

May 1971

AN INFORMATION PROCESSING APPROACH TO
REASONING BY ANALOGY

by

Robert E. Kling

Artificial Intelligence Group
Technical Note 56

SRI Project 8973

The research reported herein was sponsored by the
Advanced Research Projects Agency and the National
Aeronautics and Space Administration under Contract
NASW-2164.

TABLE OF CONTENTS

I	Introduction	1
II	Criteria for Analogy	2
III	Varieties of Analogy	3
IV	Information Transfer Between Problem Solutions	14
V	Automated Use of Analogical Information	19

I INTRODUCTION

Reasoning by analogy (RBA) has been discussed in artificial intelligence circles because of its extraordinary value in human problem solving and its elusiveness to mechanization. Without an ability to analogize, we would be unable to generalize, induce, or theorize. Moreover, thinking would be rather tedious, as we would have to solve each distinct problem afresh, without referring to previous experience. Fortunately the spectrum of similarities we are able to exploit is rather wide, encompassing many types, each with its own subvarieties. Unfortunately, we call much of this diverse behavior "reasoning by analogy." Hopefully, in the near future, we can develop some useful refinements for RBA. For the present, I'll simply describe without attempting to name.

With respect to any particular criterion of analogy, RBA includes the following activities:

- (1) Given a particular problem, theorem or situation (PTS) find a previously known (PTS) which is analogous.
- (2) Given a PTS, produce a special kind of analogous PTS. This would include producing the mechanical analogy of an electrical circuit, the n-dimensional analogy of a 2-dimensional geometric theorem, the continuous analogy of a discrete function, the interpersonal analogy of an international conflict, the French analogy of a Greek idiom, etc.
- (3) Given an explanation of the functioning of some PTS, provide an explanation for the functioning of some analogous PTS.
- (4) Given two PTS that are allegedly analogous, find at least one coherent analogy between them.

- (5) Given two analogous PTS and a set of consequences or observations drawn from one, infer an analogous set of consequences, inferences, or observations about the other.
- (6) Given two analogous problems and the solution for one, find an analogous solution for the other.

Since a machine engaging in RBA could perform any number of tasks, RBA is under-defined as an activity description.

In the preceding enumeration, the ability to construct an analogy (4) is a prerequisite to (3), (5), and (6). The following discussion focuses on problem solving and its consequences for analogy generation and use.

II CRITERIA FOR ANALOGY

When are two problems analogous? Whenever you say they are. Unless we specify what kind of analogy we want, analogy is a sort of similarity, and we can conceive (and even coyly defend) the most obscure similarities. Useful analogies between problems result when their solutions are sufficiently similar, to the extent that the solution of one may be used as a guide for the solution of the other at some suitable level of abstraction. Although we hope analogies are fruitful before they are employed, their ultimate test comes after their use.

Polya initiated a fresh discourse on analogical problem solving several years ago with his Mathematics and Plausible Reasoning.^{1*} His books are full of insightful examples, sketches of problem development, and pragmatic tips; however, his exposition is so informal and ill-organized from the point of view of automatic problem solving as to be all but useless. Polya did attempt to define analogy beyond the notion of "similarity" or

*References are listed at the end of the paper.

the dictionary definition, "correspondence in some particulars of otherwise dissimilar objects." His definitions are:

- (1) "Two systems are analogous if they agree in clearly definable relations of their respective parts."²
- (2) "Two objects are analogous if they agree in certain relations of their respective parts."³

"Clearly definable" could even be a well-specified random function and is hardly well defined! These definitions are barely operational, and too vague to be of much value in describing concrete criteria for an analogy. I believe any a priori definition of analogy would prove to be either vague or narrow. Rather, we need a typology of analogical types. In the following discussion I want to look at some of the ways in which problems are analogous, and I will draw some conclusions regarding some of the information processing necessary to recognize and exploit their analogies.

III VARIETIES OF ANALOGY

The current state of the theory of analogical problem solving is similar to the state-of-function theory in 1600. At that time the concept of "function" was little better developed than our notion of analogy is now. Much as function theory then lacked an appropriate vocabulary--continuous, differentiable, analytic, convergent, limit, etc.--we now lack a decent vocabulary for creating a sound typology of analogical types. The classes differentiated in this section are a suggestive proposal from an information processing perspective. Problem pairs are grouped by the similarity of information that is extrapolated from one problem to the other within each pair. Unfortunately, the resultant classes are neither

distinct nor hierarchically inclusive. Nevertheless, they permit a sensible description of the kinds of information transferred between problems within each pair.

