A HEURISTIC PROGRAM THAT SOLVES SYMBOLIC INTEGRATION PROBLEMS IN FRESHMAN CALCULUS, SYMBOLIC AUTOMATIC INTEGRATOR (SAINT)

by

JAMES ROBERT SLAGLE

S.B., Saint John's University

(1955)

S.M., Massachusetts Institute of Technology

(1957)

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF

PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE, 1961

Signature of Author.....
Department of Mathematics, May 13, 1961

Certified by.....
Thesis Supervisor

Accepted by.....
Chairman, Departmental Committee
on Graduate Students

A HEURISTIC PROGRAM THAT SOLVES SYMBOLIC INTEGRATION PROBLEMS IN FRESHMAN CALCULUS, SYMBOLIC AUTOMATIC INTEGRATOR (SAINT)

by

James Robert Slagle

Submitted to the Department of Mathematics on 13 May 1961 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

Some ideas about problem solving by a machine were tried out on the nontrivial problem domain of elementary symbolic integration. To this end, the author programmed a large, high speed, general-purpose digital computer (I.B.M. 7090) to use these ideas to solve some symbolic integration problems. The computer so programmed is called "SAINT", an acronym for "symbolic automatic integrator". SAINT performs symbolic integration which includes indefinite integration. It also performs definite and multiple integration when these are trivial extensions of indefinite integration. SAINT solves symbolic integration problems approximately at the level of a good college freshman and, in fact, uses many of the same methods (including heuristics) used by a freshman. Taking an average of two minutes per problem, SAINT solved fifty-two (ninety-six per cent) of the attempted fifty-four M.I.T. freshman calculus final examination problems. The author draws many conclusions from this and other experiments with SAINT and makes suggestions for future work in the field of Artificial Intelligence.

Thesis Supervisor: Marvin Minsky
Title: Assistant Professor of Mathematics

Biography of the Author

James Robert Slagle was born in Brooklyn, New York on March 1, 1934. He received full tuition scholarships to Saint Francis Preparatory School (1948) and to Saint John's University (1952), both in New York. At the end of his senior year in college he was given the University President's Award for attaining the highest scholastic average in the university that year. After receiving his S.B. summa cum laude, in June 1955, he became a staff member at M.I.T.'s Lincoln Laboratory. One year later the laboratory awarded him a staff associateship. Under this program he received his S.M. in Mathematics from M.I.T. in 1957, and will receive his Ph.D. in Mathematics in June 1961. His Master's thesis was titled Investigation into the Convergence of Multiplicative Processes.

President Dwight D. Eisenhower, in 1959, presented him with five hundred dollars, awarded by Recording for the Blind, Inc., for outstanding work as a blind student. While at M.I.T. the author has been a member of the American Mathematical Society. He was married in 1958, and has two sons.

Table of Contents

Chapter I

The SAINT Project, Its Purpose and Historical Background

A. Nature of the SAINT Project

Some ideas about problem solving by a machine were tried out on the nontrivial problem domain of symbolic integration. To this end, the author programmed a large, high speed, general purpose digital computer (I.B.M. 7090) to use these ideas to solve some symbolic integration problems. The computer so programmed is called "SAINT", an acronym for "symbolic automatic integrator". This paper discusses the construction of the SAINT program and some experiments with it.

To anticipate the later development, some typical samples of SAINT's external behavior are given so that the reader may think in concrete terms. First suppose an ordinary I.B.M. card contained punches representing (in a suitable notation) the symbolic integration problem, $\int xe^{x^2}\,dx$. Now let SAINT read this card in its card reader. In less than a minute and a half, SAINT prints out the answer, $\frac{1}{2}e^{x^2}$. Except where otherwise noted, every problem mentioned in this paper has actually been solved by SAINT. Note that SAINT neglects the constant of integration, and we, too, shall ignore it throughout our discussion. Secondly, after working for less than a minute on the problem, $\int e^{x^2}\,dx$ (which cannot be integrated in elementary form) SAINT prints out that it cannot solve this problem. The

problem $\int \frac{x^4}{(1-x^2)^{5/2}}\,dx$, is problem 3c on the May 1957 M.I.T. final examination for the second half of the first year calculus course and hence is accompanied by the notation, (2 May '57, 3c). After working for about eleven minutes on this problem, SAINT prints out the answer, arcsin x + 1/3 $\tan^3$ arcsin x - tan arcsin x. Below may be answered some of the questions which might occur to the reader if he had witnessed such a demonstration.

1. Exactly what can SAINT do? What kinds of integrands can it handle?

SAINT can perform symbolic integration which includes indefinite integration, also called antidifferentiation. It can also perform definite and multiple integration when these are trivial extensions of indefinite integration. SAINT handles integrands that represent explicit elementary functions of a real variable which, for the sake of brevity, will be called elementary functions. The elementary functions are the functions normally encountered in freshman integral calculus. SAINT does not handle hyperbolic or inverse hyperbolic notation. The elementary functions are defined recursively as follows:

a. Any constant is an elementary function.

b. The variable is an elementary function.

c. The sum or product of elementary functions is an elementary function.

d. An elementary function raised to an elementary function power is an elementary function.

e. A trigonometric function of an elementary function is an elementary function.

f. A logarithmic or inverse trigonometric function of an elementary function (restricted in range if necessary) is an elementary function.

2. What problems can SAINT solve? How does it solve them?

SAINT solves symbolic integration problems at approximately the level of a good college freshman. A full description of how SAINT solves symbolic integration problems will be found in chapter IV. For now, suffice it to say that SAINT employs many of the same methods used by a college freshman, including heuristic methods defined below. Since the SAINT program uses heuristic methods, it is by definition a heuristic program. Although many authors have given many definitions, in this paper a heuristic method (or more simply a heuristic) is a method which helps in discovering a problem's solution by making plausible but fallible guesses as to what is the best thing to do next. An algorithm is defined as a method that can decide infallibly a suitable thing to do next for a known class of problems. The SAINT program was written by the author, bearing in mind the memory limitations of the I.B.M. 7090 digital computer and the following purposes.

B. Purposes of the SAINT Project

Through the centuries, natural (human) intelligence has been directed towards achieving two general goals, the acquisition of knowledge and the solution of problems. Only in recent years has appeared the large high speed general purpose digital computer which furnishes us with a new and powerful tool for understanding, using and enhancing intelligence to help achieve these general goals. Furthermore, we stand on the threshold of artificial intelligence, i.e., intelligent behavior by machines. Thus, the purposes of the SAINT project are classified according to the kind of intelligence involved. That the SAINT project has met these purposes at least in part will be seen in chapter VI. Section C of the present chapter gives the more specific reasons for choosing symbolic integration as the problem domain. The purposes concern subjects which fall into three categories, namely, intelligence whether natural or artificial, natural intelligence and artificial intelligence.

1. Intelligence whether natural or artificial

Under this category, are listed four subjects (pattern recognition, algorithms and heuristics, problem solving and learning) for which we have certain purposes.

a. Pattern recognition

Can a computer recognize the needed kinds of patterns that occur in symbolic expressions? Just how important is pattern recognition? Can a machine learn to recognize new patterns?

b. What is the relative role of heuristics and algorithms?

c. Can intelligent problem solving behavior really be manifested by a machine?

d. Can a machine learn?

2. Natural intelligence

Can there be constructed an easily modifiable partial model of intelligent human problem solving in the domain of symbolic integration? How will advances in computers and computer programming affect teaching?

3. Artificial intelligence

What kinds of computers, symbol manipulating languages and compilers are needed for complex symbol manipulating tasks including artificial intelligence applications? How can heuristic programs be improved?

C. Choice of Symbolic Integration as the Problem Domain

To meet, at least in part, some of the above purposes, symbolic integration was chosen rather than many other possible projects (such as those discussed in section VII C) because

this problem domain has certain specific advantages. Symbolic integration is a well defined problem domain, i.e., a proposed solution can be tested for correctness. It seemed wise to try a well formulated problem domain before trying a domain in which the problems must be formulated abstractly before solutions can be attempted. Of course, this latter type of problem which includes most problems from every day life is also very important and also deserves careful consideration. In addition, symbolic integration is familiar but causes some difficulty to many people. A symbolic integration program is potentially useful in itself and has important extensions. Last but not least, symbolic integration involves the manipulation of symbolic expressions. Such symbol manipulating will probably be fundamental in future problem solving by machines. We have reason to believe that a computer can manipulate symbolic expressions only slightly more slowly than it can handle numerical ones. Symbol manipulation has much the same advantage over numerical manipulation that algebra has over arithmetic. Hence a symbol manipulating machine can do vastly more work per calculation. Some historical background to symbol manipulating languages will be found in section D, and a simplified version of one such language will be found in chapter II.

