EXPLORATION SYSTEMS AND SYNTACTIC PROCESSES

By Marvin Minsky

A "syntactic process" is a formal system (with a set of "atomic" or "kernel" sentences and a set of transformations which with the atomic sentences generate other sentences), together with a process which controls the sequence in which new sentences are generated. That is, a process whose past history determines, in a specified way, which transformation is to be next applied to which expression(s). The process itself is not here regarded as "formal;" its behavior is involved with semantic and, frequently, empirical contingencies.

In many artificial intelligence situations the syntactic process may involve in an essential way what may be called an "exploration graph." "Problem solving" seems to usually involve such a structure. Other parts of the process may involve the use of special procedures such as "interpretations" and "examples" which may use special computers or "models." The ideas of "memory," design and performance of "empirical experiments," and computation of "characters" and "abstractions" are likely to be required. And a program for self-improvement will likely be part of any important intelligence.

These various divisions might reflect actual divisions of the machine program into parts, or even separations of hardware, or they might be just a heuristic way of thinking about a relatively interpreted operation. By the use of the term "syntactic" we wish to indicate that the emphasis is on programs in whose operation expressions are used in a manner more or less suggestive of a humanoid language.

Two topics will be examined here. The first topic will be a discussion of the exploration process for problem-solving machines. The second topic is a discussion of possible ways to construct the syntax of a (meta)language so that a formal deductive system can be used, together with a semantic, and a set of empirical verification methods. We will in fact be talking about a (meta)language which in turn can talk about a logic plus a set of interpretations, plus non-logically derived properties of the models on which the interpretations are grounded.

The sentences of the metalanguage, as I see them, will often consist of (i) a quote of a sentence in the logic, plus (ii), a string denoting how this sentence is to be interpreted in what model, plus (iii), some strings denoting the present estimates of the validity of the interpreted sentence -- e.g., logical validity, empirical credibility, consistency with other valid propositions in memory, etc. It ought to be possible also to make general statements in the metalanguage, e.g., about predicates of the object language.

The expressions of the metalanguage are subject to a set of transforms, its syntactical rules. It is these that are under the control of the basic syntactic process of the machine. The metalanguage should be rich enough to discuss the structure of the syntactic process itself, when a description of the latter is presented to the machine in the form of a model and a set of expressions interpreted in the model. In this case the machine could be given the self-improvement problem.

1.1 Exploration systems. The following is a provisional hierarchy of parts of the machine program:

(i) Specific mechanical subroutines. E.g.,
- -- Application of a given rule of inference to one or more given expressions in a formal system.
- -- Computation of a "character" of an expression (assuming there is a given program for this).
- -- Memory search for expression of given character.
- -- Performing of definite action on an environment, and recording (in given way) result.
- -- "Set up, interpret, and run a given model," assuming this to be a precisely programmed procedure.

(ii) The syntax. This is a set of rules concerning use of the subroutines of (i). These include both
(iia) Obligatory rules, like, e.g.,
- -- "Don't use logically false propositions as premises!"
- -- "Don't subject logically false propositions to 'em-

pirical" test." (unless you are programming a test of the consistency of a trial interpretation!

-- "Don't use transformation BX 103 J on an expression unless it has been reduced to canonical form CF 125 b or has passed test T 35 j 67 N.

(iib) Elective rules, e.g.,

-- Transform T 64 may be applied to any expression of form F ... . (Logical rules of inference, etc.)

-- Etc.

(iii) The exploration system. This is the part of the process which directs application of the transformations (subroutines). This system is designed with several aspects of the problem-solving problem in view, e.g.,

-- Make efficient use of time.

-- Make efficient use of memory.

-- Explore the possible avenues in an orderly and efficient manner.

-- Organize the history of the process so that the relation of the immediate problem to the original problem is always clear.

(iv) Finally there is a "valuation program" which reviews the whole operation and decides what to learn from it -- e.g.,

-- What to put in permanent memory.

-- Modifications in the syntax.

-- Modifications in the exploration system.

In "A Framework for an Artificial Intelligence," examples at each level are: (i) Computation of characters, Application of methods, etc.; (ii) Contents of Character-Method box is syntax; (iii) (last few pages); (iv) Way in which matrices are "learned," i.e., changes in the "character model."