Every style of reasoning and domain of knowledge offers possibilities for analogizing. To limit the scope of this work to manageable proportions this discussion covers problems that can be solved by deduction from some initial set of axioms, derived by the application of a set of operations to a set of initial states, or can easily be transformed into this form. Although it is possible to frame a wide variety of problems^e, including geometric constructions, puzzles, and robot manipulation tasks, into this framework, the majority of problems considered here are theorems in the usual sense.* More structured than "real-world problems", this class offers a decent starting point for any mechanized analogical problem solving that hopes to be successful.

A. Change of Parameters

Two PTS are recognizable as identical up to a change of parameters--e.g.,

$$I_1 = \int_{-\infty}^{\infty} (1 + x^2)^{-3} dx \quad , \quad n > 0$$
$$I_2 = \int_{-\infty}^{\infty} (n + x^2)^{-3} dx \quad , \quad n > 0 \quad .$$

Computing I_1 and I_2 are "parameter-variant" problems.

B. Generalization

One PTS is a generalization (or simplification) of the other.

- (1) Consider the relations between the 3-ring and 5-ring Tower of Hanoi puzzles.

*In the following discussion "problem" and "theorem" will be used interchangeably. Associated terminology, e.g., "solution" or "proof", "axioms" and "premises" etc., are considered equivalent and may vary with the context.

(2) T1. Given a triangle ABC, prove that the three vertex-angle bisectors meet in a unique point.

T1'. The premises of T1 imply that this point is the center of the inscribed circle.

T2. Given a tetrahedron WXYZ, prove that the bisectors of any three dihedral angles that do not meet in a common vertex intersect in a unique point.

T2'. The premises of T2 imply that this point is the center of the inscribed sphere.

C. Similar Relational Structures

T3. The bisectors of the three vertex angles of a triangle intersect in a unique point which is the center of the inscribed circle.

T4. The perpendicular bisectors of the three sides of a triangle intersect in a unique point which is the center of the circumscribed circle.

T5. The intersection of two abelian groups is an abelian group.

T6. The intersection of two commutative rings is a commutative ring.

The pairs of theorems T3/T4 and T5/T6 are "relationally isomorphic" when represented as graphs with nodes and links of different types to represent relations and objects of different classes. (The partitions of nodes and branches is, in effect, a categorical semantics for the graph language.) In viewing the proofs of these theorem pairs, one finds that they are identical up to a set of substitutions (e.g. abelian group/commutative ring, angle bisector/perpendicular bisector, etc.) that results from the mapping associated with the analogy.

This class is an extension of the parameter-variant class, and with some provision for mapping sets (clusters of nodes) into sets of different cardinality, they may also include many generalization-type analogies. Note that the relational isomorphism is "local."

The preceding analogies were selected for their transparency, but even isomorphisms can be complex. For example, consider:

- T7. Let ABC and abc be two triangles in the same plane defined within a k -dimensional finite geometry over the Galois field $GF[p^n]$. Let these triangles be perspective for a point O , such that O, A, a are colinear, O, B, b are colinear, and O, C, c are colinear. Let α be the point of intersection of AB and ab , β that of AC and ac , and γ that of BC and bc . Then the points α, β , and γ are colinear.
- T8. Let X, Y , and Z be three subgroups of a geometric set of subgroups of G_k such that no one of them in the group is generated by the other two. We select other subgroups of the geometric set as follows: each of them is in the group $\{X, Y, Z\}$; O is any such subgroup not contained in any one of the subgroups $\{X, Y\}, \{Y, Z\}, \{Z, X\}$; x, y, z are such subgroups different from O, x, y, z and contained respectively in $\{O, X\}, \{O, Y\}$, and $\{O, Z\}$. Let μ, ξ , and ν be the subgroups of

the geometric set of subgroups common to the respective pairs of groups $\{X,Y\}, \{x,y\}, \{Y,Z\}, \{y,z\}$, and $\{Z,X\}, \{z,x\}$.

Then each of the two subgroups μ, ξ , and ν is in the subgroup generated by the other two.

