actions, and this plan can then be used as a skeleton, identifying critical subgoals along the way to a solution, around which to construct a fully detailed plan. This tactic was used in conjunction with an early version of GPS by Newell, Shaw, and Simon [7] to find proofs in symbolic logic (using abstracted operators and state descriptions that ignored the connectives and the ordering of symbols).

Finally, abstraction can be extended to involve multiple levels, leading to a hierarchy of plans, each serving as a skeleton for the problem solving process at the next level of detail. The search process at each level of detail can thus be reduced to a sequence of relatively simple subproblems of achieving the preconditions of the next step in the skeleton plan from an initial state in which the previous step in the skeleton plan has just been achieved. In this way, rather complex problems can be reduced to a sequence of much shorter, simpler subproblems. Sacerdoti applied this tactic to robot navigation tasks [10] and to more complex tasks involving assembly of machine components [11].

2.2. Hierarchical Plan Repair

A side-effect of hierarchical planning is that plans can possibly be created that appear to be workable at a high level of abstraction but whose detailed expansions turn out to be invalid. The basic idea behind the plan repair tactic is to check, as a higher level plan is being expanded, that all the intended effects of the sequence of higher-level actions are indeed being achieved by the collection of subsequences of lower-level actions. By exploiting the hierarchical structure of the plan, only a small number of effects need to be checked for. Various methods for patching up the failed plan can then be applied. This tactic was incorporated in a running system by Sacerdoti [11] and is very similar to a technique called "hierarchical debugging" articulated by Goldstein [12] for a program understanding task.

2.3. Bugging

Rather than attempt to produce perfect plans on the first attempt, it can often be more efficient to produce an initial plan that is approximately correct but contains some "bugs," and subsequently alter the plan to remove the bugs. By employing additional knowledge about bug classification and debugging, this approach allows the decisions made during the problem-solving process about which action to try next to be made with less effort, since mistaken decisions can be subsequently fixed.

Sussman, who first employed this tactic In his HACKER system [13], called It "problem-solving by debugging almost-right plans." It Is often referred to In the literature as the "debugging approach" and, Indeed, It has spawned Interesting research In techniques for debugging programs or plans developed both by machines and by people (see, for example, Sussman [14] and Goldstein and Miller [15]). Debugging, however, Is an Integral part of the execution component of any problem solver. What distinguishes this approach Is a tolerance for Introducing bugs while generating the plan, and thus it can more accurately be called the "bugging" approach.

This tactic works by deliberately making assumptions that oversimplify the problem of Integrating multiple subplans. These assumptions may cause the problem solver to produce an initial plan with bugs in it. However, if the oversimplifications are designed properly, then only bugs of a limited number of types will be introduced, and relatively simple mechanisms can be implemented to remedy each expected type of bug.

2.4# Special-Purpose Subplanners

Once a particular subgoal has been generated, it may well be the case that it is of a type for which a special purpose algorithm, a stronger method than the weak method of the general-purpose problem solver, can be brought to bear. For example, In a robot problem, the achievement of an INROOM goal can be performed by a route-finding algorithm applied to a connectivity graph representing the interconnection of rooms, rather than using more general methods applied to less direct representations of the rooms in the environment. Such a special purpose problem solver was used by Siklossy and Dreussl [9] to effect dramatic Improvements in the system's performance.

Wilklns [16] employs special-purpose subplanners in a chess problem solver for subgoals such as moving safely to a given square or checking with a given piece.

Each special-purpose subplanner encodes additional knowledge about its specialty. To take advantage of It, the problem solver must incorporate information about how to recognize the special situation as well.

2.5. Constraint Satisfaction

Constraint satisfaction, the derivation of globally consistent assignments of values to variables subject to local constraints, is not usually thought of as a problem solving tactic. While it cannot be used to generate action sequences, it can play a very important role In particular subproblems, especially in determining the binding of variables when there is no clear locally computable reason to prefer one value over another. From this perspective, constraint satisfaction can be thought of as a type of special-purpose subplanner.

Stefik [17] employs constraint satisfaction to assign values to variables in generating action sequences for molecular genetics experiments.

2.6. Relevant Backtracking

As a correct plan is being searched for, a problem solver will encounter many choice points at which there are several alternative steps to be taken. Most problem solvers employ sophisticated techniques to try to make the right choices among the alternative action sequences initially. Alternatively, a problem solver could focus on sophisticated post-mortem analyses of the information gained from early attempts that fail. By analyzing the reasons for the failure of a particular sequence of actions, the problem solver can determine which action in the sequence should be modified. This is in contrast with the straightforward approach of backtracking to the most recent choice point and trying other alternatives there. Fahlman [18] developed such a system for planning the construction of complex block structures. His system associated a "gripe handler" with each choice point as it was encountered, and developed a characterization of each failure when it occurred. When a particular line of action failed, the gripe handlers would be invoked in reverse chronological order to see if they could suggest something specific to do about the failure. The effect of this mechanism is to backtrack not to the most recent choice point, but to the relevant choice point.

The tactic of relevant backtracking, which is also referred to in the literature as dependency-directed or non-chronological backtracking, was also used in a problem solver for computer-aided design developed by Latombe [19].

2.7. Disproving

Problem solvers have traditionally been automated optimists. They presume that a solution to each problem or subproblem can be found if only the right combination of primitive actions can be put together. Thus, the impossibility of achieving a given goal or subgoal state can only be discovered after an exhaustive search of all the possible combinations of potentially relevant actions.

It may well be the case that a pessimistic analysis of a particular goal, developing knowledge additional to that employed in building action sequences, would quickly show the futility of the whole endeavor. This procedure can be of particular value in evaluating a set of conjunctive subgoals to avoid working on any of them when one can be shown to be impossible. Furthermore, even if a goal cannot be shown to be impossible, the additional knowledge might suggest an action sequence that would achieve the goal. Siklossy and Roach [20] developed a system that integrated attempts to achieve goals with attempts to prove their impossibility.

2-8. Pseudo-Reduction

One of the most costly behaviors of problem solving systems is their inefficiency in dealing with goal descriptions that include conjunctions, as was noted at the end of Section 1. Ordering the conjuncts by importance, as described in the subsection on hierarchical planning above, can help, but there may still be multiple conjuncts of the same importance. One approach to the problem of selecting an order for the conjuncts is to ignore ordering them initially, finding a plan to achieve each conjunct independently. Thus, the conjunctive problem is reduced to several simpler, nonconjunctive problems. Of course, the plans to solve the reduced problems must then be integrated, using knowledge about how plan segments can be intertwined without destroying their important effects.

This tactic creates plans that are not linear orderings of actions with respect to time, but are rather partial orderings. This renders the cross-state question-answering procedure described in Section 1.1 more complicated than for other tactics.

By avoiding premature commitments to particular orderings of subgoals, this tactic eliminates much of the backtracking typical of problem solving systems. Pseudo-reduction was developed by Sacerdoti [11] and has been applied to robot problems, assembly of electromechanical equipment, and project planning [21].

2.9. Goal Regression

The problem-solving tactics we have discussed so far all work by modifying, in one way or another, the sequence of actions being developed to satisfy the goals. Coal regression modifies the goals as well. It relies on the fact that, given a particular goal and a particular action, it is possible to derive a new goal such that if the new goal is true before the action is executed, then the original goal' will be true after the action is executed. The computation of the new goal, given the original goal and the action, is called regressing the goal over the action.

As an example, let us suppose the overall goal consists of two conjunctive subgoals. This tactic first tries to achieve the second goal in a context in which a sequence of actions has achieved the first goal (similarly to the bugging tactic).

If this fails, the second goal is regressed back across the last action that achieved the first goal. This process generates a new goal that describes a state such that if the last action were executed in it, would lead to a state in which the original second goal were true. If the regressed goal can be achieved without destroying the first goal, the tactic has succeeded. If not, the regression process continues.

This tactic requires knowledge of the inverse effects of each operator. That is, in addition to knowing how the subsequent application of an operator changes a world model, the system must know how the prior application of the operator affects a goal.

The goal regression technique was developed independently by Waldinger [22] and Warren [23].

3. WHAT'S GOING ON?

We have just finished a brief (heuristically) guided tour of some of the problem-solving tactics used recently. They constitute a diverse bag of tricks for improving the efficiency of the problem solving process. In this section we focus on the underlying reasons why these techniques seem to help.

Problem-solving is often described as state-space search, or as exploration of a tree of possible action sequences. We can find some structure for the bag of tactical tricks by remembering that search or exploration involves not only movement to new (conceptual) locations, but discovery and learning as well.

The problem solver begins its work with information about only the initial state and the goal state* It must acquire information about the intermediate states as it explores them. It must summarize this Information for efficient use during the rest of the problem-solving process, and it must take advantage of all possible information that can be extracted from each intermediate state. This can require considerable computational resources. In the simplest search strategies, the information may be simply a number representing the value of the heuristic evaluation function applied at that point. It may be much more, however. It may Include a detailed data base describing the situation, information about how to deal with classes of anticipated subsequent errors, and dependency relationships among the attributes describing the situation. All this information is typically stored in intermediate contexts in one of the forms discussed in the first section of this paper.

The information learned during the exploration process can be broken down into four kinds of relationships among the actions in a plan. These are:

order relationships - the sequencing of the actions in the plan;

hierarchical relationships - the links between each action at one level and the meta-actions above it and the more detailed actions below it;

teleological relationships - the purposes for which each action has been placed in the plan; and

object relationships - the dependencies among the objects that are being manipulated (which correspond to dependencies among the parameters or variables in the operators).

These relationships can be explicated and understood only by carrying out the instantiation of new points in the search space. It is thus of high value to a problem solver to instantiate and learn about new intermediate states. As a simple example of a problem-solving tactic that displays incremental learning, consider relevant backtracking. By following initial paths through the search tree, the problem solver learns which choice points are critical. Another clear example is constraint satisfaction, in which restrictions on acceptable bindings for variables are aggregated as search progresses.

The difficulty is that, as with all learning systems, the acquisition of new information is expensive. The generation of each new state is a major time consumer in many problem solving systems. Furthermore, the generation of each intermediate state represents a commitment to a particular line of action by the problem solver. Since problems of any non-toy level of complexity tend to generate very bushy search trees, a breadth-first search strategy is impossible to use. Therefore, once a line of action has begun to be investigated, a problem-solving system will tend to continue with it. It is thus of high value to a problem solver, whenever possible, to avoid generating intermediate states not on the solution path. Those states that are generated must represent a good investment for the problem solver.