1.2 "Path" Problems and their "validity graphs."

A "path" problem is one in which expressions are considered

as vertices in a graph: Two vertices are given: "begin," "end." One has at his disposal a set of subroutines which divide into three classes.

I. Subroutines which generate a new vertex between two others.

II. Subroutines which "validate" a direct link between two vertices.

III. Subroutines which make an estimate of "How difficult will it be to validate a (given) link?" (A link is also "validated" when its end points are connected by a chain of validated links.)

The "path" problem is solved when there is a valid chain between "begin" and "end." More generally, II can assign "degrees of validity" to a link. The problem can be considered solved when some given condition concerning the degrees of validity on a graph is met.

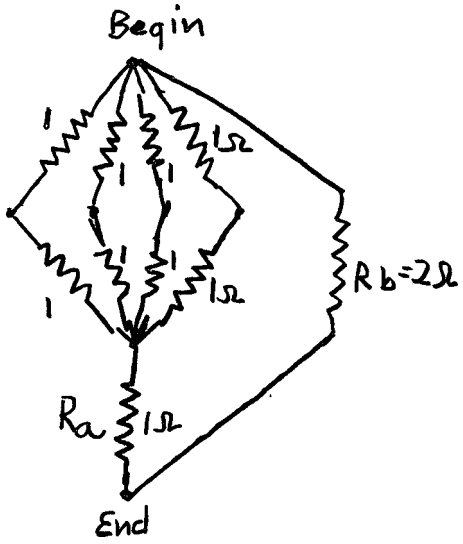
The exploration problem is this: What is the best link to work on next? This question has to be answered before one can start to worry about what method to try next!

1.2.1 The "minimal path" method. Assume that for each link in the graph we can get, by use of methods of type III, an estimate of the difficulty $D(v_i,v_j)$ to be expected in validating the link.

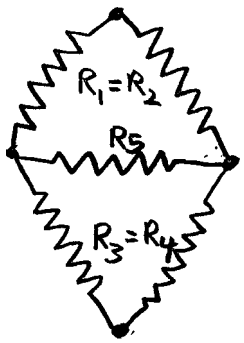
Now find a path from "begin" to "end" with minimum sum. We have then to choose a link somewhere along this path.

— If you have an additional estimate of your confidence in the estimate of $D(v_i,v_j)$, one should then choose the worst link in this respect. This would help insure against bad luck. Even so, this minimal path method doesn't use very much of the structure of the graph.

1.2.2 The "current" model. Replace each link by a resistor whose value corresponds to the difficulty estimate. Then apply a potential across the terminals of the graph and choose that unestablished link in which the highest current flows. Consider the following case:



The "minimal path" algorithm would choose R_b because it represents the minimal path. The "current" model chooses R_a because it has maximal effect on all paths in a certain weighted sense. One might rationalize this choice in several ways. Perhaps one might say "If I work on R_b and fail to establish it, my effort will have been wasted. I have a better chance of succeeding on R_a; it will not in itself solve the problem, but with so many alternate possible ways to complete the problem, I can be reasonably sure the work won't be wasted." This argument isn't very satisfactory; and for some interpretations of the difficulty estimate it is quite bad. For instance, if the numbers represent <u>reliable</u> estimates of how long it will take to verify a link, then the minimal path choice would certainly be better. In the example below, the current model itself rejects an argument of that form:

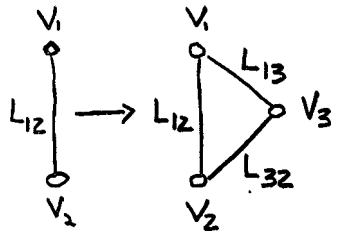


Here one might be tempted to say: "There is no reason to prefer any one of R_1, R_2, R_3, R_4. And validation of R_5 will represent a form of insurance, since it would increase the chance that validation of any other link will be part of a solution. In fact, it will almost double the utility of future work on the others." But this conservative insurance-buying is really incorrect: validation of R_5 won't simplify the next decision, and there is a chance of a solution being found that doesn't involve R_5. Work will automatically be done on R_5 if the need for it arises. [The idea of the current model was suggested by one of Shannon's game machines.]*

*See note at end of paper.