Every k -dimensional projective geometry over a Galois field $GF[p^n]$ is capable of a concrete representation by an Abelian group of order $p^{(k+1)n}$ and type $(1,1,1,\dots,1)$ by considering each subgroup of order p^n as a point in the geometric space.⁵ This association renders T7 logically equivalent and relationally isomorphic to T8, although this correspondence is hardly obvious.

D. Plans are Identical

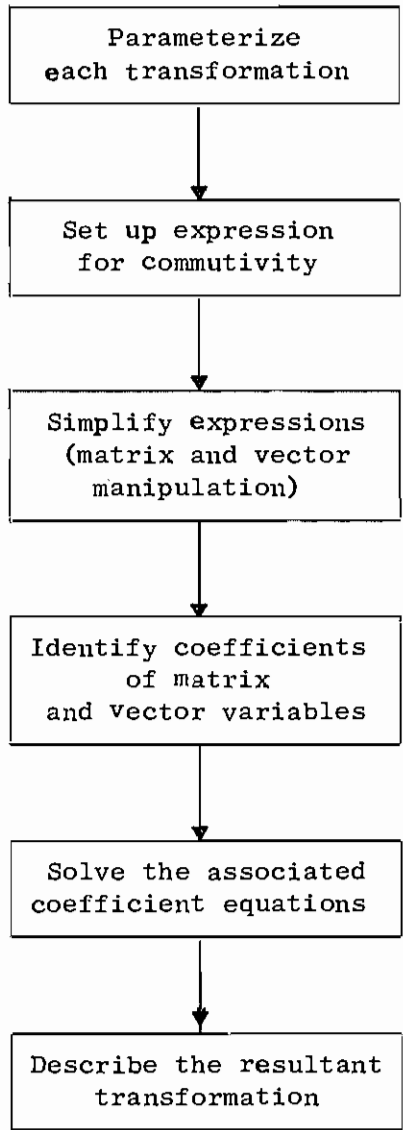
Plans for the solution of each PTS are identical (at some level of abstraction) — e.g.,

(1) T9. If a given affine transformation commutes with every other affine transformation, then that transformation is the identity.

T10. If a given affine transformation commutes with all the translations, then that transformation is also a translation (see Fig. 2 for proof plan).

(2) T11. If $F(w)$ is the Fourier transformation of $f(t)$, prove that $e^{-j\omega t} F(w)$ is the Fourier transformation of $f(t-T)$.

T12. If $F(w)$ is the Fourier transformation of $f(b)$, prove that $\frac{1}{(a)} F \frac{w}{a}$ is the Fourier transformation of $f(at)$ (see Fig. 3 for proof plan).



Affine $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$
 $\vec{v} = A \vec{u} + \vec{a}$

Translation
 $\vec{v} = I \vec{u} + \vec{a}$

Identity
 $\vec{v} = I \vec{u}$

FIG. 2 PLAN FOR PROVING THEOREM 9 OR THEOREM 10

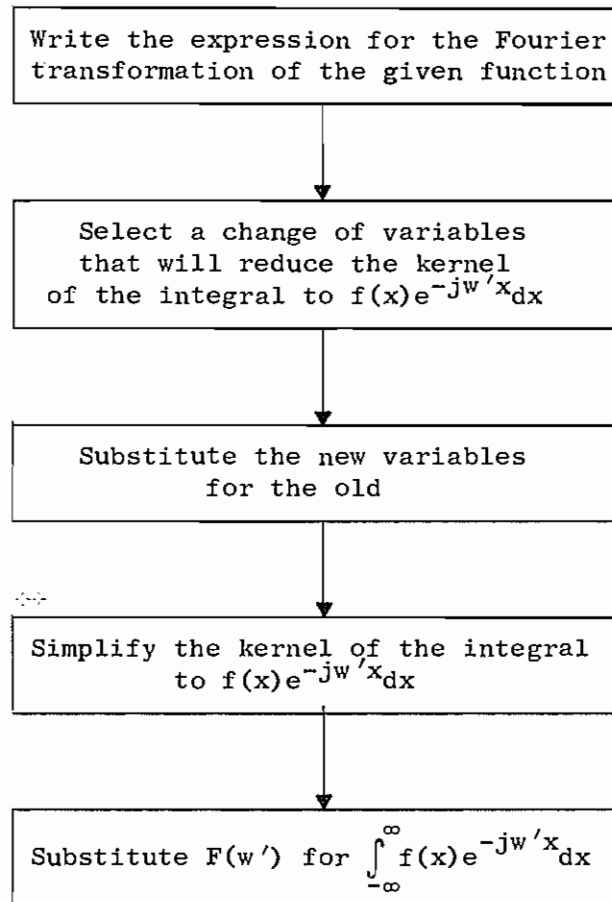


FIG. 3 PLAN FOR PROVING THEOREM 11 OR THEOREM 12

(3) T13. Let an arbitrary line l_1 intersect each of three parallel lines s_1 , s_2 , and s_3 at points p_1 , p_2 , and p_3 , respectively. Let another arbitrary line l_2 intersect s_1 , s_2 , and s_3 at points q_1 , q_2 , and q_3 , respectively. Then