D. Historical Background of the SAINT Project

This section sketches the history of symbolic integration, mathematics machines, artificial intelligence and symbol manipulating languages. Since a history of symbolic integration can easily be found and since Minsky's recent and excellent bibliography [8] covers the other three topics, very few references are included. Here as elsewhere the number enclosed within brackets refers to an entry in the author's bibliography.

We pick up the first two of the four threads in our story with one and the same man, Leibniz (1646 - 1716). He was the first to make the conjecture that, if mathematics were properly formulated, a machine could prove all the theorems. However, for the discovery of symbolic integration as a general method, Leibniz must share the limelight with Newton (1642 - 1727). Although Newton's notation is good, Leibniz introduced a still better notation including the use of the symbol "$\int$" for the operation of integration and the symbol, "d", for the inverse operation. The work of Liouville (1809 - 1882), as developed by Hardy [4] in 1905 and Ritt [10] in 1948, determined the general form of the integral of an elementary function when the integral is elementary and gave some elementary procedures for finding the integral of some special classes of elementary functions. No general method exists for determining whether

an algebraic function can be integrated in elementary form.*

In the 1920's, Hilbert revised Leibniz's dream by trying to formulate mathematics in such a way that computational methods would suffice to find the theorems of mathematics. The dream that had lasted for two and a half centuries was seemingly shattered in 1931 by Goedel's incompleteness theorem to the effect that for any machine that proves only true statements in arithmetic, there exists a true arithmetic statement which that machine can not prove. The machine of this and the following theorem is a Turing machine, roughly a digital computer supplied with an indefinitely long memory tape. A further blow came in 1936 when Church showed that for any machine, there exists a statement in the first order predicate calculus (an elementary form of logic) for which the machine cannot decide whether or not the statement is true. There is no evidence to suggest that mathematicians are free from the limitations mentioned in these two theorems. The theorems show that the machine cannot do everything in mathematics. However, they can do a great deal. Artificial intelligence is a new field which shows great

---

* Joseph Ritt, Integration in Finite Terms, Liouville's Theory of Elementary Methods (New York, 1948) pp. 20, 33.

promise of programming machines to do highly significanl mathematics.

Artificial intelligence concerns itself with other things besides mathematics. In fact, artificial intelligence refers to any attempt to cause a machine to manifest behavior which, if exhibited by a human, would be called intelligent. In 1950, Shannon [13] discussed how a computer might be programmed to play chess. In 1955, Selfridge [12] and Dinneen [2] reported some experiments with a computer programmed to recognize visual patterns. Minsky [7,8] has done much invaluable theoretical work and has unified the field of artificial intelligence to a large extent. In 1959 appeared Samuel's excellent paper [11] which describes some machine-learning experiments with a computer programmed to play checkers.

We conclude our history by mentioning some mathematics machines and artificial languages for symbol manipulation. In 1956, Newell, Shaw and Simon ran experiments with their LOGIC THEORIST [9] the program for which was written in their information processing language (IPL) which could conveniently handle list structures. The LOGIC THEORIST proved seventy per cent of the (propositional calculus) theorems in chapter II in Whitehead and Russel's Principia Mathematica in the axiom system of that book. In 1959, Gelernter [3], using

his FORTRAN list processing language (FLPL) succeeded in causing his GEOMETRY THEOREM PROVING MACHINE to prove theorems approximately at the level of a high school student. When trying to prove a theorem in geometry, this machine constructs a diagram "in general position" and rejects immediately any subgoal which is false in the diagram. In 1960, McCarthy's list processing (LISP) language [6] was made available on the I.B.M. 709 and 7090 digital computers. The SAINT program is written in LISP, a simplified version of which appears in the next chapter.

Chapter II

Symbol Manipulation by a Machine

The manipulation of symbolic expressions occurs throughout mathematics, logic and, in fact, all of verbal reasoning, e.g., proving theorems in formal systems and simplifying mathematical expressions. Hence, if a machine is to solve really difficult problems, we must have good symbol manipulating languages and machines. This chapter contains a simplified version of McCarthy's LISP (list processing) language and its realization on a computer [6]. The reader who is already familiar with LISP should skip this chapter after he has noted that we use f and r for car and cdr, respectively. The real LISP language, especially in its newer versions, is better than the simplified version presented here. The purpose of the simplified language is twofold, to acquaint the reader with some of the basic concepts of symbol manipulation for its own sake and then to supply him with some background for greater appreciation of the SAINT project. The presentation of the language is accompanied by examples, culminating with a procedure for performing differentiation of mathematical expressions. Then is given a survey of the procedures already available in LISP. This chapter concludes with some remarks on the realization of such a language on a computer.

A. A Symbol Manipulating Language

This section describes a language similar to McCarthy's LISP. The language presented here is used later to describe some symbol manipulating procedures in SAINT.

1. The set of symbolic expressions

By definition, an atom is any sequence of Arabic numerals and upper case Roman letters. The set of symbolic expressions (or more briefly, s-expressions) is defined recursively as follows:

a. Any atom is an s-expression.

b. Any list of s-expressions is an s-expression, i.e., if s_1, s_2, ..., s_n are s-expressions, then so is (s_1, s_2, ..., s_n) for n = 1, 2, ...

2. Elementary (symbol manipulating) predicates and functions

Function and procedure are treated as synonyms. Some functions are partial functions, i.e., they may not be defined for some arguments. A predicate is a function which, when defined for and applied to its arguments, has for its value one of the two atoms, TRUE or FALSE. After this introduction, we give the five elementary predicates and functions used in LISP.

a. "atom" is a predicate defined on the set of s-expressions. The value of atom [s] is TRUE if and only if s is an atom. Thus, the value of atom [NIL] is TRUE whereas the value of atom [(PLUS, X, Y)] is FALSE.

b. "eq" (abbreviation for equal) is a predicate defined only when both of its arguments are atoms. The value of eq [s_1, s_2] is TRUE if and only if s_1, and s_2 are the same atom.

c. "f" (abbreviation for first) is a function defined when its argument is a list, i.e., not an atom. The value of f[s] is the first member of the list s, e.g., the value of f [(PLUS, X, Y)] is PLUS.

d. "r" (abbreviation for rest) is a function defined when its argument is a list. The value of this function when applied to a list with one member is NIL, e.g., the value of r [(G0002)] is NIL. The value of this function when applied to a list with more than one member is the rest of the list, i.e., the list of all of its members except for the first, e.g., the value of r [(PLUS, X, Y)] is (X,Y).

e. "cons" (abbreviation for construct) is a function defined when the second of its two arguments is either NIL or a list. The value of cons [s; NIL] is (s). The value of cons [s_1, s_2] where s_2 is a list is the list whose first element is s_1 and the rest of whose elements are the elements of s_2, e.g., the value of cons [PLUS; (X, Y)] is (PLUS, X, Y).

3. Forms and functions formed from elementary functions by composition.

We distinguish between a form and a function. A form is an expression such as cons [x; cons [y; NIL]], composed of function names, variables and constants. A function may be represented

by a form prefixed by the Church lambda notation [1] from which we deviate trivially. If x_1, x_2, ..., x_n are variables that may occur in the form, then λ [[x_1; x_2; ...; x_n]; form] denotes a function, namely the function of n variables which maps x_1, x_2, ...x_n into the form, e.g., the value of λ[[x; y]; cons[x; cons [y; NIL]]] [MINUS; G] is (MINUS, G). In fact the value of this function when applied to its arguments is simply a list of those two arguments. We may name a function as follows:

$$\text{function} = \lambda[[x_1; x_2; \ldots; x_n]; \text{form}]$$

or equivalently

$$\text{function}[x_1; x_2; \ldots; x_n] = \text{form}.$$

Our presentation generally prefers the latter notation. The symbol, "=" should be read "equals by definition". To illustrate, we give the name "list2" to the function of the preceding example as follows:

$$\text{list2}[x; y] = \text{cons}[x; \text{cons}[y; \text{NIL}]].$$

For later purposes, the generalization of this function to an indefinite number of arguments is given the name "list."