Thus, there are two opposing ways to improve the efficiency of a problem solver. The first is to employ a relatively expensive evaluation function and to work hard to avoid generating states not on the eventual solution path. The second is to use a cheap evaluation function, to explore lots of paths that might not work out, but to acquire information about the interrelationships of the actions and objects in the world in the process. This information can then be used to guide (efficiently) subsequent search.

Each of the tactics described in the previous section strikes a particular balance between the value of instantiating new intermediate states and the cost of commitment to particular lines of action. While none of the tactics use one approach exclusively, each can be categorized by the one it emphasizes. Furthermore, each can be distinguished according to one of the four types of relationships they depend on or exploit. Table 1 displays a candidate categorization.

None of the tactics fit as neatly into the classification as the table suggests, because they typically have been embodied in a complete problem solving system and so must deal with at least some aspects of many of the categories.

Relationship:	Approach: Learn and Summarize	Choose New Move
Order	pseudo-red. generation relevant backtracking disproving	
Hierarchy	plan repair	spec.-purpose subplanners
Teleology	bugging pseudo-red. critics	regression
Object	relevant backtracking	constraint satisfaction

TABLE 1

Classification of Tactics

4. WHAT'S NEXT?

The current state of the art in plan generation allows for planning in a basically hierarchical fashion, using a severely limited (and predetermined) subset of the tactics enumerated above, by and for a single active agent satisfying a set of goals completely. The elimination of these restrictions is a challenge to workers in Artificial Intelligence. This section will discuss a number of these restrictions briefly and suggest, where possible, lines of research to ease them.

4.1. Integrating the Tactics

To date, there has been no successful attempt known to this author to integrate a significant number of the tactics we have described into a single system. What follows are some preliminary thoughts on how such an integration might be achieved.

First of all, the technique of hierarchical planning can be applied independently of any of the others. That is, all of the other techniques can be applied at each level of detail within the hierarchy. A number of interesting problems (analogous to that faced by the "hierarchical plan repair[1]' tactic) would have to be faced in integrating their application across levels of the hierarchy.

Approaches for dealing with each of the types of relationships shown in Table 1 can probably be selected without major impact on the approaches selected for the other relationships. Thus, we can use Table 1 as a menu of possible tactics from which various collections can be constructed that make sense together in a problem solver.

Tactics that emphasize the clever selection of new paths to explore in the search space might be difficult to integrate with tactics that emphasize the learning and summarization of information derived from the portion of the search space already explored. However, the major payoff in integrating tactics might come from exactly this kind of combination. Developing a problem solver that uses both kinds of technique when appropriate will probably require the use of novel control strategies. The interesting new results from such an endeavor will derive from efforts to employ information developed by one tactic in the application of other tactics.

4.2. Flexible Control Structure

While the tactic of hierarchical planning speeds up the problem solving process greatly, it requires that a plan be fully developed to the finest detail before it is executed. In real-world environments where unexpected events occur frequently and the detailed outcome of particular actions may vary, creation of a complete plan before execution is not appropriate. Rather, the plan should be roughed out and its critical segments created in detail. The critical segments will certainly include those that must be executed first, but also may include other aspects of the plan at conceptual "choke points" where the details of a subplan may affect grosser aspects of other parts of the plan.

Hayes-Roth *et* al. [24] are developing a program based on a model of problem solving that would produce the kind of non-top-down behavior suggested here. Their model is based on a Hearsay-II architecture [25], but could probably be implemented using any methodology that allowed for explicit analysis of each of the four kinds of dependencies described in Section III above. Stefik [17] has implemented a system that, at least in principle, has the power to produce this kind of behavior. His system incorporates a flexible means of determining which planning decision to make next. His decisions are local ones; should global ones be incorporated as well, we might see a means of determining dynamically which tactic to employ in a given situation*

4.3. Planning for Parallel Execution

Problem solvers to date have been written with the idea that the plan is to be generated by a single processor and will ultimately be executed one step at a time. The development of cooperating problem solvers and algorithms for execution by multiple effectors will force a closer look at the structure of plans and the nature of the interactions between actions*

A solid start has been made in this area. FIkes, Hart, and NIlsson [26] proposed an algorithm for partitioning a plan among multiple effectors- Smith [27] developed a problem solver that distributes both the plan generation and execution tasks. The pseudo-reduction tactic creates plans that are partially ordered with respect to time, and are therefore amenable both to planning in parallel by multiple problem solvers and to execution in parallel by multiple effectors. Corkill [28] is adapting the NOAH [11] pseudo-reduction problem solver to use multiple processors in plan generation, and we at SRI are adapting it to plan for the use of multiple effectors.

4.4. Partial Goal Fulfillment

Problem solvers to date have been designed to fully satisfy their goals. As the problems we work with become more complex, and as we attempt to integrate problem solvers with execution routines to control real-world behavior, full goal satisfaction will be impossible. In particular, a system that deals with the real world may need to execute a partially satisfactory plan and see how the world reacts to it before being able to complete the next increment of planning. Thus, we must be able to plan for the partial satisfaction of a set of goals. This implies that a means must be found of prioritizing the goals and of recognizing when an adequate increment in the planning process has been achieved.

5. A PERSPECTIVE

The problems we have posed have a common theme to their solution: increased flexibility in the planning process. The metaphor of problem solving as exploration for information that was presented above suggests that the result of pursuing the problem solving process can lead to surprises as great as those encountered in plan execution. The plan execution components of problem solving systems have been forced to be quite flexible because of the surprises from the real world that they had to deal with. Therefore, especially as the tactics used in exploring for plans become more daring, the lessons that can be learned from plan execution can be extremely valuable for plan generation.

This view suggests a direction for future work in problem solving: it will become more like incremental plan repair. The means of storing and querying state description models will have to allow for efficient updating when the orders of actions are altered and when new actions are inserted in mid-plan. Planning at higher levels of abstraction will appear very similar to planning for information acquisition during plan execution.

Therefore, the best research strategy for advancing the state of the art in problem solving might well be to focus on integrated systems for plan generation, execution, and repair. By developing catalogues of plan execution tactics and plan repair tactics to accompany this catalogue of plan generation tactics, we can begin to deal with problems drawn from rich, interactive environments that have thus far been beyond us.

REFERENCES

1. A. Newell, "Heuristic Programming: Ill-Structured Problems," in J. Aronofsky (ed.) Progress in Operations Research, Vol. III, pp. 360-414 (Wiley, New York, 1969).

2. J. McCarthy and P. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," Machine Intelligence 4, B. Meltzer and D. Michie, eds., pp. 463-502 (American Elsevier, New York, 1969).

3. C. Green, "The Application of Theorem-Proving to Question-Answering Systems," Doctoral dissertation, Elec. Eng. Dept., Stanford Univ., Stanford, CA, June 1969. Also printed as Stanford Artificial Intelligence Project Memo AI-96 (June 1969).

4. R.F. Rulifson, J.A. Derksen and R. J. Waldinger "QA4: A Procedural Calculus for Intuitive Reasoning," Artificial Intelligence Center, Technical Note 73, Stanford Research Institute, Menlo Park, California (November 1972).

5. D.V. McDermott and G.J. Sussman, "The CONNIVER Reference Manual," MIT, Artificial Intelligence Lab., Memo No. 259, Cambridge, MA (May 1972).

6. D.G. Bobrow and B. Raphael, "New Programming Languages for Artificial

Intelligence," Computing Surveys, Vol. 6, No. 3 (Sept. 1974).

7. G.W. Ernst and A. Newell GPS: A Case Study in Generality and Problem Solving (Academic Press, New York, 1969).

8. R.E. Fikes and N.J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence. Vol. 2, No. 3-4, pp. 189-208 (Winter 1971).

9. L. Siklossy and J. Dreussi, "An Efficient Robot Planner Which Generates Its Own Procedures," Proc Third International Joint Conference on Artificial Intelligence, Stanford, California, (August 1973).

10. E.D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," Artificial Intelligence, Vol. 5 No. 2, pp. 115-135 (Summer 1974).

11. E.D. Sacerdoti, A Structure for Plans and Behavior, (Elsevier North-Holland, New York, 1977).

12. I.P. Goldstein, "Understanding Simple Picture Programs," Tech. Note AI-TR-294, Artificial Intelligence Laboratory, MIT, Cambridge, MA (September 1974).

13. G.J. Sussman, A Computer Model of Skill Acquisition, (American Elsevier, New York, 1975).

14. G.J. Sussman, "The Virtuous Nature of Bugs," Proc. AISB Summer Conference, pp. 224-237 (July 1974).

15. M.L. Miller and LP. Goldstein, "Structured Planning and Debugging," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 773-779, Cambridge, Massachusetts (August 1977).

16. D. Wilkins, "Using Plans in Chess," Proc Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979).

17. M.J. Stefik, "Orthogonal Planning with Constraints, Report on a Knowledge-Based Program that Plans Synthesis Experiments in Molecular Genetics," forthcoming dissertation, Computer Science Department, Stanford University (September 1979).

18. S.E. Fahlman, ^{M}A Planning System for Robot Construction Tasks," Artificial Intelligence, Vol. 5, No. 1, pp. 1-49 (Spring 1974).

19. J.C. Latombe, "Artificial Intelligence in Computer-Aided Design: the TROPIC System," in J.J. Allen (ed.), CAD Systems (Elsevier North-Holland, New York, 1977).

20. L. Siklossy and J. Roach, "Collaborative Problem-Solving between Optimistic and Pessimistic Problem Solvers," Proc IFIP Congress 74, pp. 814-817 (North-Holland Publishing Company, 1974).

21. A. Tate, "Generating Project Networks," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 888-893, Cambridge, Massachusetts (August 1977).

22. R. Waldinger, "Achieving Several Goals Simultaneously," in E.W. Elcock and D. Michie (eds.) Machine Intelligence 8, pp. 94-136 (Ellis Horwood Limited, Chichester, England, 1977).