1.2.3. In any case, the choice of link will depend on the value of the "difficulty" estimate for links. The difficulty estimate of a link ought to be dependent on the amount of time already wasted, and the "power" of the methods which have already failed there. The exploration system will servo itself if the "difficulty" of a link goes up with wasted time, and with the failure of methods.

When a new vertex is inserted, the old link is not removed, but a record is kept of methods tried on it so far, and the time spent. This data is used to revise the difficulty estimate. At each step, several such revisions might be appropriate.



1.2.4. Once a link is chosen, a subroutine of type II (see 1.2) is chosen and applied. This choice could be dictated by a "character-method" system such as I described earlier. The subroutine itself might involve a (more or less) brief exploration process. If type II direct validation fails, then a type I (subgoal generating) transform could be tried. These decisions should all be handled by the character-method machine. As indicated in the earlier paper, that machine can use the same data to obtain an a priori difficulty estimate. The exploration graph machine combines the a priori estimate with its data on wasted time and methods wasted to obtain its pro tem difficulty estimate.

1.3 Other exploration systems.

The "path problem" with its "validity graph" is a description of a special type of exploration system, which I think is not too far from certain ways we do things, particularly in logical and mathematical reasoning. For combinatorial games, an exploration of "minimax" branches of a "move tree" with terminal positional evaluations might lead to a somewhat different structure. I think that the path formulation will be adequate in what follows here because the syntactic process

to be described will treat all problems in more or less the same manner, namely, as we treat logical problems. This comes about through the devices of:

(i) generalizing "yes-no" logical validity to a "degree of validity", plus assuming that we can work out a reasonably effective method for combining these "degrees" for a chain of "inferences."

(ii) syntactically treating all methods of confirmation of a proposition (that is, confirmation up to some "degree of validity") in the same way as we treat logical rules of inference. Thus it might be said that some "modal" logic is assumed.

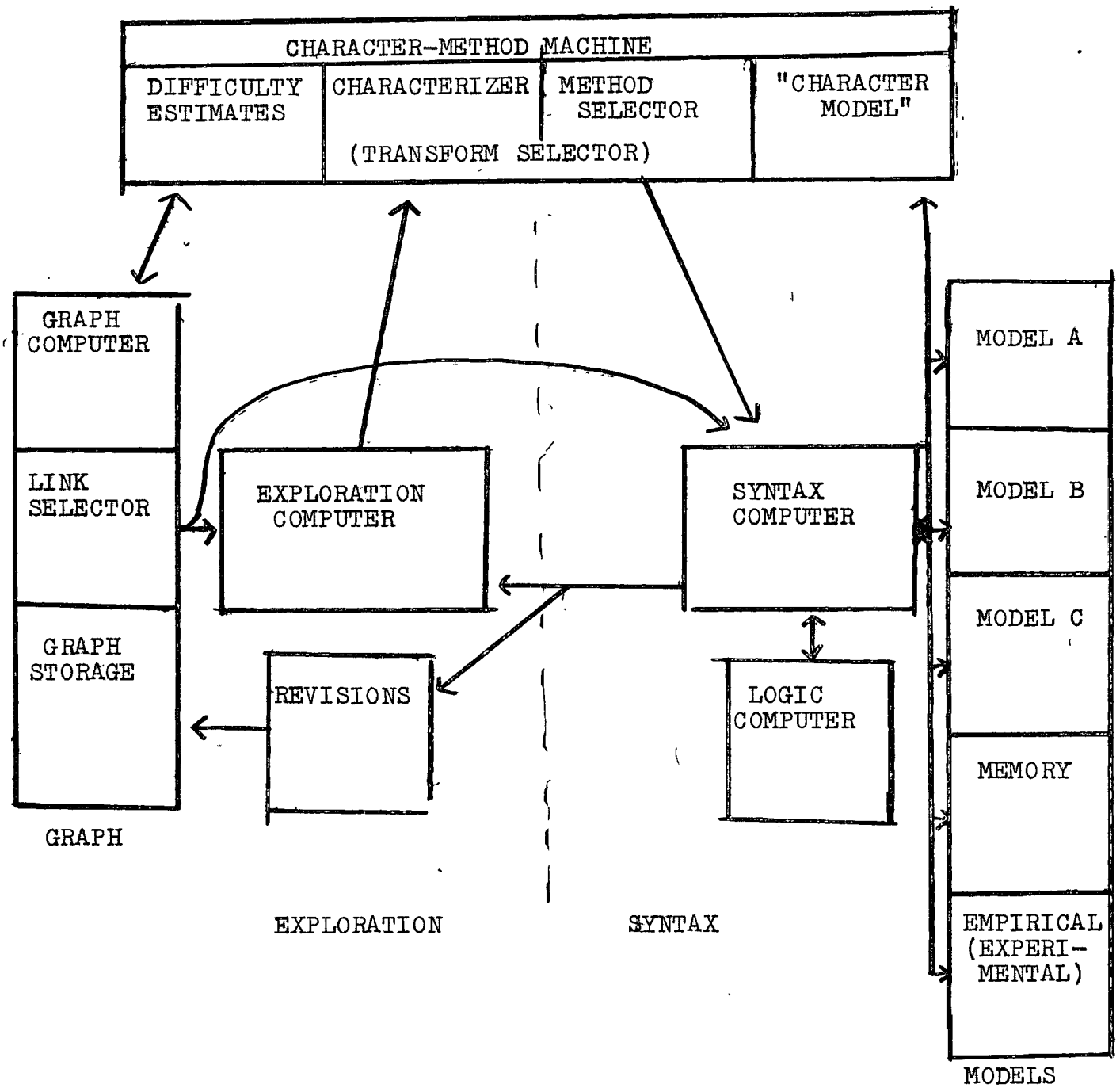
2.0 The Syntactic Process Machine.