$$\frac{\overline{p_1 p_2}}{\overline{p_2 p_3}} = \frac{\overline{q_1 q_2}}{\overline{q_2 q_3}} .$$

T14. Let an arbitrary line l_1 intersect each of three parallel planes s_1 , s_2 , and s_3 at points p_1 , p_2 , and p_3 , respectively. Let another arbitrary line l_2 intersect s_1 , s_2 , and s_3 at points q_1 , q_2 , and q_3 , respectively. Then

$$\frac{\overline{p_1 p_2}}{\overline{p_2 p_3}} = \frac{\overline{q_1 q_2}}{\overline{q_2 q_3}} .$$

(see Fig. 4 for proof plan).

Although a coherent planning language for this diverse set of problems has not yet been written, it is clear that they are "identical" at some level of abstraction easily accessible to people.

E. Change of Representation

Both PTS's solutions involve a common change of representation and style of argument.

(1) P1. Consider the classical truncated checkerboard domino-covering problem.

P2. Consider a 3×3 cubical apple with a worm on its surface. The worm travels from cube to adjacent cube, boring a hole without ever returning to a

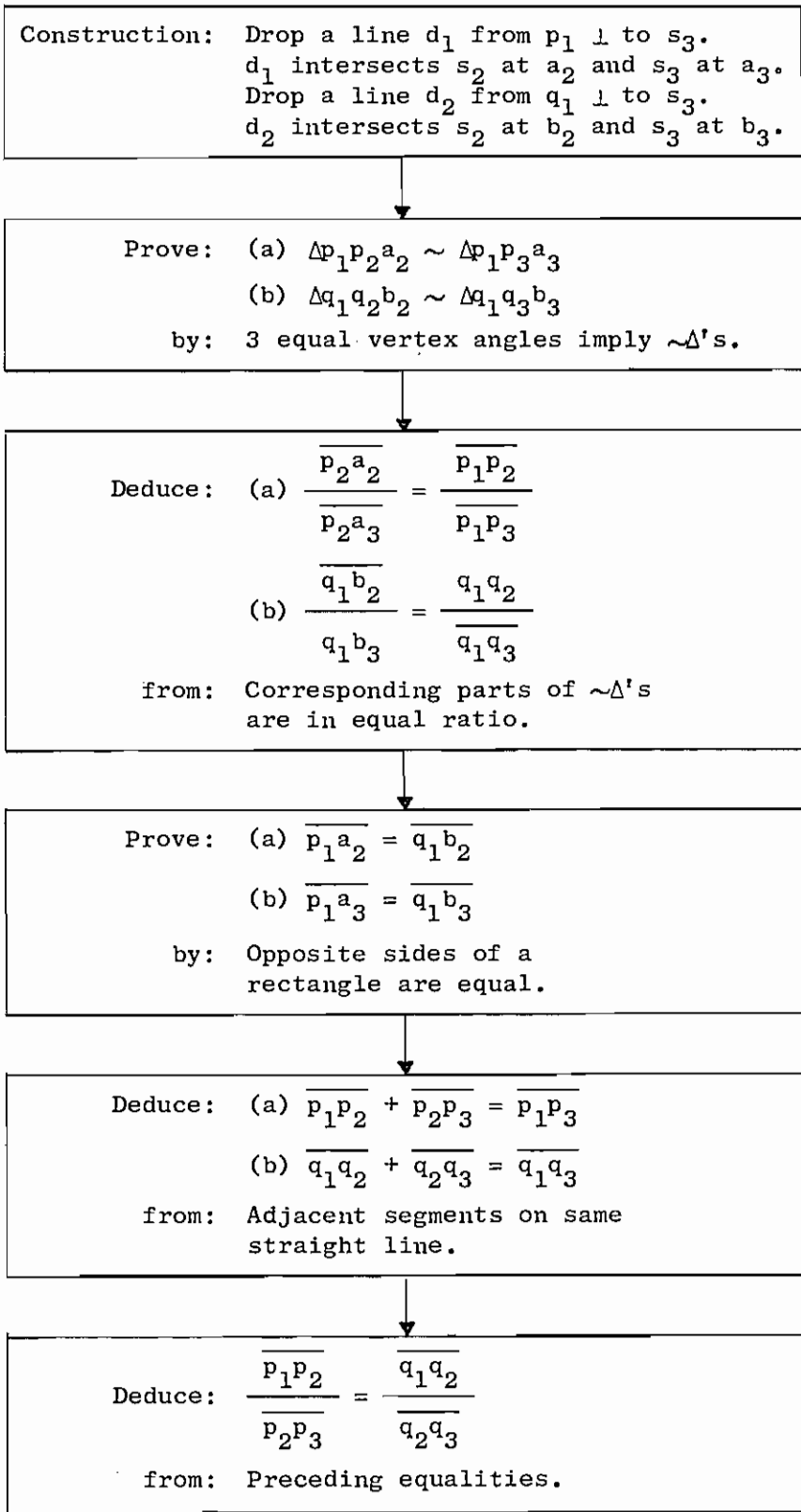


FIG. 4 PLAN FOR PROVING THEOREM 13 OR THEOREM 14

previously drilled subcube. Prove that it is impossible for the worm to terminate his path in the centermost cube.

(2) If A is a matrix with transpose A^T ,

$$T15. (A^T)^T = A$$

$$T16. (A B)^T = B^T A^T.$$

(3) If A is a matrix with inverse A^{-1} ,

$$T17. (A^{-1})^{-1} = A$$

$$T18. (A B)^{-1} = B^{-1} A^{-1}.$$

Each of the preceding problem pairs entails similar representations.

The truncated checkerboard and the cubical apple problems are both solved by coloring adjacent cells black and white and then using a parity argument.

If we restrict ourselves to a simple operator-free matrix algebra, T15 and T16 are most easily proved by representing each matrix by its ij^{th} element and manipulating the ij^{th} terms according to the specified