23. D.H.D. Warren, "WARPLAN: A System for Generating Plans," Department of Computational Logic, Memo No. 76, University of Edinburgh, Edinburgh (June 1974).

24. B. Hayes-Roth, F. Hayes-Roth, S. Rosenschein, and S. Cammarata, "Modeling Planning as an Incremental, Opportunistic Process," Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979).

25. V.R. Lesser and L.D. Erman, "A Retrospective View of the Hearsay-II Architecture," Proc. Fifth International Joint Conference on Artificial Intelligence, pp. 790-800, Cambridge, Massachusetts (August 1977).

26. R.E. Fikes, P.E.Hart, and N.J. Nilsson, "Some New Directions in Robot Problem Solving," in B. Meltzer and D. Michie (eds.) Machine Intelligence 7, pp. 405-430 (Edinburgh Univ. Press, Edinburgh, 1972).

27. R.G. Smith, "A Framework for Distributed Problem Solving," Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979).

28. D.D. Corkill, "Hierarchical Planning in a Distributed Environment," Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, Japan (August 1979).

Artificial Intelligence Research Strategies In the Light of AI Models of Scientific Discovery

Herbert A. Simon
Department of Psychology
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract and Introduction

Some recent artificial intelligence programs whose task is to simulate the processes of scientific discovery can be taken as models of the history *and* processes of discovery within the AI discipline Itself. Consistently with these models, AI research relies basically on the methods of heuristic best-first search. Because of Its necessarlly vague end open goals, it works forward inductively (rather than backward In menea-ends fashion), guided by a crude evaluation function that tests running programs to identify promising directions.

AI reseerch Is empirical and pragmatic, typically working with examples rather than theorems, and exemplfyin the heuristic of learning by doing. In Its essential reliance on weak methods and experiment Insteed Of proof, it is edepted to the exploration of poorly structured task domains, showing considerable contrest In the respect to operetione reseerch or numerical analysis, which thrive best In domains possessing strong formal structure.

At scientific meetings it is customary to schedule, in addition to papers reporting specific pieces of research, "addresses, "keynote speeches," and the like, which may be described .as meta-papers. The task of meta-papers la not to report research but to Interpret the past and to peer into the future of the discipline. This is such a meta-paper. Presumably you expect me to say where artificial intelligence has been and where it is going.

Clearly, this is not a task for human intelligence. Human beings are notoriously incapable of reviewing history — especially history in which they have participated — without rationalizing outrageously to make the pest conform to their picture of the present. And human forecests of the future almost always reveal much more ebout the forecasters' hopes, fears, desires and dreads than they do about the shape of the world to come.

Early in the history of AI, in 1957, Allen Newell and 1 made some predictions that became rather notorious (Simon & Newell, 1958). Skeptics and opponents of AI used them as evidence of the recklessness and irresponsibility of the advocates of AI. (Optimistic forecasts seem to attract such charges much more often than do doomsday forecasts.)

Of course our forecasts were neither reckless nor irresponsible. As we said at the time, they represented our attempt to define in concrete terms the nature of the revolution in human affairs that was going to be produced by computers in general and artificial Intelligence in particular. As scientists privileged to witness the early stages of a momentous development, we felt a responsibility to interpret that development to leymen, and the predictions were our interpretation. Nor was our forecasting seriously inaccurate, if one allows a time-stretch factor of two or three — a not unreasonable mergin of inaccuracy in such crystal-ball ventures.

I cite this little piece of history not to defend my record as either a seer or a historian, but as empirical evidence for my doubts, expressed earlier, that either foresight or hindsight are fit tasks for human intelligence. Such doubts undermine the *very* foundations of meta-pepers, including this one.

If human Intelligence Is unequal to the needs of history and prophecy, perhaps we should call on artificial intelligence. Perhaps we should ask what AI has to say about the processes of discovery. After all, we do have, todey, a number of artificial Intelligence programs that are capable of making discoveries of one kind or another — I have in mind particularly Doug Lenat's AM *program,* and Pat Langley's BACON. Perhaps these programs can tell us more about the research process than human beings can.

An AI program that makes genuine discoveries, or one that solves difficult problems, provides us with a theory of the discovery process, Indeed, a theory in the most concrete and explicit form that is conceivable. Since these programs reveal to us some of the essential requisites and structure of the discovery process, we can use them to illuminate the history of discovery in the domain of artificial intelligence itself, and to provide soma insight into the ways in which we can best proceed in future research and development aimed at new discoveries in that field.

This is the path I propose to pursue in this paper. First, I will summarize what seem to me some of the salient characteristics of successful artificial intelligence problem-solving systems, especially those whose basic task is to make discoveries. Next, I will ask whether this list of program characteristics suggests why the process of discovery in the AI field itself has taken the particular course that it has. Finally, I will turn to the future, and ask what lessons we might learn from this experience In our continuing efforts to extend the boundaries of AI, particularly in the directions of greater capabilities for discovery and for solving Ill-structured problems. If this route seems somewhat circular — AI illuminating itself -- 1 remind you that circles may be either vicious or virtuous, and I will argue that this is one of the virtuous kind.

I will not try to cover every aspect of AI, and w undoubtedly overemphasize problem-solving and heuristic search at the expense of such areas as visual pattern recognition. This lapse will be the less serious to the extent that the techniques of heuristic search are today Invading the domain of pattern recognition, bringing about a greater degree of unity in outlook throughout the whole field of artificial Intelligence. So I will take, as Allen Newell and I did In our Turing Lecture, heuristic search as the central paradigm for artificial intelligence (Newell & Simon, 1976).

1. AI Programs at Theories of Discovery

By a discovery program I mean a computer program whose output is not inferable in any obvious way from its Input. The phrase, "in any obvious way," is essential to the definition, since we know that a program does exactly what we program It to do — which is usually not at all the same es doing what we supposed we had programmed it to do.

2* The Nature of Discovery

Novelty, In computers as In human beings, lies in the eye of the beholder. The result Is novel if it was not expected from the outset. But even this definition is ambiguous. As the numerous documented cases of Independent Invention attest, a discovery may be novel to the discoverer but not to the whole society, for others may already have found it. However, to produce a novelty a second time, without knowledge that it has already been discovered by others, presumably requires the same kinds of cognitive processes as were required to produce It the first time. Anything we can learn by examining the program of the original discoverer we should be able \o learn also by examining the program of the rein venter.[1]

It is probably true today that within any one hour period some computer program somewhere in the world has followed a path *rsevnr* before traversed, to produce a novel successful result. This must occur for example, more than once in almost every game played by a hobbyist's minicomputer chess program, since chess games rarely fully repeat others that have been played In the world. However, we generally do not *$pp\y* the term "discovery" to every novelty of this kind, however rational or adaptive the output may be. We require, In addition, that the novelty be in some sense remarkable or socially valuable. In particular, and borrowing language from the patent law, to be an invention, a novelty must not be "obvious to a person skilled in the art." While a minicomputer playing Class D chess discovers many novel solutions to its problems, these solutions would Rresumably be discovered easily by strong players, ence would not qualify as inventions in the legal sense.

Even today, after a quarter century of AI efforts, it is hard to point to fully convincing examples of discoveries by artificial intelligence programs that satisfy this stricter definition, of being neither rediscoveries nor obvious to one skilled In the art. If pressed on this point, I might want to defend certain products of chess programs, programs for musical composition and visual design, and theorem-proving programs as meeting the stricter requirements of invention, but such a defense would take me away from *my* main concern here.

I will draw my examples of discovery from computer programs that have mainly rediscovered what was already known, but whose discoveries are hardly trivial, and would indeed have been adjudged important if they had been genuinely new. I have in mind such examples es the discovery by Lenat's AM program of the concept of prime number and it* conjecturing of the fundamental theorem of arithmetic — that *e^ery* positive integer can be represented uniquely as a product of powers of primes (Lenat, 1977). Examples of a slightly different kind are BACON'S induction from empirical data of Kepler's Third Law, Ohm's Law, and the Laws of Boyle and Charles (Langley, forthcoming).

Before we take the programs that found these concepts and laws as exhibiting the essential processes for discovery, we must satisfy ourselves on one point: that the human programmers did not, in some explicit or implicit way, embed the results at the outset in the programs and their inputs. Since we have already agreed that the outputs of programs are determined by the programs (and data), what can we mean by this requirement? Simply that the derivation of the outputs from program and data be sufficiently non-obvious. This is, of course, the same criterion we apply to a mathematical theorem to determine whether it is "deep"; and it is the same as the legal requirement for invention, quoted above.

This is not to say that it is a precise criterion, statable in a formal way. The only way I know to decide whether AM or BACON, or any other program purporting to have powers of discovery (but exhibiting those powers through rediscovery) genuinely possesses such capabilities is to search the code carefully for hideaways where the conclusions may be concealed in the premises. The severity of the test will depend on how thoroughly the search is made and how strict a criterion of obviousness is applied. Since I know of no way at the present time to quantify either of these two dimensions of the test, we must still depend (as we do in evaluating the merit of scientific discoveries) on informal judgement.

From close familiarity with the AM and BACON programs, I am satisfied that these two programs pass any reasonable tests of this kind. Let me, then, comment on the structure of the programs — on the sources of their powers of discovery. For the sake of those of you who *9re* not acquainted with AM or BACON, I will first state briefly what each program does. As already noted, a fuller description of AM, by Doug Lenat, will be found in the Proceedings of the Fifth UCAIU977), and of BACON, by Pat Langley, in the Proceedings of this conference.

Lenat's AM Program AM is a system that discovers new concepts and that conjectures new relations among them. Its input consists of an initial stock of concepts (in one application, the basic notions of set theory), goals and criteria (the goal of discovering new concepts and possible relations among concepts, and criteria for evaluating the worth or interest of concepts), and heuristics for searching for new concepts. Among the criteria for judging if a concept is interesting is how closely it is related to other interesting concepts, and whether examples of it can be constructed — not too easily, but with not too much difficulty. The search heuristics include the aovice to construct examples, to

[1] *Thia claim raqutras torn* qualifications. TKt rtinvtntor may poitttt knowtadfe — not tha mvantion itastf, but knowttdft ramvtnt to it - that waa not availabla to tha ordinal invantor, but which mahoe tha Job ttsiar ♦ha Mcond time. Latar, I wiM hava mora to sty on thit point ■■ it •ppfct apactf ieaHy to At

pay particular attention to borderline examples, to particularize when examples are found too easily, to generalize when they are hard to find This is the Kind of initial information available to AM — initial concepts, goals and criteria, and search heuristics.