The composition of functions forms more functions, e.g., atom $\circ$ f represents the same function as λ[[x]; atom [f[x]]]. Sequential compositions of f and r functions are abbreviated as follows. For f $\circ$ r $\circ$ r, we write simply frr; etc., etc.

4. Conditional expressions

A conditional expression is of the form $[p_1 \rightarrow e_1; \ldots; p_k \rightarrow e_k]$ where the p_i are propositions (predicates applied to arguments) and the e_i are expressions. Such a conditional expression is evaluated as follows: starting with p_1, evaluate the propositions p_i until one is found whose value is TRUE. The value of the conditional expression is the value of the expression corresponding to p_i. If none of the propositions has the value TRUE, the value of the conditional expression is undefined. To help the reader, we begin in this section a sequence of graded examples which culminates with a differentiation procedure in section 7.

a. $\sim p$ (We use this notation as an abbreviation for not [p].) where p is a proposition

$\sim p \quad [p \rightarrow$ FALSE, TRUE $\rightarrow$ TRUE].

In words, if the value of p is TRUE, then the value of $\sim p$ is FALSE; otherwise (if the value of p is FALSE), the value of $\sim p$ is TRUE.

b. $p \,\&\, q = [p \rightarrow q;$ TRUE $\rightarrow$ FALSE].

The function, "and" can be defined with an indefinite number of arguments. Similarly, we can define the function, "or", which we denote by the symbol "V".

c. null [s] = atom [s] & eq [s; NIL].

5. Recursive function definition

A recursive function may be defined as follows:

recuf [x_1; x_2; ...; x_n] = form

where the form may contain recuf e.g.,

a. equal [x; y] = [atom[x] & atom [y] & eq [x; y]] v[~ atom [x] & ~ atom [y] & equal [f[x]; f[y]] & equal [r[x]; r[y]]].

This predicate is TRUE if and only if both of its arguments are the same s-expression.

b. pair [x; y] = [null [x] & null [y] → NIL; ~ atom [x] & ~ atom [y] → cons [list [f[x]; f[y]]; pair [r[x]; r[y]]]]

6. Functions with functions as arguments

Functions may have functions as arguments, e.g., the first three functions below, maplist, map2, and search. In preparation for the differentiation procedure of the next section, we conclude this section with the definition of the function, "sublis" which uses search. In what follows the value of the argument fn is a function.

a. maplist [s; fn] = [null [s] → NIL; TRUE → cons [fn[s]; maplist [r[s]; fn]]]

b. map2 [s1; s2; fn] = [null [s1] → NIL; TRUE → cons [fn[s1; s2]; map2 [r[s1]; r[s2]; fn]]]

c. search [s; p; fn; u] = [null [s] → u [ ];

p[s] → fn[s];

TRUE → search [r[s]; p; fn; u]]

Note that u is a function with no arguments.

d. Sublis [pairs; s] = [null [pairs] → s; null [s] → NIL;

TRUE → search [pairs; λ [[j]; equal [s; ff [j]]]; frf; λ [[ ];

[atom [s] → s;

TRUE → cons [sublis [pairs; f[s]]; sublis [pairs; r [s]]]]]]]

This function makes a list of substitutions designated by "pairs" in the expression "s".

7. Procedure for differentiating mathematical expressions

This section gives a procedure for differentiating expressions formed by the operations of PLUS and TIMES and concludes by extending this procedure to other operations so long as their gradients are supplied. The b-expressions (basic expressions) are defined recursively as follows:

a. Any atom is a b-expression.

b. If b_1, b_2, ..., b_n are b-expressions, then so are (PLUS, b_1, b_2, ..., b_n) and (TIMES, b_1, b_2, ..., b_n).

Below is a procedure for differentiating such a b-expression s with respect to the variable v.

Diff [s; v] =

[atom [s] → [eq[s; v] → 1.0; TRUE → 0.0];

```
eq [f[s]; PLUS] → cons [PLUS; maplist [r[s]; λ [[si]; diff
[f[si]; v]]]];
eq [f[s]; TIMES ] → cons [PLUS; maplist [r[s]; λ [[si]; cons
[TIMES; maplist [r[s]; λ [[sj];
   [equal [si; sj] → diff [f[sj]; v];
   TRUE → f[sj]]]]]]]]]
```

In addition to PLUS and TIMES, new operations (as the first member of a list) may be introduced provided only that the gradient of each such operation is made available to the function "gradient" whose argument is the name of an operation and whose value is the gradient of that operation. The form of such gradients is a pair of lists of equal length. The first list is a list of variables, the second list is a list of partial derivatives with respect to those variables. Suppose for example we represent U^V by (POWER, U, V) and LOG_b U by (LOG, b, U). The gradient of POWER is represented as follows, ((U, V), ((TIMES, V, (POWER, U, (PLUS, V, -1.0))), (TIMES, (LOG, E, U), (POWER, U, V)))). Our differentiation procedure can be extended to the operations whose gradients have been made available to "gradient" if we add to the definition given previously a fourth (and final) case as follows:

```
TRUE → cons [PLUS; map2 [sublis [pair [f ∘ gradient ∘ f[s];
r[s]]; fr ∘ gradient ∘ f[s]]; r[s]; λ [[j; k]; list [TIMES;
f[j]; diff [f[k]; v]]]]]
```

8. Some other symbol manipulating functions

Besides the procedures already mentioned, many other functions are available in the LISP system. Some of the functions defined by the author and used by SAINT will be found in chapter III. A few of the more important functions defined by others and used by SAINT are discussed below.

a. apply

This function is the interpreter whose inputs include an s-expression representing a function and a list of arguments for that function and whose output is the same as the value of that function when applied to those arguments.

b. distrb (abbreviation for distribute)

The value of distrb [s; p] where s is a mathematical expression and p is a predicate is an expression mathematically equivalent to s determined as follows. For each product in s, multiply out all factors which are sums and the list of whose summands satisfies p.

c. smplfy (abbreviation for simplify)

The value of smplfy [s] is a simplified expression mathematically equivalent to s. This procedure tries most of the standard algebraic simplifications, e.g., evaluation of all numerical subexpressions, collecting like terms in sums and products, etc.

B. A Symbol Manipulating Language on a Computer

This chapter concludes with a brief description of how the LISP language is implemented on a computer.

1. Representation of an s-expression in a computer

The mathematical expression xe^{x^2} is represented by the s-expression, (TIMES, X, (POWER, E, (POWER, X, 2))). How a computer represents this expression is pictured in figure 2.1. Each rectangle represents a computer memory register and each of its two component squares represents half of a register. In particular, on either the I.B.M. 709 or 7090 computer, the left and right squares represent the address and decrement of a register. The half register represented by a square occupied by an atom points to (contains the location of) the computer representation of that atom. How an atom is represented is discussed in the next section. The half registers represented by squares that are blank point to other registers, as indicated by the lines (between rectangles) which should be thought of as running to the right or down.

2. Representation of atoms, association lists and objects

In a computer, an atom (or object) is represented by an association list, i.e., a list with a special indication in the left half of the register representing its first element. The association list contains information associated with that atom, such as, its print name, its gradient if any, its function

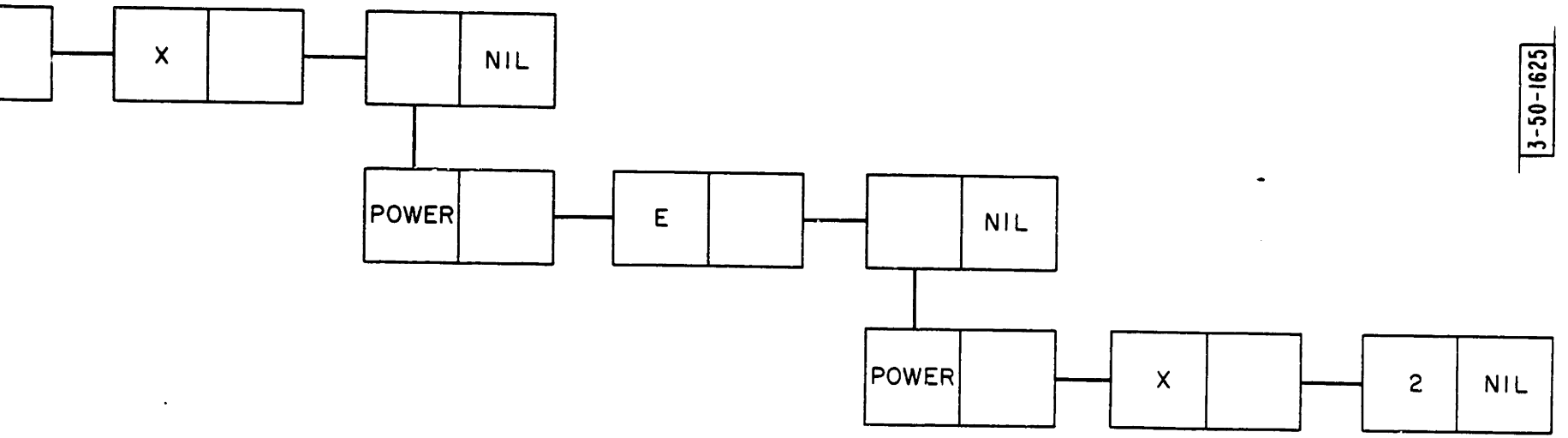


FIGURE 2.1 COMPUTER REPRESENTATION OF THE S-EXPRESSION, (TIMES, X, (POWER, E, (POWER, X, 2)))

definition if any, etc. SAINT's goals and subgoals discussed in chapter IV are objects.