computations. On the other hand, T17 and T18 are both

easily proved by simple algebraic operations. What a person

extrapolates from T15 to T16 or from T17 to T18 is a specific representation

in which problem-solving ability is enhanced. If a person were faced with

the problem $(A^T)^{-1} = (A^{-1})^T$, he might be unsure of which representation

to choose, and would try either one. (It turns out that either representation affords straightforward proofs.)

F. Common Subproblem

Both problems involve a common subproblem.

(1) Let \tilde{A} be a matrix with elements a_{ij} and an inverse \tilde{B} . Then

$$b_{ij} = \frac{c_{ij}}{\det[A]}$$

and

$$\det[A] = \sum_{j=1}^n (-1)^{i+j} a_{ij} c_{ij}$$

where c_{ij} is the i - j th cofactor of \tilde{A} . Thus, the computation of \tilde{A}^{-1} and $\det[a]$ share the common subproblem of computing some cofactor of \tilde{A} .

(2) Consider a robot in a room full of scattered metal furniture.

P1. The robot is asked to paint the floor of the room.

P2. The robot is asked to replace each piece of furniture with a similar wooden piece from the next room.

Each of these problems can be solved by first clearing all the metal furniture out of the given room, and in that sense they are analogous.

Problems that involve only one common subproblem can be really rather different, and still allow useful problem-solving extrapolations. Probably these extrapolations are best regarded by treating the subproblems as substantial problems unto themselves. For example, every time we encounter a trigonometric integral in solving some problem, we become better integrators and increase our facility for rapidly guessing appropriate substitutions. Thus, the extrapolation of integration techniques from one problem to another is due to recognizing the need for our developed skill as an integrator, rather than noting some gross aspects of problem structure.