The control structure of AM guides it in a best-first search} the criteria of concept worth determine which of the concepts already attained should be the starting point for the next quantum of search. On its most celebrated run, starting with the concepts of set theory, AM discovered — among other things — the integers, the arithmetic operations of addition, subtraction, multiplication and division, the concept of prime number, and, as I mentioned earlier, the prime number representation theorem. When it began concerning itself with numbers possessing maximal (instead of minimal) numbers of prime factors, Lenat thought it had entered on truly new ground, only to find that this territory had earlier been explored by the self-taught Indian mathematician, Srinivasa Ramanujan. Therefore, AM must be evaluated as a rediscoverer, rather than a discoverer of new mathematical truths.

Lanelev's BACON Program BACON is a program that induces general laws from empirical data. Given sets of observations on two or more variables, BACON searches for functional relations among the variables. Again, it carries out a form of best-first search, in which a criterion of "simple things before complex" guides what to try next.

BACON'S search is highly selective; it does not try all possibilities. It arranges the observations monotonically according to the values of one of the variables. Then it determines whether the values of some other variables follow the same (or the inverse) ordering. Picking one of these other variables, it searches for an invariant by considering the ratio (resp., product) of this variable with the original one. If the ratio (product) is not constant, it is introduced as a new variable, and the process continues. Thus, the newly defined variables in BACON correspond to the new concepts in AM, and the process Is driven by a search for invariants.

It is easy to see how BACON, discovering that the product of electrical current by resistance in an electrical circuit was constant, would be led to Ohm's Law. The case of Kepler's Third Law requires BACON to generate, successively, ratios of powers of the radii of the planets orbits to powers of their periods of revolution, arriving at the invariant, D^3/P^2, after a search of a small number of possibilities.

I have mentioned only a few of the salient features of BACON. The system has at least crude means for ignoring noise as data, and a number of other interesting features, but I will leave their fuller description to the program's author. What is interesting for our purposes is that a program, equipped and organized as I have described, detects regularities in data sufficiently perceptively to rediscover important scientific laws.

AM and BACON use similar schemes of memory organization. The ability to apply the same basic processes to given information and to newly generated concepts or variables, respectively, hence to operate recursively, is guaranteed by using a homogeneous format for the storage of all data. Though the details of the data structures are different for the two programs, both use schemes — structures of *proper\y* lists -- to describe the objects with which they deal or which they generate. The main element of rigidity in their memory organizations is that the specific properties that may occur in these schemes — the "slots" — a*re* specified In advance and known to the programs.

Discovery Mechanisms The theory of discovery that emerges from an examination of how these programs work contains little that should surprise us — unless we have been seduced by the often-repeated myth that discovery processes, being "creative," somehow stand apart from the other actions of the human mind. In AM and BACON we see discoveries being produced by precisely the same kinds of symbolic processes that account for the efficacy of other AI problem-solving programs: theorem provers, chess players, puzzle solvers, diagnosis systems. A space of possible concepts a*nd* relations (AM), or of possible invariants (BACON) is searched in a highly selective, best-first manner. The search mainly works forward inductively from the given concepts or data.

The discovery programs are distinguished from most other problem-solving systems in the vagueness" of the tasks presented to them and of the heuristic criteria that guide the search and account for its selectivity. Because the goals are very general ("find an interesting concept or relations, "find an invariant"), the use of means-ends analysis to work backward from a desired result is not very common. By and large, the programs work forward inductively from the givens of the problem and from the new concepts and variables generated from these givens.

Both programs work at a very concrete level. AM makes a major use of examples, which it is capable of generating, in searching for new concepts. BACON works with numerical data. If we observed human scientists working in the manner of these programs, we would *regard* them as very pragmatic. We are reminded of Faraday's notebooks, in which he recorded, day after day, the experiments that were suggested to a curious mind by the findings of the previous day's experiment. Or, we think of Mendeleev arranging and rearranging his lists of the elements until their periodic structure begins to emerge from his worksheets.

Both programs discover, they do not prove. Their task is to find regularity and pattern in nature, not to demonstrate the necessity of that pattern. Although their heuristics appear *very* general and weak — they do not rely et all on semantic information about the task domain that is being explored — they accomplish the search tasks with a remarkably small amount of trial and error. In the best tradition of heuristic schemes, they operate without a*ny* guarantees that they will succeed, but they do succeed in finding many interesting results. We would not even know how to define completeness for programs given these kinds of ill-defined tasks.

Because the tasks addresssed by these systems are poorly defined, we do not have good measures of how powerful they are. Of course, we can make our personal evaluations of the quality of their discoveries — of how Impressed we are that AM finds the prime number factorization theorem, or that BACON readily Induces Ohm's Law from the data. But we do not have the precision of comparison with human performance that a chess program gives us, or a program for medical diagnosis. The difficulty of evaluating them is compounded by the absence of a yardstick for measuring the knowledge with which they are endowed at the outset, or that is embedded in the program structures. We do not know whether BACON had the same starting point as Ohm, or whether one of them was faced with an essentially simpler problem of induction than the other. Of course the same uncertainties surround all of our attempts to evaluate human discovery also. AM and BACON pose no new methodological puzzles in this respect.

How does the behavior of these programs compare with the behavior of the human scientists who have labored in the vineyards of artificial intelligence during the past 25 years? Do AM and BACON provide a true, if rough *and* approximate, description of that discovery effort? And what of the future of AI? Can these discovery programs help us in either prediction or strategy? Let me turn first to the history.

3. The Discovery Process in AI

Artificial intelligence has sometimes been criticized as being atheoretical, and consequently as having no solid substance. Of course, the premise might be true but the consequent false, unless we believe that all truth takes the form of rigorously proved theorems. Artificial intelligence has certainly been short of theorems, and in a field as densely populated with mathematicians and former mathematicians as is computer science, its nakedness in this respect has not gone unnoticed.

It may be objected that 1 am neglecting the A* algorithm, or the various interesting properties of Alpha-Beta search, or even the theorems that Kadane and I have proved about optimal evaluation functions for best-first all-or-none search (Simon & Kadance, 1975). But these isolated examples, even if we add to them all the others known to us, do not constitute a theory of artificial intelligence. At best, they provide us with some islands of theory, separated by wide expanses of an atheoretical ocean. Moreoever, the heuristics of best-first search implied by these examples were known empirically and used in running AI programs for many years before the mathematics was developed (Newell A Simon, 1956).[1].

AI as Empirical Inquiry I am afraid that we must resign ourselves to the fact (or celebrate it, depending on our taste in science) that artificial intelligence has not been a branch of mathematics, but rather a field of inductive, empirical inquiry. The main strategy of investigation has been to propose tasks requiring intelligence for their performance, to write programs for handling those tasks, and to test the efficacy and efficiency of the programs by giving them a sample of tasks drawn from the domain In question. Nearly everything we have learned about artificial intelligence over the past 25 years (and much of what we have learned about human intelligence as well), has been found by following this experimental strategy. And the body of knowledge that exists in AI today Is better described as a store of experimental data and inferences drawn from them than as a collection of mathematical truths.

But the process I am describing corresponds closely to the kind of process that is carried out by AM and BACON. We have seen that both programs are inductive and experimental ~ even if the product of the former's efforts are mathematical constructs and conjectures, and of the letter's, postulated functional relations among numerical variables. Neither AM nor BACON proves anything. If they produce conviction, it is the conviction of the empirical scientist, relying on some postulate of the uniformity of nature, rather than the conviction of the mathematician, relying on the certainty of the laws of logic.

Inferring Principles From Programs The problem-solving tasks that AI research has addressed during the past 25 years, like the tasks addressed by AM and BACON, seem largely fortuitous — targets of opportunity: theorem- proving in logic and group theory, the Eights Puzzle and Missionaries & Cannibals, chess, Euclidean geometry, medical diagnosis, mass spectrogram analysis, speech recognition, parsing natural language, to mention a few. An assiduous historian could no doubt track down the reasons why each of these domains was attempted, but those reasons would not add up to a grand strategy for artificial intelligence. Probably the choice was neither much more nor much less considered than the choice of sweet peas and fruit flies as favored organisms for genetic research.

The true comparison is between these tasks, on the one hand, and the examples generated by AM or the data sets of BACON, on the other. The central inductive problem for AI has been to generalize from the performance of programs dedicated to individual tasks some principles (empirical principles, not necessarily theorems) about the mechansims required for intelligent problem-solving behavior. How successfully this problem has- been solved can be judged by assessing how far new AI programs make use of the heuristics and structural principles of the programs already in existence, and by examining the extent to which AI textbooks are organized in terms of general principles.

On both scores there is evidence of steady progress in AI. In the first decade or two, one can find a number of reinventions of general principles by investigators who were exploring different task domains (or even, occasionally, the same task domain). For example, best-first search apparently appeared intially, as already noted, as a component of one version of The Logic Theory Machine, disappeared in early versions of GPS, which tended to be oriented toward depth-first search, and reappeared in the MATER chess combinations program (Baylor A Simon, 1966). As another example, schemes appear in programs as early as 1956, but were subsequently reinvented and rechristened "templates" or -frames" (Minsky, 1975; and Simon, 1972). During the past five or ten years, however, the main structural components of AI programs have been identified, and a reasonably consistent vocabulary adopted for referring to them.

This gradual progress toward awareness of general principles Is reflected by the textbooks in the field. Early textbooks were little more than collections of

[1] The logic Theory Machine. In 1056,already Incoporated best first heuristic search, white the Alpha-Bate houristic it to be found In cKrat programs at early •• 1956.

examples of more or loss successful problem-solving programs. Beginning with Nilsson's book, Problem-solving Methods In Artificial Intelligence (Nilsson, 1971), some genera) threads of organization began to appear, and specific programs were not merely described, but were analysed for their contributions to these threads. With all this progress, the contemporary books still reflect the pragmatic and empirical foundations of the field, and resemble textbooks in geology more than they resemble treatises in analytical mechanics.