3. The push-down list

The push-down list is contained in a certain sequence of computer registers which hold the information needed for the execution of the recursive functions. During computer operation, the amount of push-down list acutally being used is a rough measure of the depth or complexity of the present computation. If the maximum allowable length (typically a thousand) of the push-down list is exceeded, the computer reports this fact and stops.

4. The free-storage list, the cons counter and the reclaimer

The execution of a cons function consumes a computer memory register, which is taken from the free-storage list, a list of available registers. A count is made of how many cons functions have been executed, and the contents of this "cons counter" may be printed out at any time during the computation. When the free-storage list is about to be exhausted, a program called the reclaimer returns all abandoned registers to the free-storage list.

Chapter III

Some Prerequisite Procedures for SAINT

The author had to construct many procedures to enable the computer to perform various skills which are prerequisite for the performance of symbolic integration. A few of these functions are described in this chapter in order that the reader might become more familiar with the problems of programming symbol manipulation. From this chapter, the reader must know only the notation of Section E in order to follow the development in later chapters.

A. Procedures for Inserting Elements in Ordered Lists and for Sorting Lists

B. Set Theoretic Functions

A set is represented by a list of its members, e.g., the list (B,(C,D)) represents the set which consists of two members, namely, B and (C, D). In addition to functions for taking unions, intersections and power sets, the author defined predicates for telling when an s-expression is a member of a set and when a set is a subset of another set.

C. The Universal and Existential Quantifiers for Finite Sets

Duality is used to define the existential quantifier in terms of the universal quantifier below.

1. "forall" is a predicate. The value of forall [s; p] where s represents a set and p is a predicate, is TRUE if and only if every member of s satisfies p.

forall [s; p] = null [s]V [p o f [s] & forall [r[s]; p]]

2. therex [s; p] = ~ forall [s; λ [[si]; ~ p [si]]]

D. Procedures for Manipulating Polynomials

The author constructed a predicate which tests an expression for being a polynomial and some functions for finding the degree of a polynomial and for adding, subtracting, multiplying, dividing and expanding polynomials.

E. Recursive Matching Procedure

For many types of symbol manipulating tasks including symbolic integration, recursive matching, i.e., the ability to recognize when a given expression is an instance of a specified pattern or form is important, e.g., the ability to recognize when an integrand is of standard form or an instance of a pattern which suggests certain methods. This pattern recognition function in SAINT corresponds to "matching" in the LOGIC THEORIST of Newell, Shaw, and Simon [9]. To illustrate, SAINT requires a function to answer questions such as the following: "For what values of the constants m and n (where $m \neq \pm n$), is sin x cos 2x an instance of the pattern sin mx cos nx?" The value of the function should be "when m = 1

and n = 2". The reader should note several things about this example. First note that the value assigned to each of the form variables (m and n) must satisfy a predicate, namely, "is constant". Also notice that the values paired with the form variables must satisfy a relation, namely, $m \neq \pm n$. This pattern recognition procedure of SAINT can handle such predicates and relations. SAINT can also handle some important recursive patterns, such as, rational functions and elementary functions, e.g., "rational function of sines and cosines", denoted by raf (sin v, cos v) and "elementary function of secants and squares of tangents", denoted by elf (sec v, $\tan^2 v$). Other forms which this powerful procedure can recognize will be found later, especially in Chapter IV. The precise definition of this function is very lengthy and is omitted. However, the above examples suffice to give the reader a fairly good idea of the power of this function. The value of the function when applied to its arguments is NIL if and only if there is no pairing of form variables and expressions which makes the given elementary expression an instance of the specified elementary pattern or form; otherwise, its value is a list of all such enabling pairings.

F. Equation Solving Procedure

The author designed a function which will solve for any variable v an equation of the form g(v) = c where c is a constant expression and g(v) is an elementary expression in

which v occurs once. To illustrate, if this function were given the problem to solve for x the following equation arccsc $\log_2$ (c + 5 + tan x) = arccsc 3, its value would be arctan (4 - c).

Chapter IV

The Symbolic Integration Procedure of SAINT

This chapter describes a procedure and its computer program realization for performing formal integration of explicit elementary functions of a real variable. We believe that this process is similar to that used by college freshmen. The procedure employs heuristic methods and succeeds in a reasonably large class of problems. For complete understanding of this chapter, the reader should be familiar with elementary methods of formal integration as found in any standard first year calculus text. Other readers may get the general idea by thinking in terms of some other more familiar problem domain, e.g., some solitaire card game.

A. Indefinite Integration Procedure

This section describes how SAINT performs indefinite integration and concludes with an example. An attempt is made to orient the reader before a detailed description of the procedure is given. The executive organization of SAINT is similar to that of the LOGIC THEORIST of Newell, Shaw and Simon. The "try for an immediate solution" mentioned twice in Figure 4.4 may be described roughly as follows: As soon as a new goal g is generated, SAINT uses its straightforward methods in an attempt to achieve it. While doing this, SAINT may add g or certain of

g's subgoals to the "temporary goal list". If g is achieved, an attempt is made to achieve the original goal. It will help to take a preview of section 14 (especially figure 4.4) before beginning the more detailed description below.

1. Goals

Throughout the expcsition of this procedure, "goal" is used interchangeably with "problem", with "goal" generally preferred. In each application of the present procedure, the solutions of certain problems, namely, performing integrations with side conditions, are goals. How goals are generated, manipulated, and achieved is described later. For now, let us limit ourselves to describing what we shall call the "original goal" which consists of the originally given integration request (integrand and variable of integration) and a side condition which we shall call the "resource allotment". The role of the resource allotment is described in Section 7, and the resource allotment for an integration will be mentioned only when it is relevant.

2. The goal list

The original goal is made the first member of a list called the goal list. From time to time new goals may be generated. Each newly generated goal is added to the end of the goal list.

3. Standard forms

Whenever an integrand of a newly generated goal is of "standard form", that goal is immediately achieved by substitution. An

integrand is said to be of standard form when it is a substitution instance of one of a certain set of forms. The use of standard forms in SAINT is analogous to applying the method of substitution in the LOGIC THEORIST of Newell, Shaw, and Simon. Below are listed the 26 standard forms currently used in this procedure, together with the corresponding form of the solution.

The standard forms (and the algorithm-like and heuristic transformations to be described later) are ordered with some care. Thus, for example, in standard form "u" there is no need to specify that $c \neq -1$ since "t" (which precedes "u") would already have caught this case.

a. $\int c\ dv = cv.$

b. $\int e^v\ dv = e^v.$

c. $\int c^v\ dv = \frac{c^v}{\ln c}.$

d. $\int \ln v\ dv = v \ln v - v.$

e. $\int \log_c v\ dv = v \log_c v - \frac{v}{\ln c}.$

f. $\int \sin v\ dv = -\cos v.$

g. $\int \cos v\ dv = \sin v.$

h. $\int \tan v\ dv = \ln \sec v.$

i. $\int \cot v \, dv = \ln \sin v.$

j. $\int \sec v \, dv = \ln(\sec v + \tan v).$

k. $\int \csc v \, dv = \ln(\csc v - \cot v).$

l. $\int \arcsin v \, dv = v(\arcsin v) + \sqrt{1 - v^2}.$

m. $\int \arccos v \, dv = v(\arccos v) - \sqrt{1 - v^2}.$

n. $\int \arctan v \, dv = v(\arctan v) - \frac{1}{2} \ln (1 + v^2).$

o. $\int \operatorname{arccot} v \, dv = v(\operatorname{arc\,cot} v) + \frac{1}{2} \ln (1 + v^2).$

p. $\int \operatorname{arcsec} v \, dv = v(\operatorname{arcsec} v) - \ln (v + \sqrt{v^2 - 1}).$

q. $\int \operatorname{arccsc} v \, dv = v(\operatorname{arccsc} v) + \ln (v + \sqrt{v^2 - 1}).$

r. $\int \sec^2 v \, dv = \tan v.$

s. $\int \csc^2 v \, dv = - \cot v.$

t. $\int \frac{dv}{v} = \ln v.$

u. $\int v^c \, dv = \frac{v^{c+1}}{c+1}$.

v. $\int \sec v \tan v \, dv = \sec v$.

w. $\int \csc v \cot v \, dv = - \csc v$.