The syntactic process itself consists of a series in each term of which some "transformation" is applied to one group of sentences to obtain another, this sequence being directed by the exploration process. In a Logic machine application, the transformations might simply be the rules of inference for the logic. For more general applications I propose that in the syntax language, practically all operations by which the machine can generate new sentences from old have the same linguistic status. The following is a list of kinds of transformations that might be among those one might desire:

2.1 Transformations that might be performed by specific subroutines within the syntax computer.

-- Apply grammatical transformation G_i to expression E_j.
-- Apply rule of inference I_i to expressions $E_1 \ldots E_n$.
-- Modify "validity coefficient" of E_i by rule R_j.
-- Answer questions (i.e., if answer is "yes" replace the question by an affirmative declaration). E.g., "Does S_o follow from $S_1, \ldots, S_n$ by an application of one of the rules of inference?"

(The last question can be answered by a serial application of specific subroutines for inference followed by a conditional application of the grammatical rule which replaces a question


CHARACTER-METHOD MACHINE
DIFFICULTY ESTIMATES
CHARACTERIZER
(TRANSFORM SELECTOR)
METHOD SELECTOR
"CHARACTER MODEL"
GRAPH COMPUTER
LINK SELECTOR
GRAPH STORAGE
GRAPH
EXPLORATION COMPUTER
REVISIONS
EXPLORATION
SYNTAX COMPUTER
LOGIC COMPUTER
SYNTAX
MODEL A
MODEL B
MODEL C
MEMORY
EMPIRICAL (EXPERI-MENTAL)
MODELS

by its affirmative, or by its declarative denial. For syntactical purposes this can be regarded as a single transformation.) In the machine language, the grammatical transformations can have a somewhat simpler form than in English. Each sentence could be a simple declarative followed by a string of marks which indicate whether it is a question, an imperative, a declarative, etc. Also its logical status, pro tem degrees of validity, etc., would be carried along, and these "characters" of the sentence would also be operated on by the transformations.

2.2 Some of the transformations will not be realized within the syntax computer proper, but will be performed in auxiliary computers. These, however, will be under the control of the syntax computer, through certain obligatory transformations which may be performed either within the syntax computer or in the auxiliaries: Let $C_1,\ldots,C_n$ be the auxiliary computers.

-- Apply C_k to expression E_j in manner R_i.
I.e., code E_j in suitable form, and
Select, or prepare program for C_k, and
Run the program through C_i, and
Uncode the result to form a grammatical sentence for the syntax computer.

-- "Evaluate plausibility of E by test in standard example S."

-- "Evaluate consistency with axioms by test in given finite model."

-- "Find sentences in memory which have similar characters, assuming given character criteria." (Use Character model and memory.)