Departures from the Discovery Model There is one respect in which the history of At research departs significantly from the trace of a computer discovery *programs* for in the AI world, many lines of inquiry can be pursued simultaneously — provided that the discipline is sufficiently well populated by researchers, and that the researchers are not too much driven by fads. Hence, when we try to interpret the annals as exemplifying best-first search, we must use that term loosely. To be sure, there was a period of several years during which attempts at theorem- proving nearly dominated AI research, and a more recent period when much of the inquiry was focused on problem-solving in knowledge-rich domains. When a topic like one of these seems to be progressing rapidly, it attratts much of the field's research effort, as would be true of a best-first search system. But other lines of investigation are never wholly dormant.

What has happened when the AI research strategy has departed from the discovery model? The most instructive examples are the cases where pragmatism was sacrificed to the demand for more theory and formal development. One such case is theorem-proving, where mathematical tastes have exercised greater influence than in most other AI task domains.

From the time of Hao Wang's early and successful program for proving theorems in the propositional calculus (Wang, 1960), most theorem-proving efforts have placed great emphasis on the completeness of their programs and upon employing elegant proof methods (e.g., natural deduction and resolution) from symbolic logic. Since completeness Is most easily proved for breadth-first programs that do not use pragmatically constructed selective heuristics, the mainstream of research eschewed best-first search and heuristics that lacked guarantees of completeness.

When heuristics could be used that did not threaten completeness (e.g., set of support), they were adopted readily, but heuristics possessing this guarantee were not in sufficiently long supply to prevent the exponential explosion of search trees. The net result has been a general disillusionment with the progress of theorem-proving research, and a diversion of effort to other task domains within AI. Some exceptions can be found, of course. For example, in the impressive work of Bledsoe and his associates, we see exhibited a much more pragmatic attitude towards heuristics than has been characteristic of theorem-proving research in general (Bledsoe, 1977).

AI as I Residual Domain Some years ago, Allen Newell described artificial intelligence as the domain of weak methods, a description that still seems to hold (Newell, 1969). this Is not because anyone prefers weak methods to strong. No one would solve a problem by heuristic search if he thought that the simplex algorithm of linear programming would do the job. But strong methods apply only in domains that have sufficiently rich and smooth structure to support them. The simplex method works only in a problem space that is convex, bounded by linear inequalities, and with a linear criterion function to be maximized. The method exploits the mathematical structure of the space to home in on solutions in a relatively direct and straightforward fashion.

AM, and to a lesser extent BACON, are designed to work in spaces that have little regular structure, or which have structure that is intiailly unknown to the program. They use the weak methods of heuristic search for the same reason that artificial intelligence has used those methods — because not enough was known, In advance, of the shape of the problem space for stronger methods to be used.

Similarly, attempts to derive measures of computational complexity for typical AI domains have not yet yielded much of a mathematical harvest. Proofs about the dependence of amount of computation, in the worst or average cases, upon problem size depend on knowledge of structural features of the problem domain, and where such structural features are unknown or absent it becomes difficult to obtain strong mathematical results.

Necessity should not be redefined as a virtue. Yet, it makes some sense to define artificial intelligence as a residual domain -- the domain in which it has not yet been possible to substitute powerful special-purpose techniques for weak methods. At any time that such techniques are discovered for a particular subset of problems, those problems are removed from the jurisdiction of AI to that of operations research or numerical analysis. But human intelligence, applied, for instance to the discovery of new knowledge, is not limited to working in orderly domains that have strong structure, and it is the task of AI to show how intelligence works, and even to complement its working, in less well structured domains.

4. From Past to Future

If we take AM and BACON as our models of the discovery process, then we should despair of making exact forecasts of where artificial intelligence research is likely to go in the next few years. For the discovery process illustrated by those programs is myopic, its best-first search responding to intimations of opportunity. Consequently, targets will continue to shift, as they have shifted in the past, to those task domains that exhibit from time to time, most promise of movement.

Allocation of Effort One important difference has already been noted between a discovery process programmed for a serial digital computer and the social discovery process of the AI community. That community is a parallel, rather than a serial, machine. With the increase in manpower that has been attracted to the field in the past five years, the prospects are now brighter than they were earlier for maintaining sustained research activity in a number of AI domains at the same time. At the present time, for example, a more or less continuous effort of several research groups is being devoted to chess programs, to natural language understanding, to visual pattern recognition, to medical diagnosis, and to various kinds of information retrieval tasks.

The fact that a computing system has modest parallel capacity does not, however, invalidate the main features of the best-first search model. The parallel capacity Is still highly limited, and does not grow exponentailly (at least not for long), as it would have to in order to avoid decisions about what part of the tree to search next. The effort allocation problem for a parallel, but not exponentially growing, system is merely a little less poignant than the problem for a strictly serial system. At any given moment, several branches that are most promising for exploration have to be chosen, instead of a single branch. Hence, with limits of both manpower and funding in AI, increased activity in some directions means decreased activity in others.

For example, research on speech recognition appears to have receded again to a relatively low level of activity with the termination of the special ARPA funding, as has AI research on robotry. (I will have a bit more to say about robotry research later, but will simply observe now that the current boom in industrial robotry is only tenuously connected with the main stream of AI research, and makes only limited use of AI methods.) Automatic programming has never reached the level of attention that Its potential importance and centrality to AI would seem to justify. Theorem- proving — and problem-solving in general -~ appear to be attracting relatively little effort currently.

Judged in terms of the contents of the Proceedings of UCAI5 (I don't have the corresponding numbers for the current conference), natural language is attracting the most attention in AI research, followed closely by vision and the representation *and* acquisition of Knowledge. These three areas together accounted for about 607. of all the papers.

The Evaluation function From the shifts in allocation of research effort, we can draw some conclusions about the evaluation function that is used to guide the best-first search. But we must note carefully whose evaluation function it is. To those of us who have been working in AI, it is obvious that the shifts in emphasis among speech recognition, robotry, and automatic programming (these especially, but not exclusively) have been determined to a much greater degree by the judgments of funding agencies as to what kinds of work were more likely to lead promptly to practical application, than by the judgments of the researchers as to what lines of inquiry held the greatest promise for advancing fundamental knowledge.

In part, this vulnerability of the research agenda to genuine or imagined priorities for applications is the price that AI pays for being a "big science" field, dependent for its progress on the availabity of expensive computing equipment. But some other big science fields — for example, radio astronomy — have attained considerable freedom in selecting their research goals, and we can only hope that AI can gradually acquire similar autonomy as the field becomes better established and the fundamental character of the phenomena it studies more widely understood.

If I were to contrast my own personal evaluation function with the function Inferred from the actual present allocation of effort, I would be inclined to give considerably more attention to the domains of robotry (that is, the AI aspects of robotry) and automatic programming than these areas are now receiving. Later, I will have a few words to say about the reasons for my preferences.

Common Themes One factor that mitigates the possible damage done by the whims of funding agencies and the fads of AI research itself, is that there is a considerable overlap in the basic problems encountered, and in the basic AI mechanisms required to solve those problems in all the task domains where AI research is carried on. Best-first search, for example, is a recurring theme, regardless of whether we are concerned with theorem-proving, chess playing, or robot planning. Similar problems of data representation, organization, and access must be faced in almost all task domains. Many tasks call for natural language capabilities of wider or narrower extent. The context-dependence of knowledge acquired through search, and the extrapolation of knowledge from one context to another is a recurrent theme. Because of these commonalities, progress in our understanding of any new task is likely to contribute substantially to progress for other, temporarily dormant tasks.

But these benefits of commonality will be realized only If we pay explicit attention to the transfer problem. The existence of multiple parallel research efforts in different tasks domains increases the danger that the same principles *and* mechanisms will be reinvented, perhaps more than once, by specialized investigators who are unaware of work going on outside their own narrow areas. As the AI research field grows and more investigators enter it, specialization will undoubtedly grow also (it has already), and the dangers of duplication will increase correspondingly.

Perhaps the most important preventive step against reinventing wheels is to define research goals not simply in terms of constructing programs that will perform specific tasks well, but in terms of using programs as examples and test beds for generating and illuminating general principles. Computer science has its roots in both scientific and engineering traditions. For the engineer — at least the nonacademic engineer ~ the device Is the thing; the proof of his pudding is in how well the system he has designed works. For the computer scientist, the device (the program) is not an end in itself, but a means for testing whether particular methods and principles, incorporated in the device, perform the functions for which they are intended. Journal referees and reviewers of funding proposals can contribute much to the development of AI by insisting on these broader goal specifications for AI research projects.

There would also appear to be room in AI research for more generalists and theorists who would devote their attention to extracting general principles by comparative analysis of programs in different task domains. Of course such activity goes on at the present time, but perhaps it would be encouraged further if we did not restrict the term "theory" to formal, mathematical developments.

It might *appear* that I have fallen into a contradiction. Using AM and BACON as my models of discovery programs, t pointed out the futility of trying to predict the course of discovery. Now, only a few paragraphs later, I am expressing my views about the allocation of effort. There is, in fact, no contradiction. In best-first search, choosing *an* evaluation function and using it to guide the allocation of effort is unavoidable. This does not *mean* that one can predict where the search will lead; but a well-choosen evaluation function can indicate the most productive points at which it can start. Let me offer a few illustrations.

Research on Robots One criterion of a promising task domain is that successful AI programs in the domain will rely on important components of intelligence that have not been much explored in other research. Robotry is a promising domain, because it takes us away from planning actions In simple worlds of the Imagination — where the consequences of our actions can be deduced precisely — into planning actions in complex real worlds, where we must be prepared to readjust our estimates of the state of the world repeatedly as our actions fall short of or beside our intentions.

Methods for matching the predicted to the actual state of the world, and for correcting the former to reflect the latter, are fundamental to the success of systems that can survive in complex environments and particularly in environments where there is much uncertainty.

When I refer to robotry research, I have in mind *something rather* different from the development of Industrial robots that is now burgeoning in a number of countries. Most industrial robots are being designed to carry out fairly restricted ranges of tasks in factory environments that are carefully tailored to the robots. Moreover, AI techniques have not played a prominent role in these developments, most of which come out of the tradition of engineering control theory.