III INFORMATION TRANSFER BETWEEN PROBLEM SOLUTIONS

Many researchers in artificial intelligence believe that a bold step towards RBA will be taken when an automatic algorithm for creating an analogy between two problem statements is developed. Presumably, such an algorithm need only know

*

the two theorem statements and have access to the data base from which the theorem/^{prover} used in solving both problems are stored. I no longer share that belief. Even if one has a detailed analogy that is limited to the relations and objects explicitly mentioned in the theorem statements, one still must know how to use this information to accelerate the search for a solution to the unproved theorem. Secondly, while some useful information may be gleaned from this "restricted analogy," a casual look at any nontrivial pair of analogous proofs will reveal that much of the information that people would extrapolate into the proof of the new theorem involves additional relations, facts, and patterns of inference that are absent from the problem statement. Any interesting analogy-generating algorithm will need to operate upon theorem proofs as well as theorem statements. Thirdly, the search for analogous "additional information" helps pin down the viability and level of abstraction that can be expected from a given analogy.⁶

In any but the most simple problems, the solution is derived in terms of relations that don't appear in the problem statements.

The following discussion is designed to stimulate further thinking about the use of information extrapolated from the solution of a solved problem. For congruence with my preceding description of analogical types, the discussion is general. To be sufficiently specific to describe algorithms, the details of the particular representations (e.g. prenex normal form of first-order predicate calculus) and their logic (e.g. resolution, natural inference, etc.) need be known. For example, ZORBA,⁷

an analogy program which operates on top of a resolution-based theorem prover, uses clauses which are natural to resolution rather than the operator system indigenous to GPs.

Suppose we had a magical system which could offer information helpful to proving an unknown theorem if it were given an analogous proved theorem. What kind of interesting advice could we expect from this program? At one extreme it might be clairvoyant and offer a complete solution to the baffling problem. Short of such omniscience, what kind of partial information would be helpful? Textbook writers often append hints of two types to the problems they provide:

- (1) Problem difficulty (easy, hard)
- (2) "Hints" which include:
 - (a) suggested representation
 - (b) appropriate methods
 - (c) relevant principles or theorems
 - (d) valuable subproblems.

Let's examine this second class more closely. There seem to be three different levels of "heuristic detail," each with possible attendant information.

A. Representations

Representations are mentioned in Part E of the preceding section.

A style of argument may be added--e.g. induction, parity, etc.

Additional details such as which parameter to induct on may be included.

B. Plan

Consider a problem solution as a sequence of states S_j and state transition operators P_j , as in Fig. 5 below:

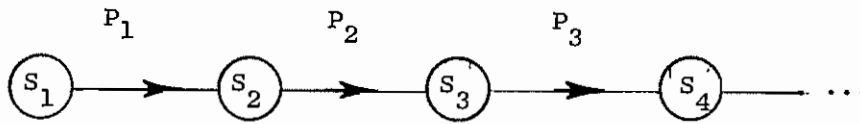


Figure 5

A Plan Depicted as a Sequence of States

Although the depicted operators have unary inputs, several inputs may be possible, as in inference from several intermediate results (states)⁶. Likewise, there may be several outputs; e.g., a problem may be split into several intermediate subproblems. A plan is any sequence of state descriptions and/or operator descriptions that parallels and alleged problem solution. These descriptions are usually abstracted versions (patterns) which may have several candidates in the object language. In this sense they are weakly specified. Several varieties of information may be offered as a plan:

- (1) A sequence of operations or methods may be specified (Fig. 3).
- (2) A sequence of patterns which describe the expected state sequence (a state "monitor").
- (3) A sequence of subgoals described in the object language (Fig. 4).

A functional planning language needs at least four features:

- (1) its own logic
- (2) a well-defined nexus between the planning states/methods and the object-language states/methods,
- (3) an ability to reference future results,
- (4) some facility to manipulate data representations so that the flow between different processes is smooth.

These features are integral to an autonomous planning system. When we focus on the kinds of information that can be extrapolated from problem to problem at a planning level we find:

- (1) Plan , as described above
- (2) Estimates on the difficulty of various subproblems
- (3) Conditions describing when to terminate a process: "simply the kernel of an integral to $f(x)e^{-jw'x}$ (Fig. 3)
- (4) Operator inputs: "deduce S_j from Theorem T_k and S_{j-1} ."

This wealth of side information (nonsequential) associated with a plan can range in abstraction from being detailed in the object language (a particular theorem) to some more abstract description (e.g. a theorem relating groups and homomorphisms, a sufficient condition for a set to be a group, etc.)

C. Object Language Level

- (1) Explicit subproblems and lemmas (Fig. 4)
- (2) Relevant theorems that will be used in the proof (Fig. 4)
- (3) The set of relations to be used in the problem solution
- (4) Problem difficulty.