$m \neq \pm$ in "x", "y" and "z".

x. $\int \sin mv \cos nv \, dv = - \frac{\cos(m-n)v}{2(m-n)} - \frac{\cos(m+n)v}{2(m+n)}$.

y. $\int \sin mv \sin nv \, dv = \frac{\sin(m-n)v}{2(m-n)} - \frac{\sin(m+n)v}{2(m+n)}$.

z. $\int \cos mv \cos nv \, dv = \frac{\sin(m-n)v}{2(m-n)} + \frac{\sin(m+n)v}{2(m+n)}$.

If the original integrand, i.e., the integrand of the original goal, is of standard form, it is easily integrated. As an illustration, to perform $\int \frac{dx}{x}$, this original goal is made the first member of the goal list; it is seen to be of standard form, namely, standard form "t"; hence the solution is ln x.

4. Algorithm-like transformations

Whenever an integrand is found to be not of standard form,

it is tested to see if it is amenable to an algorithm-like transformation. By an algorithm-like transformation is meant a transformation which, when applicable, is always or almost always appropriate. For a goal, a transformation is called appropriate if it is the correct next step to bring that goal nearer to achievement. Below are listed the eight algorithm-like transformations used in SAINT.

a. Factor constant, i.e., $\int c\ g(v)\ dv = c \int g(v)\ dv$.

b. Negate, i.e., $\int -\ g(v)\ dv = -\int g(v)\ dv$.

c. Decompose, i.e., $\int \Sigma g_i(v)\ dv = \Sigma \int g_i(v)\ dv$.

d. Linear substitution, i.e., if the integral is of the form $\int$ elf $\{c_1 + c_2 v\}\ dv$, substitute $u = c_1 + c_2 v$, and obtain an integral of the form $\int \frac{1}{c_2}$ elf $\{u\}\ du$, e.g., in $\int \frac{\cos 3x}{(1 - \sin 3x)^2}\ dx$ (1 Jan. '53, 1b 2) substitute $y = 3x$.

e. Expand, i.e., $\int [\Sigma g_i(v)]^n\ dv$ (where n is a positive integer) = integral of the expansion.

f. Combine factors, i.e., if possible, combine factors of the numerator and denominator of the integrand. So far, SAINT can factor only by finding common factors in sums. This transformation will cancel common factors from the numerator

and denominator. As an illustration, for the integrand $\frac{x}{x^2 + x}$, factor the denominator and combine the x terms which will result in cancelling an x from the numerator and denominator.

g. Divide polynomials, i.e., if $P_1(v)$ and $P_2(v)$ are polynomials such that the degree of $P_2(v)$ does not exceed the degree of $P_1(v)$, then

$$\int \frac{P_1(v)}{P_2(v)}\, dv = \int \left[Q(v) + \frac{R(v)}{P_2(v)} \right] dv,$$

where $Q(v)$ and $R(v)$ are the polynomial quotient and remainder respectively obtained by fully dividing out the fraction $\frac{P_1(v)}{P_2(v)}$.

h. Half angle identities, i.e., an integrand of the form $\sin^m v \cos^n v$ where m and n are even nonnegative integers should be transformed to one of the following forms:

(1) $\left(\frac{1}{2} \sin 2v\right)^m \left(\frac{1}{2} + \frac{1}{2} \cos 2v\right)^{\frac{n-m}{2}}$ if $m < n$.

(2) $\left(\frac{1}{2} \sin 2v\right)^n \left(\frac{1}{2} - \frac{1}{2} \cos 2v\right)^{\frac{m-n}{2}}$ if $m \geq n$.

These two trigonometric identities are easily obtained by combining the trigonometric identity

$\sin v \cos v = \frac{1}{2} \sin 2v$

with each of the two identities

$\cos^2 v = \frac{1}{2} + \frac{1}{2} \cos 2v$

$\sin^2 v = \frac{1}{2} - \frac{1}{2} \cos 2v$

5. The goal tree

When a heuristic transformation (to be described in Section 11) or an algorithm-like transformation is applied to a goal, new goals are generated. These goals, in turn, may generate more goals, and a certain hierarchy is created. Such a hierarchy is conveniently represented by a graph or tree growing downwards. To facilitate understanding, the terminology of ordinary and family trees is adopted by analogy, e.g., pruning, alive, dead, child, parent, descendant, ancestor, etc.

Suppose we have an integration to perform, or more generally, any goal g which we shall represent graphically by a point. A goal may be transformed into one or more subgoals which may be related to the goal in many ways. This integration procedure incorporates two common relations, namely, AND and OR.

a. AND relationship

An AND relationship is created when two or more subgoals are generated and the achieving of all of them is required to achieve the goal. Fig. 4.1 depicts a relationship with three subgoals. The arc joining the three branches denotes the AND relationship.

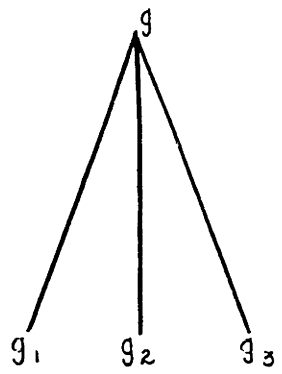


Figure 4.1 An AND relationship

To illustrate this, suppose the original goal is $\int(2 + \cos x)\, dx$. As always, this original goal is made the first goal on the goal list. Next it is determined to be not of standard form. The first two algorithm-like transformations don't apply, but algorithm-like transformation "c" succeeds, and adds to the goal list two new goals, namely, $\int 2dx$ and $\int \cos x\, dx$. See Fig. 4.2a. Next the second goal on the goal list is seen to be of standard form, namely, standard form "a". The black dot in Figure 4.2b denotes that this subgoal is achieved. Then it is determined that the achieving of this subgoal does not suffice to achieve the goal. Finally the third goal on the goal list is seen to be of standard form, (namely, standard form "g") and hence to have the integral, sin x. With the

achieving of this subgoal, we can and do achieve the original goal $\int(2 + \cos x)dx = 2x + \sin x$.

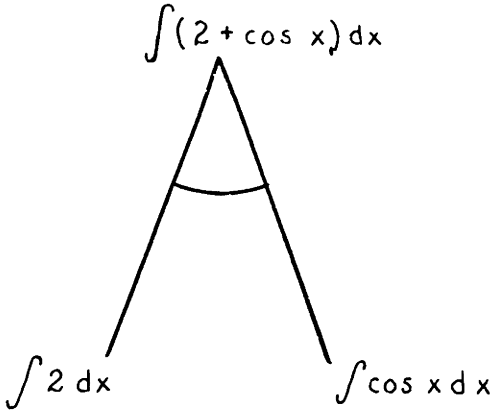


Figure 4.2a

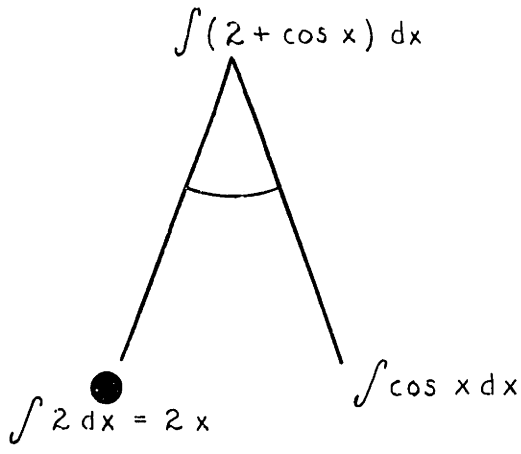


Figure 4.2b

b. OR relationship

An OR relationship between a goal and its subgoals exists when the achieving of any subgoal will allow the achieving of the goal. Examples of this will appear later. From these two basic relationships, more complicated relationships among goals may be built up, e.g., see Figs. 4.3a and b in section 12.