-- "Test empirical credibility by standard method." We assume that the transformation refers to some known subroutine by which an experiment can be designed and performed. The result, when decoded for the syntax computer, will be a new sentence. In the machine language, such an "experimental" subroutine can be called for by specifying a single syntactical transformation. It would differ from the logical transformations in that the resulting sen-

tence would have an empirical "validity-label" rather than a logical one. (It would also differ in that the result of the transformation would not be uniquely defined by the input sentences. In this respect the language is not simply equivalent to a formal system.)

2.3 Again, using the auxiliary machines one can have transformations which involve sequences of specific routines.
-- "Does E fit into one of the standard models?"
-- "Does E represent (in one of the standard interpretations of its terms) a proposition I can answer empirically?"

Now such subroutines may involve a bit of exploration of their own. One way to program them that might be convenient linguistically, would be through the use of various types of imperatives. An imperative could be a sentence (or a question) plus a mark indicating its importance -- i.e., the force with which it is authorized to tie up the whole machine.

For example, the exploration system may decide that a certain link is worth a certain exploration time. This sub-exploration can be performed by the character-method machine which will continue to provide trial methods in estimated order of merit. The time-force of the expression "on the floor" is reduced during the search and when it loses all its imperative force control reverts back to the exploration system. Thus, the syntax computer may receive an order, "Verify E," from the exploration system, with a time limit. During this time the syntax computer program performs an exploration process, but its exploration system is just the character matrix machine, with the graph generated by the sequence of methods recommended by that device. (Some of the methods, in turn, may involve an exploration.)

3. The interpretation of expressions in "auxiliary computers," "models" and "examples."

It would be very difficult to define precisely a reasonably small set of "models" plus "interpretations" such that they would automatically provide "examples" for a wide class

of problems for an intelligent machine. I think, however, that it might not be quite so difficult to provide a program for the construction of models for each special case, with a way of constructing a flexible "approximate interpretation." The basic idea is that of "feedback over interpretations," using an "inconsistency actuated feedback"! I also expect that a similar idea can be applied to the syntax computer itself; namely, the logic used in the syntax computer need not be consistent. In fact, it might be beneficial to use a rich humanoid language, PLUS a servo system which discourages the use of syntactical transformations in those situations in which they lead to paradoxes. Detection of this type of situation would not appear to me to be essentially different from other functions performed by the character machine, and it is certainly very much like what is done by humans with respect to the set-theory paradoxes. We learn what kinds of sentences are dangerous.

3.1 "Locally consistent interpretations."

3.1.1 Let us consider the case of a high school student attempting to prove a theorem in plane geometry. He has been given a set of axioms and is presumed to know how to apply rules of logical inference so as to obtain deductions and recognize proofs. Now the teacher and the student can both pretend that they are operating in a formal system and that the diagrams are heuristic devices they could, in principle, do without. But the fact is that most if not all individuals make use of diagrams in finding proofs, either on paper or "in their head."

A "heuristic" diagram, I claim, represents a certain kind of "interpretation" of the formal problem. We must note that a diagram of a problem in plane geometry, while in a sense serving as an "interpretation of the problem," cannot be a legitimate "interpretation of the formal system," or even of some of the expressions in the system, unless a great many ad hoc provisions are made. For one thing, the

diagram is confined to a finite part of a plane, and in this finite portion one can find non-intersecting lines which are not consistent with the metric conditions on parallel lines.

What one does with these diagrams is to find an interpretation which is not, in fact, supposed to be consistent with the axioms or any large part of the system of plane geometry. Instead, one looks for a kind of consistency which is much more modest. The idea is that, for a given problem, we have to find an interpretation which is defined only for some of the terms in the expressions for the problem, and for the reasonably few terms which will occur when rules of inference and other syntactical transformations are applied to those expressions. Sometimes one knows in advance just what transformations are liable to be used, and the problem of finding the diagram may not be too complicated. (This would be the case if the student were able to characterize the problem accurately and see in advance what methods will be used.) Otherwise, in the course of finding the proof, the diagram may fail to support a consistent interpretation and will have to be altered. In other cases the diagram, while not exactly inconsistent, can turn out to be degenerate and lose its heuristic value. Actually, in our syntactical system, degeneracy will often turn out to be inconsistency. For a syntactic transformation that will usually be allowed is one which permits introduction of sentences like "Assume that the n points are taken in 'general position.'" This will lead to a logical inconsistency if some of the points in the interpretation (diagram) happen to be more collinear than warranted.