The set of relevant theorems need not be structured with their relevant subgoals as in Fig. 4, but may be an unordered set of which the PSS is conspicuously conscious.

It is now clear that range of helpful advice is rather broad, both in level of detail and degree of structure. Although a restricted analogy could be generated first and the "helpful information" later, it would be nice if some of it were a byproduct of the analogy-generating program.

V AUTOMATED USE OF ANALOGICAL INFORMATION

In this section I will coalesce several themes that have run through this paper and apply them to the automation of analogical problem solving. First, I will summarize some of the key points that I have mentioned in the preceding sections.

A. The idea of analogy is ill defined. There are at least several kinds of related analogies.

B. Each of these analogical varieties would best be recognized by somewhat different means.

C. The kind of information that is extrapolable between analogous problems of each variety are quite different. Thus, the algorithms and designs for using these diverse types of information are likely to be quite different.

D. One of the key issues in extending analogical information is knowing in advance the level of generalization which will hold for each analogized parameter, method, operation, theorem, or fact.

E. A set of strategy/planning languages that would allow an appropriate degree of generality would be quite complex. These facts imply:

F. No analogy-oriented PSS (APSS) should be expected to process all varieties of analogy, since each involves a somewhat different style of information processing.

G. An APSS that attempts to extrapolate general sequential plan-like information or patterns of inference, and attempts to actively direct a problem solver which incrementally infers and tests inferences against its supervisory schema would be quite complex.

Many of the example problems presented in Section III push the limit of contemporary PSS and will probably be non-trivial for any of the planning oriented systems that will emerge in the next few years. Thus, we end up wanting to use analogical information without creating plans or other forms of skeletal solution structures.

The means of doing this are actually very simple if we review our situation again. A typical APSS will have a large data base and be presented with a pair of problems: One is unsolved and the other has already been solved and its solution is available to the APSS. I want to underscore a critical way in which this situation differs from the typical PSS. Most PSS work on a "minimal" data base for which the user has selected an adequately small set of axioms.* When the data base of a typical PSS is expanded to

*All known resolution systems and GPS operate this way. Gelernter's Geometry theorem-prover seems to have run on a large data base, but his work is inadequately documented to verify this fact.

include some irrelevant axioms, they begin to generate a substantial set of irrelevant inferences (due to the interaction of relevant and irrelevant axioms and their descendent inferences). Consequently, they begin to flounder in their search and may fail to solve problems that are easily solvable with a minimal axiom set. To be concrete, consider a PSS which uses 8 axioms to prove some theorem T with a search that generates 100 inferences for, say, a 20 step proof. Adding 10 more axioms to the data base may force S1 into generating 500 inferences before finding its 20 step proof. In a sense, these figures are doctored since a set of 10 axioms can be chosen that will have no appreciable effect on the search space size while another set of 10 may be added which can explode the search space almost arbitrarily.

Since an APSS will be proving somewhat diverse theorems a (usually) common data base, it is in principle bound to seek proofs in a context abundant in excessive and irrelevant data. One key method for exploiting analogical information is to select subsets of axioms appropriate for proving the new theorem. What we are doing then is constricting the context in which theorem proving takes place by narrowing the set of accessible axioms. The usual strategies fo the particular PSS can be used unmodified; the analogical information is merely used to narrow the context sufficiently to reduce the search space to a more manageable size.

* In the first case add axioms that use many distinct predicate letters and many distinct function symbols. In the latter case use one or two predicate letters, and that the axioms will resolve with most others, preferably recursively.

Selecting an appropriate axiom set is one of several kinds of information that may be added independently of PSS strategy. A more complete and suggestive listing includes:

- . Restricting the set of admissible relations
- . Restricting the set of admissible operator symbols
- . Restricting the set of admissible axioms
- . Restricting the order of operator nesting
- . Generating analogous subproblems, solving them, and adding them as axioms.

This list can be extended depending upon the kinds of information used by a particular PSS. Thus, if a PSS has a look-ahead estimator (like REF-ARF) then that too may be analogized without modifying the PSS structure. The key idea is that an effective means of exploiting analogical information is to modify the context in which a particular theorem prover operates rather than subjecting it to a planning-like scheme that supervises the sequence of its inference making.

Now, the actual means for generating analogies and the means of extrapolating analogous axioms depends upon the representations and PSS used. These details have been developed and implemented for a resolution-based theorem prover and are described in Reference 7.