In this application, the residual status of AI methods is again apparent. If an environment can be sufficiently smoothed and simplified, then the methods of servomechanism and control theory may provide the best means for designing flexible devices to operate in that environment. AI methods are likely to have a comparative advantage in rough and complex environments that have to be dealt with in their raw, natural form. For this reason, research on vehicles capable of locomoting autonomously on remote planets is probably more relevant to basic issues in artificial intelligence than is research on industrial robots that are to operate in factory environments. The former kinds of systems will have to be flexibly intelligent to a much higher degree than the latter.

However, I do not want to overstate the case. As the development of industrial robots goes forward, there is a need for strong capabilities in visual pattern recognition, a domain in which artificial intelligence concepts are likely to play a role of increasing prominence. The point of my example is that we don't simply want to seize on robotry as a task domain, but want to ask what aspects of robotry call especially for AI approaches, and what light Is likely to be cast on general AI concerns by research focused on those aspects.

Automatic Programming A second domain I singled out as promising for AI research today is automatic programming. Here again, the general value of the research for advancing our basic understanding of artificial intelligence depends on how the problem Is defined, I have especially In mind systems that would take ill-structured and incomplete descriptions of a desired program (of the sort we would give as instructions to a human programmer), and transfer them into executable code. Automatic programming, so defined, is an excellent domain in which to experiment with the automatic design of problem representations — a problem we must address if we are to extend AI further Into ill-structured domains.

An additional reason why automatic programming tasks deserve high priority on the research agenda is that they offer excellent opportunities for work on natural language and knowledge representation. Research in the latter two fields has sometimes suffered from vagueness in the specification of the task. To study natural language effectively we must study particular kinds of situations in which information and meanings have to be communicated for a definite purpose. The automatic programming task defines that purpose (as does also the closely related task of understanding problem instructions written in natural language). By the same token, we are apt to learn most effectively about the problems of knowledge representation in the context of a specific task domain like automatic programming.

If we accept necessity as the mother of invention, we must remember that another parent is needed too. Automatic programming deserves a high rating for its research potential only if there is reason to believe it can be done — that our basic knowledge has reached the point where it is reasonable to talk about automatic design of task representation. I would argue that both the progress ~ modest though it be — that has already been *made* in automatic programming, and the progress in the design of representations for other domains provide favorable indications that we are ready for the next step (Hayes & Simon, 1974).

Local and Global Knowledge A problem that has plagued heuristic search systems from the beginning is that information gathered at one node in a search through a problem space is not generally usable by the system to guide its search in other parts of the space. The same information may have to be generated again and again at different nodes.

Partly, this is a problem of information organization, solvable through such devices as blackboard schemes (Lesser & Erman, 1977). In such schemes, information is not stored in association with the nodes at which it is generated, but is placed in a common space where it becomes permanently available to all parts of the *program,* and at all times during the exploration of the problem space.

But there is a deeper problem with making information more broadly available: the information may be true only in a local context. Then the boundaries of this context must be determined and associated with the information before it can be exported safely. There is still not much theory (or experience) in the AI literature as to how this is to be done, but some progress has been made toward solving the problem in connection with research on speech recognition programs and chess programs, both of which *are* promising environments in which to pursue this issue (Lesser ft Erman, 1977; Perdue ft Berliner, 1977).

Learning Systems In AI a great deal more progress has been made in constructing performance programs than In designing programs that learn. In the early history of artificial intelligence, the topic of self-organizing systems was pursued vigorously but, as it turned out, not particularly successfully. As the best-first search progressed, the nodes associated with this topic received low evaluations, and were gradually abandoned.

Yet the topic of learning in AI is not at all dead; rather it has been redefined. In early efforts, great importance was attached to starting systems off at or near ground level. The guarantee that they were learning was that they started off Knowing almot nothing. Today, we characterize learning in a somewhat different way; wa look for adaptive change, and we look for that change to be recursive and cumulative.

In the broadest sense, any program is a learning program that gradully changes over time so that on each new encounter with a particular kind of task it behaves in a more appropriate way. In neither human beings nor computers should we expect to find just a *very* limited number of processes called "learning processes," for there generally are a multitude of ways in which a complex system can modify itself adaptively.

Learning will generally be incremental. That is, each new step in adaptaation will itself improve the capacity for further adaptation. A problem-solving system becomes a learning system whenever it is designed so that problem solutions can be stored and used to contribute to subsequent problem-solving. Clearly, discovery programs like AM and BACON are learning programs, since their explicit task is to produce novel outputs and to use those outputs recursively.

With this broader definition of learning, a whole spectrum of AI systems qualify as learning systems. Learning can connote all degrees of pasivity or activity of the learner. Thus, at one extreme, we have interactive systems aimed at making it easier for the programmer to add new knowledge to an information structure, where the program itself is a wholly passive learner. At the other extreme, we have adaptive production systems that are able to extract information from their experiences, and use the information to improve themselves even without explicit instruction from outside. Most systems that learn from experience are aided, of course, if the experience is organized for them in a favorable way — in a succession of carefully graded lessons. It is the skill of a good teacher to present experience in this way.

One might ask whether it is time to revive learning as a major explicit goal of AI research. Since learning pervades almost all aspects of Intelligent performance, the right search strategy is probably to incorpooate learning goals In our performance systems. That seems to be a quite natural thing to do in building systems for visual pattern recognition, for example, for automatic programming, or for understanding natural language instructions.

But there *are* apprentices in the world as well as Journeyman; and presumably the apprentice's first concern in his learning rather that his performance. So perhaps there is room, on the tree of AI research, for an active branch that works with tasks In which learning and adaptation are the central concerns. Considering the recent rapid progress that has been made in constructing adaptive production systems, good progress can be anticipated along that branch, and I would assign it a rather favorable evaluation. But the fact that some investigators specialize in learning processes should not deter the rest of us from experimenting with learning components in our performance systems.

5. Conclusion

In this paper I have reviewed the AI community as if it were a medium-size slightly parallel processor searching its way in inductive, best-first fashion through the problem space of intelligent action. I have compared it with some of the existing AI programs that best characterize the discovery process. The comparison does not yield any great surprises, but perhaps provides some reassurance.

As a typical example of a discovery program, the AI community uses weak methods uder the guidance of a somewhat imprecise evaluation function and vague ultimate goals. It tries to discover the mechanisms that enable a system like the human mind to behave purposefully, adaptively, and sometimes even effectively over a wide range of difficult and ill-structured tasks.

The search is highly pragmatic, steered and redirected by concrete empirical evidence culled from experiments with programs operating in an accidentally determined collection of task environments. The output of the research is mostly encapsulated in heuristics, not yet formalized in coherent theories of broad scope. All is confusion *and* mild chaos, as it should be at an exciting frontier of fundamental scientific inquiry. Although only a quarter of a century old, the search has already yielded a solid body of empirical knowledge about the nature of Intelligence and the means of capturing it in programs.

6. References

Baylor, G.W, ft Simon, MA A chess mating combinations program. AFIPS Conference Proceedings, 1966 Spring Joint Computer Conference, Boston, April 26-28, 1966, 28, 431-447. Washington, DC: Spartan Books.

Bledsoe, W.W. Non-resolution theorem proving. Artificial Intelligence, 1977, 9 (1), 1-35.

Hayes, J.R., ft Simon, H.A. Understanding written problem instructions. In L.W. Gregg (Ed.), Knowledge and cognition. Potomac, MO: Lawrence Eribaum Associates, 1974.

Langley, P. Rediscovering physics with BAC0N.3. Proceedings of the 6th International Joint Conference on Artificial Intelligence, forthcoming.

Lenat, D.B. The ubiwuity of discovery. Proceedings of the 5th International Joint Conference on Artificial Intelligence. 1977, i, 1093-1105.

Lesser, V.R., ft Erman, L.D. A retrospective view of the HEARSAY-II architecture. Proceedings of the 5th International Joint Conference on Artificial Intelligence. 1977, 2 790-800.

Minsky, M. A framework for representing knowledge. In P.H. Winston (Ed.), The. psychology of computer vision. NY: McGraw-Hill, 1975.

Newell, A. Heuristic programming: Ill-structured problems. In J. Aronofsky (Ed.), Progress in operations research. Vol. III. NY: Wiley, 1969.

Newell, A., ft Simon, H.A. Computer science as empirical inquiry: Symbols and search. Communications of the, ACM. 1976,12, 111-126.

Newell, A., ft Simon, HA, The logic theory machine. IRE Transactions on Information Theory. Vol. IT-2, No. 3, Sept. 1956.

Nilsson, NJ. Problem-solving methods in artificial intelligence. NY: McGraw-Hill, 1971.

Simon, HA Artificial intelligence systems that understand. Proceedings of the 5th International Joint Conference on ArtificiaTIntelligence. 1977, g, 1059-1073.

Simon, H.A. The heuristic compiler. In H.A. Simon ft L. Siklossy (Ed.), Representation and meaning: Experiments with information processing systems. Englewood Cliffs, NJ: Prentice-Hall, 1972.

Simon, H.A., ft Kadane, J.B. Optimal problem-solving search: All-or-none solutions. Artificial Intelligence. 1975, §,235-248.

Simon, H.A., A Newell, A. Heuristic problem solving: The next advance in operations research. Operations Research. 1958, 6, 1-10.

Wang, H Toward mechanical mathematics. IBM Journal fit Research and Development. January 1960, 2-22.

Skill of Intelligent Robot

Kunikatsu Takase
Electrotechnical Laboratory
Tokyo, Japan

ABSTRACT

This paper reviews the development of the intelligent robot and separates skills from high level intelligence. It demonstrates that skills can be represented as virtual mechanisms programmed in software. Virtual mechanisms are defined by controlling both motion and force of a robot arm in a task related cartesian coordinate system. By adding monitoring, a skillful robot system can be built. The skill of an intelligent robot is accumulated in the form of task - particular knowledge that is a specification of the mechanism.

1. INTRODUCTION

Various intelligent robots have been developed by combining computer decision making with mechanical mechanisms, especially with robot arms. In this field the efforts have been mainly directed to developing high level intelligence system such as for solving puzzles, interpreting drawings or recognizing three-dimensional objects. There has been little study of low level intelligence relating to skill or dexterity of a robot arm. In order to make the motion of a robot arm smoother and more adaptive, a series of studies - kinematics, trajectory calculation, sensory feedback of a robot arm - were undertaken in the Artificial Intelligence Project at Stanford University. But in general we are at the dawn of the study of the skill of robot arms.