An example wherein a diagram may not support consistent interpretation occurs frequently in this way: During the investigation of a problem it becomes necessary to interpret an expression like "the intersection point of two non-parallel lines L_1 and L_2." The axioms directly assert the existence of such a point. But a figure repre-

senting the problem can only represent L_1 and L_2 by finite line segments, and the figure may have been drawn so that even when they are "extended" they do not intersect within the available paper.

If we were interested only in geometry and not in intelligence, we could dismiss this simple example as frivolous and use more paper or scale down the figure. But the previous diagram is being modified: even the instruction: "extend the line L_1" involves a change in the diagram. Instead of laughing away the problem, we can introduce the corrections into the program for the diagram building machine as elective transforms: "If there is no intersection point when our deductions say there should be one, then (1) use more paper and extend the lines or (2) scale down the whole figure, or (3) tilt the lines toward each other while making the other changes in the figure required to preserve that part of the consistency already required.

3.1.2 This example shows how the process of construction of a diagram can be controlled by a machine which uses interpretation consistency as a source of feedback control. We start with a set of expressions for a problem, and find a diagram in which each of the terms in the expressions has an interpretation. As the syntactic process proceeds, new terms may occur in new expressions. If the interpreting program fails to be able to find an interpretation of a new term, then the diagram must be altered: the type of inconsistency which arises should have a large effect on how the diagram-constructing machine makes the next modification. Thus the consistency required is only with respect to the actual expressions that are generated by the exploration in the investigation of a particular problem.

3.2 The use of locally consistent interpretations.

We continue with the plan of a machine which proves theorems in plane geometry by combining a Logic Machine and syntactic process which can handle the axiomatic formalism

of geometry and a device which provides a "locally consistent interpretation" of the problem of the moment in the form of a diagram.

3.2.1 The most immediate use that is made of a diagram, it seems to me, is to test whether a conjecture is true. For example, in the course of a proof I might ask "Is angle A equal to angle B?" or "Is line LM parallel to line L'M'?" Now such a question might arise as a "subgoal" in the logical analysis. If we have a good logic machine we could turn it loose trying to prove that the answer is affirmative. If the logic machine has available a decision procedure for that question, it could use that. But in general it would seem that one should hesitate to set out to prove a proposition before one has "evidence" that it is true, and it is a fact that in geometry as well as in the rest of mathematics, diagrams are used to supply such evidence. We simply look at a geometric diagram and see, i.e., by genuine undignified metric measurement, whether the two angles are approximately equal, or if the two lines are reasonably parallel. (It would not be difficult to set up a collection of metric criteria for making such decisions. The criteria would be a function of the accuracy of the diagram drawing process and the accuracy of the direct measurement methods. I believe that the general tolerances required for diagrams in plane geometry can be as poor as 5 per cent without appreciable loss of heuristic power.)

If the diagram indicates that the proposition is "metrically plausible" a decision must be made whether to

i. Try to find the proof.

ii. Obtain further metric confirmation. (Note that if the first diagram test is negative, then the proposition is false and further tests are not necessary, unless there is a serious inconsistency in the diagram interpretation. If the diagram is good, then the proposition should be abandoned as a subgoal.)

ii. (cont.) Now "further metric confirmation" may be taken to mean greater precision of measurement: one could tighten up both the precision of the drawing and the criteria of equality, say. While this is often convenient, I do not believe that it is a very intelligent way to obtain more evidence. A more powerful approach, heuristically, is to try to show that the measured equality is not due to an accident in the arbitrary choice of the metric parameters of the diagram. In any geometric diagram we have to make a number of arbitrary choices in the position of parts of the figure, length of certain line segments, etc. These choices are made by the program of the diagram-building machine. This program has built-in, or learns by sad experience, a number of heuristic subroutines:

-- If told to draw two intersecting lines without further specification, choose a "general" angle. Namely, always avoid getting within the "metric perception tolerance" (a given angle determined by the figure-inspection machine) of 90°, 60°, 45°, 30°, 0°. If other angles occur in the problem, or if other angles have been chosen for the previously drawn parts of the figure, avoid these too, if possible.