6. The temporary goal list

The first attempt on new goals is performed by the procedure "imsln" described in Section 13 below. Any goal encountered by imsln which is neither of standard form nor amenable to an algorithm-like transformation is added to the end of the

"temporary goal list" (not to be confused with the "goal list") and later transferred to the "heuristic goal list" described in Section 10 below. If the goal were added directly to the heuristic goal list rather than to the temporary goal list, time might be wasted by finding the goal's character (cf. Section 8) as described in Section V C.

7. The role of the resource allotment

The resource allotment is a side condition of the original goal. Before proceeding to apply heuristic transformations, it must be verified that the resource allotment has not been exceeded. If the resource allotment has been exceeded, SAINT reports this fact as its final answer. Although other kinds of resources, e.g., time, should also be considered, the only kind of resource that is handled by SAINT is the total amount of work space. For hand simulation, the work space can be measured by the number of pages or by the number of lines used for the final and all intermediate results. For implementation in the LISP language on a machine, the work space is measured by the number of "construct" functions which have been executed.

8. Character of a goal

When a goal is taken off the temporary goal list its "character" is obtained, i.e., an ordered list of

"characteristics." A characteristic of a goal is a feature which might be useful either in estimating the cost of attempting its attainment or in selecting appropriate heuristic transformations (see Section 11). In SAINT, the character is composed of the following eleven characteristics of the integrand:

a. Depth

The depth of a symbolic expression s is the maximum level of parentheses which occurs in that expression. As one might guess, this helps us get a crude estimate of the problem's difficulty. The following computational procedure is its formal definition.

depth [s] =

[atom [s] → 0.0;

TRUE → max [[1.0 + depth ∘ f[s]]; depth ∘ r[s]]].

b. Length

The length of a symbolic expression is the number of elements in the list that represents it. This was supposed to serve the same kind of purpose as depth in Section "a", but we never got around to using it.

c. Rational function?

The value of this predicate is TRUE if and only if the integrand represents a rational function.

d. Algebraic function?

Similar to c.

e. Exponent base

Let v be the variable of integration. The exponent base of an integrand is b or NIL according to whether or not there is some constant b and some integers $n_1, n_2, \ldots, n_k$, such that the integrand is of the form

elf $\{b^{n_1 v}, b^{n_2 v}, \ldots, b^{n_k v}\}$, i.e., formed by elementary operations from expressions of the form b^{nv} where n is any integer. For example, the exponent base of $\frac{e^{3x}}{1 - e^{2x}}$ is e, whereas, the exponent base of $\frac{e^{3x}}{1 - 2^{2x}}$ is NIL as is the exponent base of $\frac{e^{3x}}{x - e^{3x}}$.

f. Rational function of sines and cosines?

The value of this predicate is TRUE if and only if the integrand is of the form raf {sin v, cos v}, i.e., formed by rational operations on sines and cosines.

g. Elementary function of sines and cosines?

h. Elementary function of secants and tangents?

i. Elementary function of cosecants and cotangents?

j. Elementary function of trigonometric functions?

The value of this predicate is TRUE if and only if the

integrand is of the form elf {sin v, cos v, tan v, cot v, sec v, csc v}.

k. Partial?

The value of this predicate is TRUE if and only if either the main connective of the integrand is logarithmic or inverse trigonometric, or the integrand is not an algebraic function and is the product of factors, not all of the same type. This predicate is useful in deciding when to use the method of integration by parts.

The types of factors are:

(1) Inverse trigonometric, when the factor is either an even quadratic raised to some power, e.g., $1 - 2x^2$ or $(3 + cx^2)^s$, or an expression which contains an inverse trigonometric function of the variable of integration, e.g., $(2 + \arctan x)^3$.

(2) Logarithmic, when the factor is either $\frac{1}{v}$ or an expression which contains log v.

(3) Exponential, when the factor contains an expression of the form c^v.

(4) Trigonometric, when the factor contains a trigonometric function of the variable of integration.

(5) NIL, when the factor is not of any of the preceding four types.

Thus, xe^x is partial since it is the product of a factor of type NIL and one of exponential type. However, $\frac{\log x}{x}$ is not partial since, although it is a product, both of its factors are of the same type, logarithmic.

As an illustration of all eleven characteristics, the character of $(\sec x)^{-1}$ is (2, 3, FALSE, FALSE, NIL, FALSE, FALSE, TRUE, FALSE, TRUE, FALSE), i.e., its depth is 2; its length is 3; it is not a rational function; etc.

9. Relative cost estimate

Although other estimates could and should be tried, we take for the relative cost estimate of a goal, simply the depth of its integrand (first characteristic). This makes use of the fact that, ordinarily, the deeper the integrand the more will be the resources needed to investigate that goal.

10. The heuristic goal list

A list of goals requiring heuristic transformations, or, more briefly, a heuristic goal list, is an ordered list of those goals which are neither of standard form nor amenable to an algorithm-like transformation. A member of the heuristical list is called a heuristic goal. New such goals are inserted in order of increasing relative cost estimate.

11. The heuristic transformations

A transformation of a goal is called heuristic when, even though it is applicable and plausible, there is a significant risk that it is not the appropriate next step. A transformation may be inappropriate either because it leads no closer to the solution or because some other transformation would be better. The heuristic transformations are analogous to the methods of detachment, forward chaining and backward chaining in the LOGIC THEORIST of Newell, Shaw and Simon. Below are listed the ten heuristic transformations used in SAINT. With each such transformation is given an example of a case in which the transformation is inappropriate.

a. Transformation of an elementary function of trigonometric functions

An integrand (which was not itself generated by a transformation of this type) and which is an elementary function of trigonometric functions, i.e., is of the form elf {sin v, cos v, tan v, cot v, sec v, csc v} is transformed into two or three of the following forms:

(1) elf $\{\sin v, \cos v, \frac{\sin v}{\cos v}, \frac{\cos v}{\sin v}, \frac{1}{\cos v}, \frac{1}{\sin v}\}$

when the integrand is not already an elementary function of sines and cosines (seventh characteristic);

(2) elf $\left\{ \frac{\tan v}{\sec v}, \frac{1}{\sec v}, \tan v, \frac{1}{\tan v}, \sec v, \frac{\sec v}{\tan v} \right\}$ when the integrand is not already an elementary function of secants and tangents (eigth characteristic);

(3) elf $\left\{ \frac{1}{\csc v}, \frac{\cot v}{\csc v}, \frac{1}{\cot v}, \cot v, \frac{\csc v}{\cot v}, \csc v \right\}$ when the integrand is not already an elementary function of cosecants and cotangents (ninth characteristic).

For example, heuristic a(1) is appropriate when it transforms $\int \frac{dx}{\sec x}$ into $\int \cos x \, dx$ (which is then easily integrated since it is a standard form). The heuristic is inappropriate for the $\int \frac{\sec^2 t}{1 + \sec^2 t - 3 \tan t} dt$ (2 June '59, 1b) for which heuristic transformation b (3) leads directly to a solution.

b. Substitution for a trigonometric function

The form of the integrand often suggests a substitution for a particular trigonometric function.

(1) When the integrand is an elementary function of sines and cosines (seventh characteristic) and, in fact, is of the form elf$\{\sin v, \cos^2 v\} \cos^{2n+1} v$ where elf$\{\sin v, \cos^2 v\}$ is an elementary function of sines and even powers of cosines and where n is an integer. Try substituting $u = \sin v$ which transforms the goal into the often simpler subgoal,

$\int$elf {u, 1 - u^2} (1 - u^2)^n du.

(2) Similarly, when the integrand is of the form elf {cos v, $\sin^2 v$} $\sin^{2n+1} v$, try u = cos v.

(3) Similarly, when the integrand is of the form elf {tan v, $\sec^2 v$} try u = tan v.

(4) When the integrand is of the form elf {cot v, $\csc^2 v$} try u = cot v.

(5) When the integrand is of the form elf {sec v, $\tan^2 v$} $\tan^{2n+1} v$, try u = sec v.

(6) When the integrand is of the form elf {csc v, $\cot^2 v$} $\cot^{2n+1} v$, try u = csc v.

For example, in $\int \frac{\sec^2 t}{1 + \sec^2 t - 3 \tan t}\, dt$, (2 June '59, 1b) heuristic transformation b(3) is appropriate when it suggests x = tan t.