The practical study of assembly automation which started as one of the applications of intelligent robots is developing as an independent area with remarkable results. Reliable assembly systems are being released one after another. Although those are comparatively special-purpose low intelligence systems, their skills are far more dexterious than those of intelligent robots. It seems the increased skill of the intelligent robot could be obtained only through the integration of task-particular knowledge that is normally obtained, for example, during the development of special purpose systems. The "Move" command in AL, for example, and its modification are not sufficient for specifying the dexterious actions of robot arms. "Composing virtual mechanisms by software" seems to be the best control structure for the purpose.

Recently the performance of arithmetic LSI has improved remarkably enabling us to carry out the exact calculation of dynamic models of a robot arm in the servo cycle, and to compose virtual mechanisms in software.

2. INTELLIGENT ROBOTS

In the beginning of 1970*s several Hand-Eye systems had been developed at universities and research laboratories in the world. Those robotic systems consisted of a camera and arm with 6 degrees of freedom and computer. At Stanford University a robot which could successfully solve the "instant insanity" puzzle was developed C1D. In this puzzle four cubes with different colored faces must be stacked up so that no two similar color appear on any sides. A robot made at Hitachi could understand simple drawings and build block structures as specified in the drawings C23. At the Electrotechnical Laboratory a robot was developed, which could insert a beam into a box with small clearance using visual feedback [3]. In these robotic system intelligence had been applied to recognizing a 3-dimensional pattern, understanding a drawing, or solving puzzles.

The author has been a visiting scientist of Electrical Engineering at Purdue University, West Lafayette, IN, since August 1978 until August 1979.

In order to realize computer control of an arm, several elements had been studied: arm design, servo system, transformation between joint coordinates and cartesian coordinates, computer interface, and robot operating system. Although a good deal of effort was devoted to the computer control of the arm, the arm only could perform "pick and place" operations, that is, to pick up an object, to transfer it and then to place it. Some attempts were made to make the a robot arm more dexterious, for example, by adding tactile sensors in order to grasp objects more dexteriously. None of those attempts was, however, successful in developing a reliable elements to be incorporated in the robotic system.

It was in the Assembly Automation Systems developed at Stanford University (later referred to as the AL System) that an intelligent robot was first able to act in the practical world, emerging from the stage of "play" in the block world C5,6,7D. Assembly processes, such as parts mating and inserting, required adaptive object handling capabilities found only in a human workers were demonstrated. These tasks could not be performed by conventional hard automation machines.

In many cases of low volume assembly production programmability was the key factor to enhance the utility of the assembly system. In the AL system much attention paid to the design of task description language. AL offered the programming structure in which a user could describe an assembly procedure in terms of multilevel commands ranging from the simple "move" command to task-level commands [4]. High-level assembly statements would be expanded into the sequence of move-level commands in a general manner.

In an assembly system robot arms have to be skillful enough to be able to carry out parts mating and other adaptive motions. In the AL system, precise motions were controlled, based on trajectory calculation and the modification of motions based upon force feedback or touch feedback to adapt to the outer world. To integrate these functions into the system, software servoing was used, where a digital computer controlled each motor drive level directly. By using such a control scheme, the capability of the robot arm was remarkably improved.

3. PRACTICAL ASSEMBLY AUTOMATION

While general assembly automation was being studied at Stanford University, other practical research was started at several other laboratories and companies. A system, which consisted of position controlled robots and various specially designed tools was developed at Kawasaki Unimate to assemble a gasoline engine. The work demonstrated a practical approach to assembly automation. Hitachi developed a precise insertion robot which operated reliably based on active accommodation by a flexible force sensing mechanism C81. A position controlled robot equipped with a compliance mechanism (RCC) at the wrist which could perform precise insertion quickly without force feedback was developed at Draper Laboratory C93. At SRI a system which could carry out assembly using a passive accommodation table had been developed C103. In these practical system, global motion was controlled by simple positioning robots and fine accommodation was accomplished by tools specially designed for the tasks.

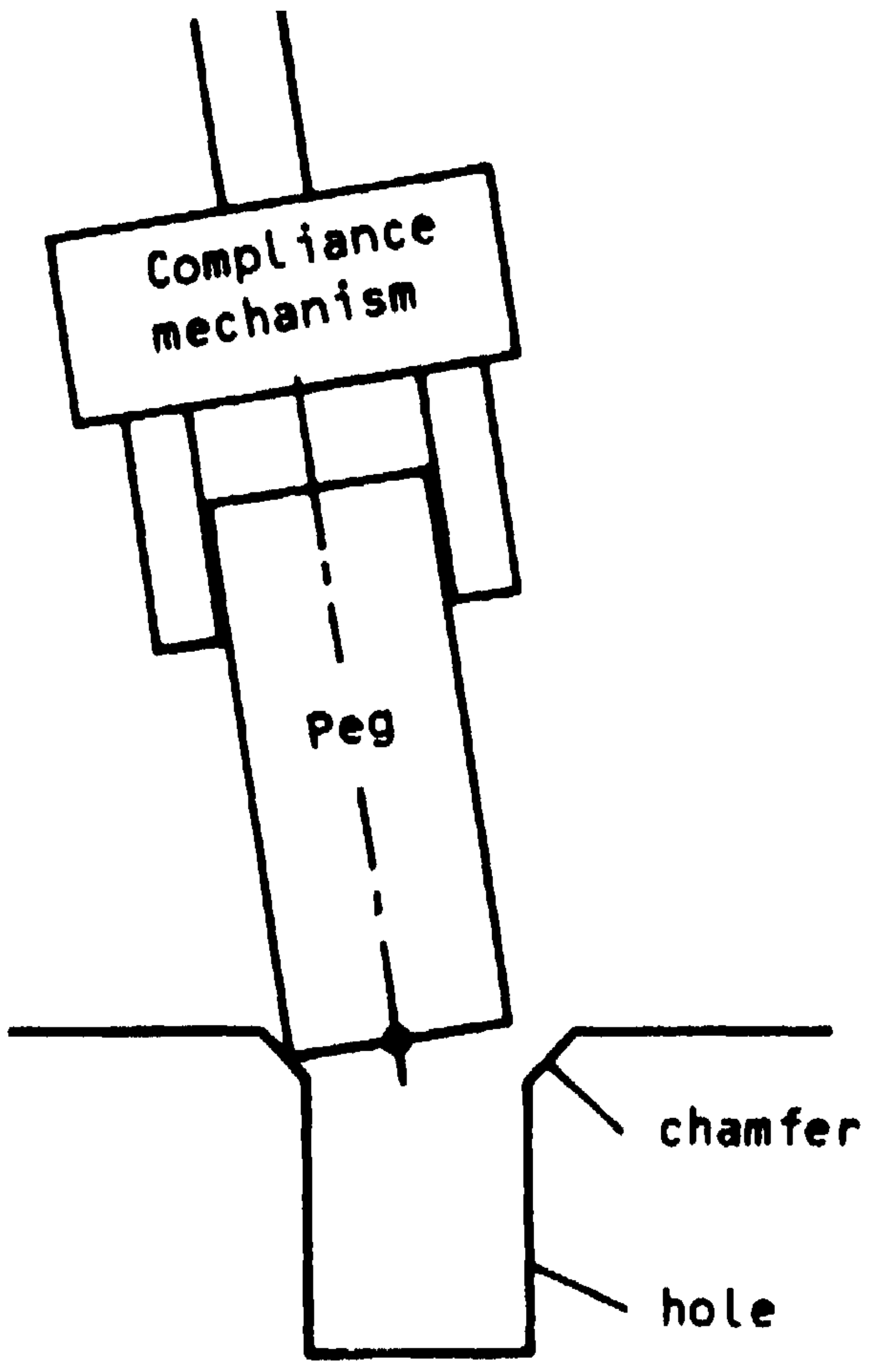


Fig. 1 Peg insertion by compliance mechanism

It would be instructive to compare the operation of a practical system with that of an intelligent robot. We will take peg insertion as an example. The robot attached with a RCC at the wrist will move a peg into a hole. As shown in Fig. 1, if there is a misalignment between the peg and the hole, the peg will be guided to the hole by the chamfer., The RCC is a compliance mechanism which provides the accommodation. If a lateral force is exerted at the tip of the peg, it will translate without rotation, and if a torque is exerted at the tip, it will rotate without translation. With the help of the chamfer any translation error