-- Strengthen the above rule so that triangles come out in "general form." In certain kinds of problems, be prepared to construct two figures to study both acute and obtuse angles.

-- When two or more distances are to be chosen, make sure that their ratios do not come too close to rational numbers with small numerators and denominators.

-- Etc. I do not believe that it will be difficult to write a very powerful program of this nature.

Since these decisions can be kept on record, then the diagram-drawing program can construct new, presumably equivalent, figures by changing one or more of them. Metric measurement of a few different but "typical" such diagrams provides fairly good insurance that the empirical truth of a proposition is not a metric accident. If each of the free parameters is separately varied and the proposition still measures true, then its logical truth is almost certain. I believe that a good student is simply able to perform such variation of figures "in his head" rather rapidly, and that this process can be automatic and almost unconscious.

3.2.2 Pattern recognition, "characterization of a figure," and the syntactic process.

A second use of diagrams, after a subgoal has been chosen, is to suggest a transformation or "method" to be used in the logical process for finding a proof of the proposition (subgoal). For example, a high school geometry student ought to have prominent in his memory of stored theorems and methods those relating to the situation when two parallel lines are crossed by a third. "Corresponding angles on a side are equal," "alternating interior, or opposite angles are equal," etc. The situations in which these methods are applicable are rather easily characterized by the phrase underlined above. Assuming that we have a figure-inspection machine which can make such characterizations, the character-method machine can be given the job of learning when to apply these useful methods.

I believe that it would be practical to write such a pattern recognition program to detect useful configurations in geometric diagrams. The problem seems to me to be considerably more simple than that of other pattern recognition processes that have been discussed. I conjecture that it would be practical to write a program that would perform as well as a good high-school geometry student without great difficulty, although it would require a great deal of labor unless a powerful translation program were prepared first.

A virtue of such an artificial intelligence program would be that one could program the figure to appear as a figure on an output device of the machine, say an X-Y plotter. With this kind of output it might not be too difficult to see what the machine was thinking about. As N. Rochester has observed, it is extremely important to obtain a reasonably compact presentation of what the machine is doing.

3.3 Some proposals for locally consistent interpretation machines.

-- Geometric diagrams for use with plane geometry machine.

-- Venn Diagrams for propositional calculus and elementary set theory decision methods.

-- Geometric diagrams for solution of analytic geometry problems and for other branches of mathematical analysis and topology.

-- Use of representation of game board and pieces in position may have some value over more abstract representation for certain games, namely those in which there are some useful geometric positional abstractions. In sufficiently highly combinatorial games this may not have any advantage, but in those where humans use effectively ordinary pattern-recognition it might help.

I will mention two other systems which I am working on and hope to present soon:

-- An interpretation-machine system to solve problems of the following type: Given a finite set described abstractly, how many points does it have? For example, how many ways can three dice come up so that there are not more than two of a kind?

With such a machine one could solve many problems of the "sample-space" type in probability theory. This problem was suggested by N. Rochester. If this machine was available, it could be used as a powerful subroutine for test of the validity of mathematical propositions in other fields by considering finite set examples as interpretations and testing validity in the finite case, by numerical substitution. A much simpler but

similar technique would be testing algebraic identities by substitution of numerical values. Plausibility of inequalities could be tested as well. It might be only fair to allow a computer to use its superhuman numerical arithmetic ability to make up for its poor rapid access memory and serial operation limitations.

— An interpretation-machine which uses strings of expressions of standard form and bounded length to handle deductions in a system using the axioms of arithmetic. This machine will use a formal system in which expressions like "$x_1, x_2, \ldots, x_n, \ldots$" occur and are handled in the way that a mathematician does. The special symbol "...." means that another symbol can be inserted to the left of the symbol. The symbol ",...," has a more complicated interpretation. One shows that a set is infinite by showing that it is legitimate to represent it in that way. Both the axiom of infinity and the axiom of induction are built into the rules for manipulating the sequence 1,2,3,... .

- - - - - - - -

*1.2.2 (cont.) The "current" method of selecting links has no advantage over the minimal chain method unless the graph contains cross-links. These will occur if the exploration machine has a "clean-up" routine which checks whether there are duplicate vertices in the graph, and if so identifies them. Note also that the links in the graph are oriented, i.e., validity in one direction is different than in the other, and that the representing electrical nets would thus contain a diode in each link.