However, heuristic b(3) is inappropriate for $\int \frac{dx}{\sec^2 x}$ which is best handled by heuristic transformation a(1).

c. Substitution for a subexpression whose derivative divides the integrand

Let g(v) be the integrand. For each nonconstant nonlinear subexpression s(v) such that neither its main connective is MINUS nor is it a product with a constant factor and such that

the number of nonconstant factors of the fraction $\frac{g(v)}{s'(v)}$ (after cancellation) is less than the number of factors of $g(v)$, try substituting $u = s(v)$. Thus, in $\int xe^{x^2} dx$, substitute $u = x^2$, (When SAINT actually tried this problem it used this heuristic but surprised me by substituting $u = e^{x^2}$, which is somewhat better.) In $\int \frac{x}{\sqrt{x^2 + 2x + 5}} dx$ the substitution $u = x^2$, suggested by this heuristic, is inappropriate; heuristic transformation "e" is appropriate.

d. Integration by parts

For each partition into a product of two factors $G \cdot h$ of an integrand which is partial (eleventh characteristic) such that $G \neq 1$ and such that this procedure, with the allotment of no resources, finds $\int h\, dv = H$, try integration by parts, i.e.,

$$\int Gh\, dv = GH - \int \frac{dG}{dv} H\, dv, \text{ e.g., } \int \frac{xe^x}{(1 + x)^2} dx =$$

$$- \frac{xe^x}{1 + x} - \int - e^x dx. \quad \text{In } \int (\arcsin x + \sin x)^2 ([1 - x^2]^{-\frac{1}{2}} + \cos x)\, dx,$$

integration by parts is inappropriate; heuristic transformation "c" is appropriate.

e. Elimination of the middle term of a quadratic subexpression

For each quadratic subexpression, $q(v) = c_3 + c_2 v + c_1 v^2$

where c_1 and c_2 are nonzero constants, eliminate the middle term by the substitution $u = v + \frac{c_2}{2c_1}$ which transforms $q(v)$ into $c_3 - \frac{c_2^2}{4c_1} + c_1u^2$, e.g., $\int \frac{x}{\sqrt{x^2 + 2x + 5}}\,dx = \int \frac{u - 1}{\sqrt{u^2 + 4}}\,du$,

which should be next transformed by heuristic "f". In $\int \frac{x^2 + x}{\sqrt{x}}\,dx$, this transformation is inappropriate; heuristic transformation "f" is appropriate.

f. Distribution of nonconstant sums

If at least one nonconstant sum occurs as a factor of a product in the integrand, try transforming the integrand by distributing the products with respect to all such sums, e.g., $\int \frac{x^2 + x}{\sqrt{x}}\,dx = \int (x^{\frac{3}{2}} + x^{\frac{1}{2}})\,dx$. In $\int \frac{x + 1}{\sqrt{2x - x^2}}\,dx$, this heuristic is inappropriate; heuristic transformation "e" is appropriate.

g. Trigonometric substitution

For each quadratic subexpression of the form $c_2 + c_1v^2$, where c_1 and c_2 are nonzero constants:

(1) If both c_1 and c_2 are positive, then try the substitution $v = \sqrt{\frac{c_2}{c_1}} \tan u$, which replaces the quadratic by $c_2 \sec^2 u$.

(2) If c_1 is negative and c_2 is positive, then try the substitution $v = \sqrt{\frac{c_2}{-c_1}} \sin u$ which replaces the quadratic by $c_2 \cos^2 u$.

(3) If c_1 is positive and c_2 is negative, try the substitution $v = \sqrt{\frac{-c_2}{c_1}} \sec u$ which replaces the quadratic by $-c_2 \tan^2 u$.

For example, heuristic transformation g(2) is appropriate when it transforms $\int \frac{x^4}{(1 - x^2)^{5/2}} dx$ (2 May '57, 3c) $\int \frac{\sin^4 u}{\cos^4 u} du$ by substituting $x = \sin u$. However it is inappropriate in $\int \frac{x^2 + 1}{\sqrt{x}} dx$; heuristic transformation "f" is appropriate.

h. Expansion of positive integer powers of nonconstant sums

If the integrand contains at least one nonconstant sum raised to an integer power $n > 1$, try expanding all such sums, e.g., $\int x(x^{\frac{1}{2}} + x^{-\frac{1}{2}})^2 dx = \int x(x + 2 + x^{-1}) dx$. This heuristic transformation is inappropriate for $\int (\sin^2 x + 1)^2 \cos x \, dx$; heuristic transformation b(1) is appropriate.

i. Exponent base

Suppose that the integrand has an exponent base b which is not NIL, i.e., has the form elf $\{b^{n_1 v}, b^{n_2 v}, \ldots, b^{n_k v}\}$ where $n_1, n_2, \ldots, n_k$ are integers. Let g be the greatest common divisor of these integers. Try the substitution $u = b^{gv}$,

e.g., $\int \frac{e^{6x}}{e^{4x} + 1}\, dx = \int \frac{u^2}{2(u^2 + 1)}\, du$. This heuristic transformation is inappropriate for $\int e^{2x} \ln (1 + e^{2x})\, dx$; heuristic transformation "c" is appropriate with the substitution $u = 1 + e^{2x}$.

j. Rational function of sines and cosines

When the integrand is a rational function of sines and cosines (sixth characteristic), try substituting $u = \tan \frac{v}{2}$ which replaces sin v by $\frac{2u}{1 + u^2}$ and replaces cos v by $\frac{1 - u^2}{1 + u^2}$ and replaces dv by $\frac{2}{1 + u^2}\, du$. This substitution transforms the integrand into a rational function, e.g., $\int \frac{dx}{1 + \cos x} = \int du$.

This heuristic is inappropriate for $\int \frac{dx}{\cos^2 x}$; heuristic transformation a(2) is appropriate.

12. Pruning the goal tree

Whenever some goal g has been achieved, the goal tree is pruned, i.e., certain closely related goals are automatically achieved and certain other goals newly rendered superfluous, are killed. The process of pruning with respect to a newly achieved goal g, represented by a predicate, prune [g], is recursive and is described as follows.

a. If g is the original goal, the original problem is solved and the value of prune [g] is TRUE.

b_1. Otherwise, kill g and kill every descendant of g which is thus rendered superfluous, i.e., which then no longer has a direct line of living ancestors generated from the original goal.

b_2. Achieve and prune any of g's parents which have become achievable from the achieving of g.

The pruning procedure will be clarified by an example. In Fig. 4.3a the achieving of g_{221} allows g_{22} to be achieved, (since, as indicated by the black dot, g_{222} has already been achieved). In turn, the achieving of g_{22} allows g_2 to be achieved (since there is an OR relationship). Since the achieving of g_2 now has rendered g_{23} superfluous, it is killed. However, another of g_2's children g_{12} is not killed since, through its other parent g_1 it has direct living ancestry to the original