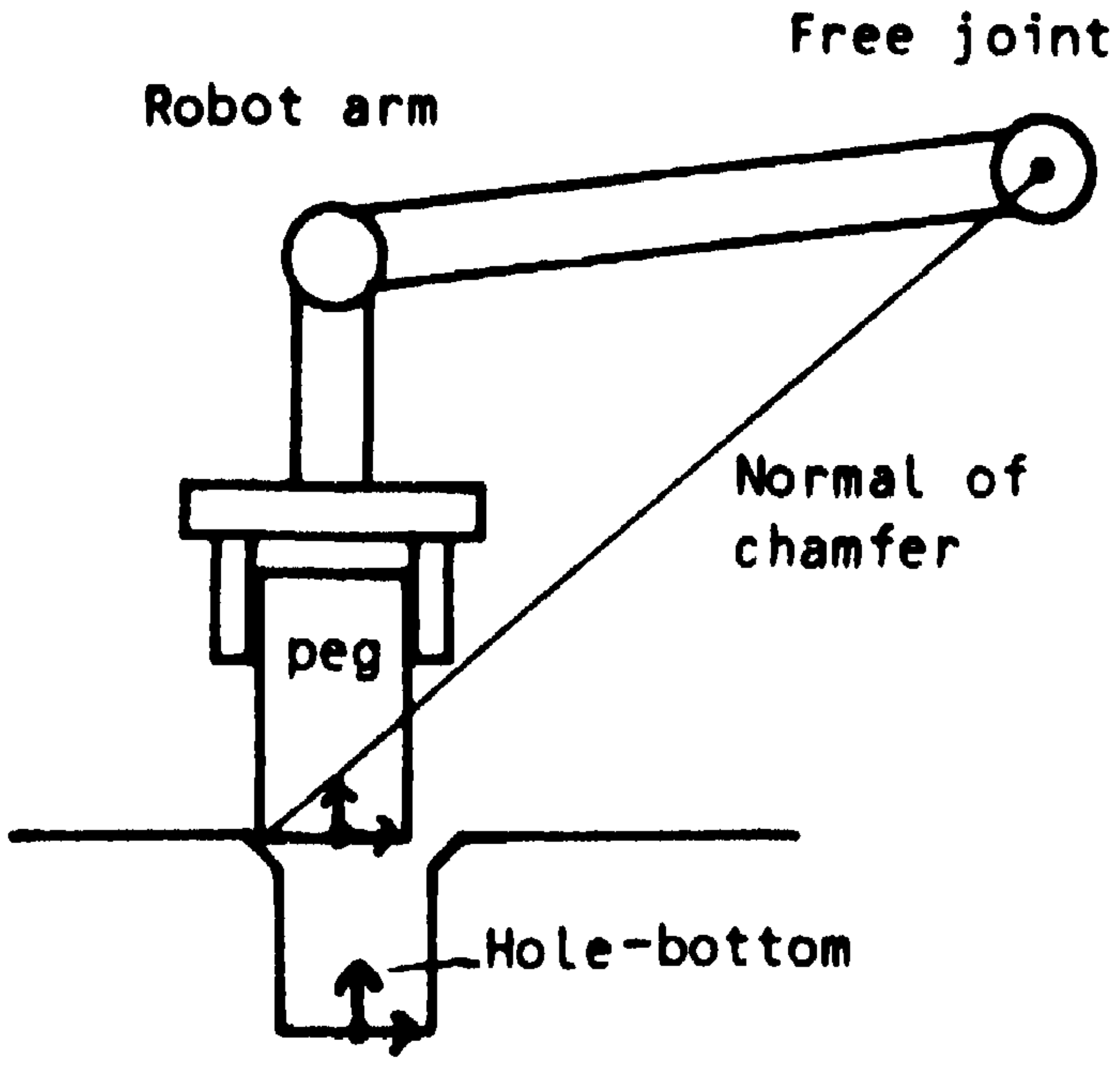


Fig. 2 Peg insertion by AL system

is corrected- During the insertion process into the hole any rotation error is corrected.

In the AL system the peg insertion could be described as:

```
MOVE peg TO hole-bottom
   WITH FORCE = 0 ALONG X,Y OF hole-bottom
```

This command means "move coordinate frame peg to hole-bottom with two proper joints free so that the x and y components of force are zero." With this method the peg insertion sometimes becomes impossible. For example, if a free joint lies close by the normal of the chamfer as shown in Fig. 2, the angle between chamfer tangent and free surface becomes less then arctangent of friction coefficient. The peg will be locked. This is indeed a probable case. This disadvantage of the AL system shows not only that the system can not perform the peg insertion in this way, but also that the system lacks the systematic representation capability of the necessary skill. It is undesirable that the operation of a robot arm is dependent on its configuration. A user would like to write machine independent programs. In general a robot arm skill consists of patterns of action to the task environment, interpretation of results, and the decision of further action. We have to represent the knowledge about an action and its result, which is task particular, so that it is robot arm independent.

4. BASIC CONTROL SCHEME FOR SKILLFUL ROBOT ARM

4.1 Control in Cartesian Coordinate Systems

If the form of assembly is defined, it is most efficient to use the special devices or tools. Not only in assembly, but in general, it is most efficient to design mechanisms suited for the task, and to control the operation of these mechanism using sensory feedback. Hitachi, Draper Laboratory, and SRI had taken this approach in developing assembly systems. Therefore it is senseless to simply compare the capabilities of special-purpose systems and intelligent robots in which generalities are most important. The skill representation capability of the intelligent robot should also be general so as to provide the skill of special-purpose tools. If we could build up virtual mechanisms by software, we would be able to utilize all task particular knowledge which is obtained from the study of special-purpose tools. And we will be able to exchange tools and skills by software.

What kind of robot arm control would be required in order to build up virtual mechanisms by software? It is motion and force control in cartesian coordinate systems. A few methods have been proposed to realize such a type of control [11,12]. We will describe here the direct servoing method in cartesian coordinates that seems to have the highest generality. As shown in Fig. 3 this system provides for the real time transformation between joint coordinates and cartesian coordinates, the transformation of acceleration and force in cartesian coordinates into joint torques. In addition, coliolis force, centrifugal force and gravity loading force are canceled in real-time.

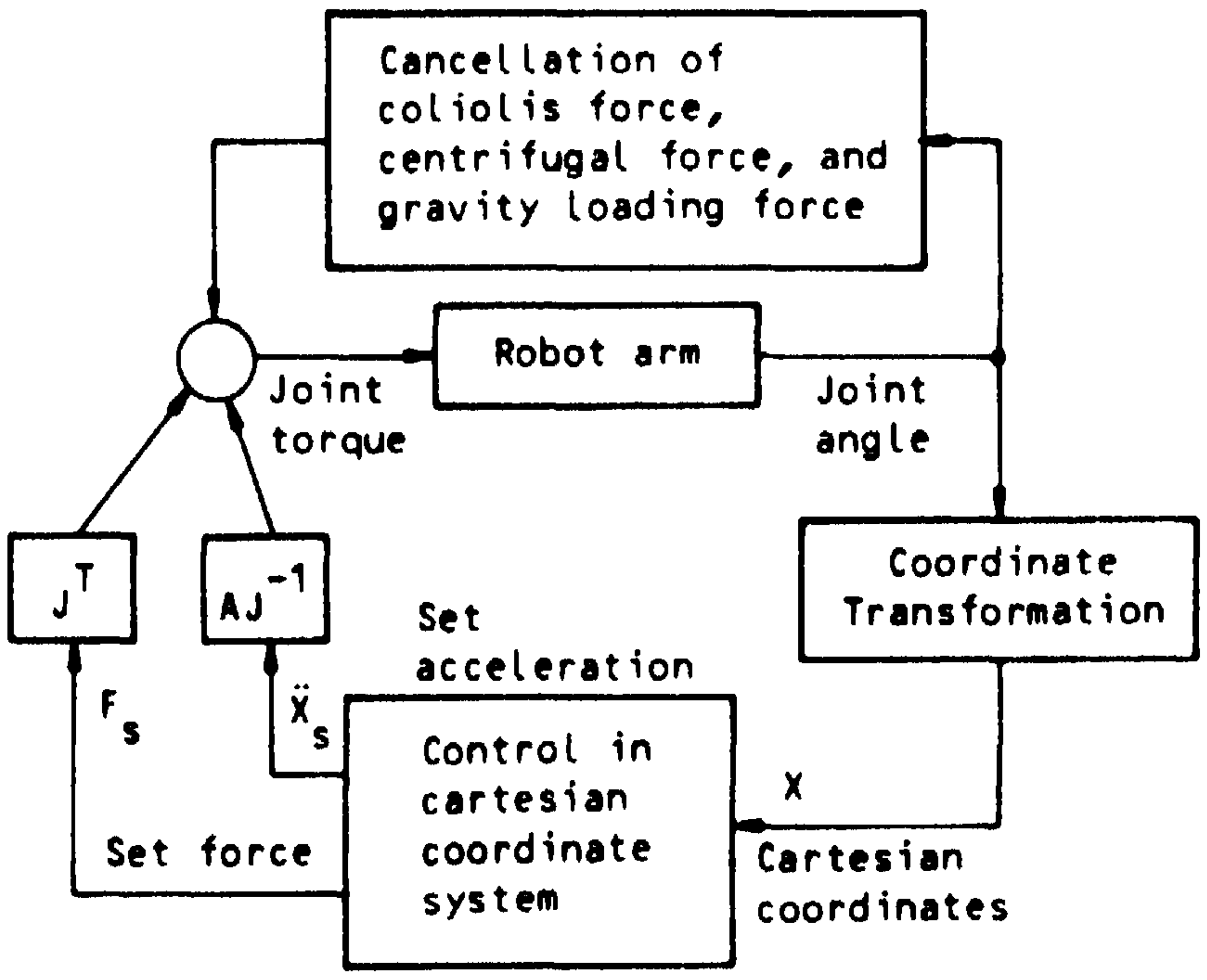

Cancellation of coliolis force, centrifugal force, and gravity loading force
Robot arm
Joint torque
Joint angle
J^T
AJ^{-1}
Set acceleration
Coordinate Transformation
F_s
$\ddot{X}_s$
Control in cartesian coordinate system
X
Set force
Cartesian coordinates

Fig. 3 Servo system for arm control in cartesian coordinate system

4.2 Real-time Computation

As the dynamic model of a robot arm is very complicated it has been considered impossible to compute joint torques in real-time. However, recent improvement both in calculation algorithms and in arithmetic elements has now made it realistic. Dynamic model in matrix representation, while simple, was *Very* time consuming to compute. Bejczy developed an approximation model which was very fast to compute. The author formulated the equations of motion in a vector representation. Walker devised a recursive algorithm for evaluating the model in vector representation without redundant repetitive calculation [13]. As shown in Table, we can now calculate joint torques with this algorithm about 200 times faster than using the matrix method.

Table: List of time to compute joint forces and moment on the PDP 11/45 (Reproduced from [13])

Model	Language	Computation time
Matrix Model	Fortran	7.9 sec
Rejczy	Fortran	0.0025 sec
Walker	Fortran	0.0335 sec
Walker	Floating Point Assembler	0.0033 sec

Powerful arithmetic elements such as 16^{bit} x 16^{bit} LSI multiplier with 200 nano second execution time, have also become available. It is not difficult to develop special-purpose processors with these arithmetic elements for performing fast vector calculation such as dot product or cross product. It would also be possible to make the calculation speed shown in Table ten times as fast. Above considerations show that sophisticated calculation for cartesian coordinates control could easily be carried out in less than 10 milLi seconds.

4.3 Torque Controlled Robot Arm

Reliable force sensing systems have not been obtained yet, although force sensing, especially wrist force sensors have been actively studied. At the present stage it would be impossible to incorporate the force sensors into the arm control system controlling motion and force in cartesian coordinates. The author and co-workers have developed a robot arm driven by magnetic powder clutches and have