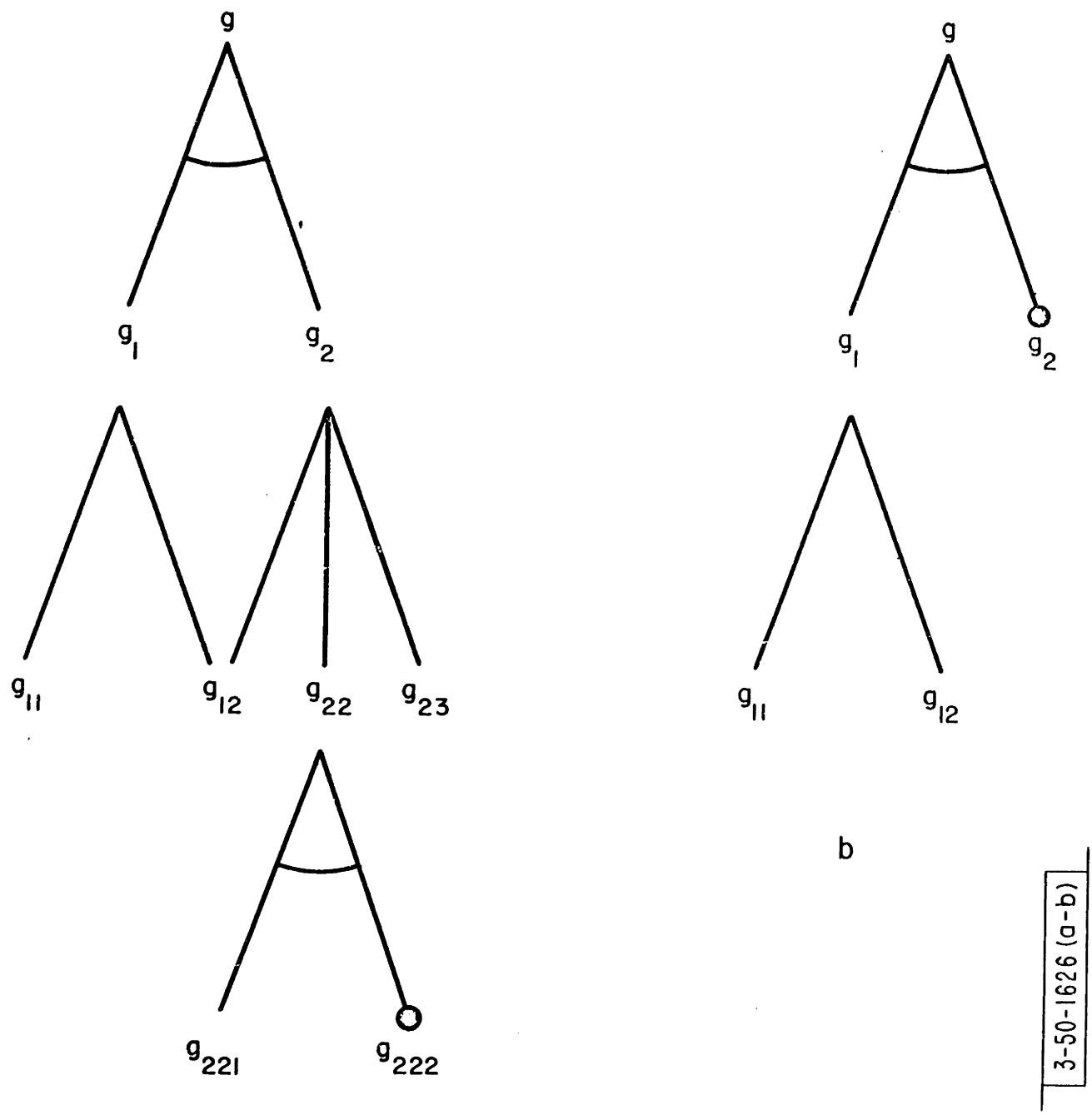


FIGURE 4.3 THE PRUNING OF THE GOAL TREE a AFTER THE ACHIEVING OF THE GOAL g_{221} RESULTS IN THE GOAL TREE b. LATER ACHIEVING OF g_{11} OR g_{12} WOULD RESULT IN THE ACHIEVING OF THE ORIGINAL GOAL g

goal g. The original goal g cannot be achieved from the achieving of g_2 since there is an AND relationship and g_1 has not yet been achieved. Therefore, the result of the pruning process is as shown in Fig. 4.3b. If either g_{11} or g_{12} is later achieved, the original goal could and would be achieved.

13. Trying for an immediate solution

As soon as a new goal g is generated, SAINT uses its straightforward methods in an attempt to achieve it. While doing this, SAINT may add g or certain of g's subgoals to the temporary goal list. If g is achieved, an attempt is made to achieve the original goal. This is embodied in the following iterative procedure, imsln [s] "immediate solution" where s is some final segment of the goal list. In general, the final segment of a list $(g_1, g_2, \ldots, g_n)$ is either the empty list or one of the n lists $(g_i, g_{i+1}, \ldots, g_n)$ for each i = 1, 2, ..., n. During the execution of this "imsln" procedure, any goals appended to the goal list will also be appended to the final segment. The initial value of s is (g) where g is either the original goal or a goal generated by a heuristic transformation. Below is given the iterative procedure imsln [s].

a. If s is empty, return with FALSE.

b. Let us consider the goal g_i, the first member of s. If g_i is dead, take imsln $\circ$ r[s] , i.e., delete g_i the first member of s and go to step "a".

c. If g_i is the same as some other unachieved goal, h, which precedes g_i on the goal list, then make the parent of g_i another parent of h and calculate imsln $\circ$ r[s].

d. If g_i is directly achievable either because it is the same as some previously achieved goal or because it is of standard form, achieve it. Then, if pruning with respect to g_i is TRUE, terminate with TRUE; otherwise calculate imsln $\circ$ r[s].

e. If some algorithm-like transformation is appropriate for g_i, apply it, kill g_i and calculate imsln $\circ$ r[s].

f. Otherwise, append g_i to the end of the temporary goal list and calculate imsln $\circ$ r[s].

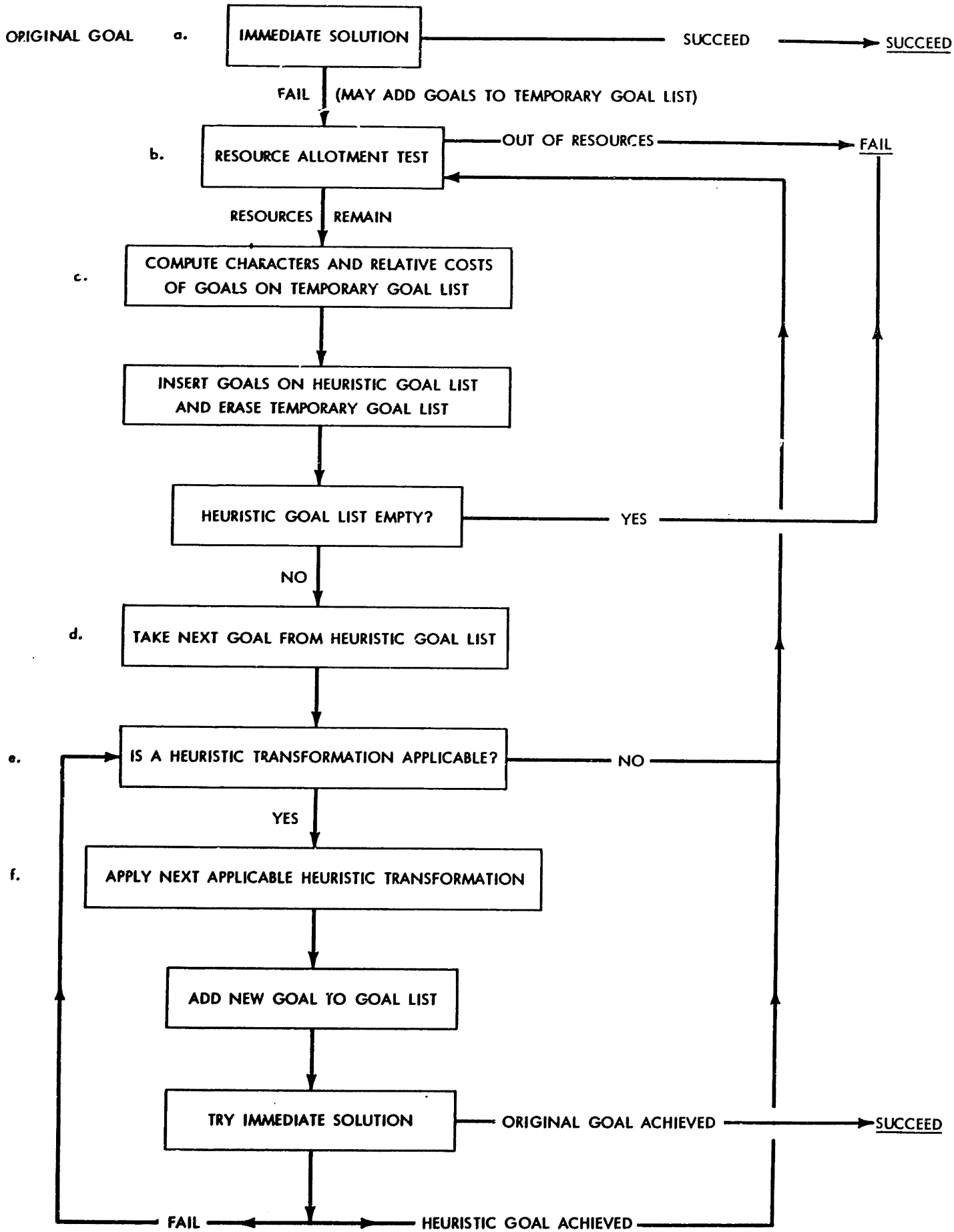
The procedure is well illustrated in finding $\int x(x + 1)\, dx$. This problem is made the first goal on the goal list. Next, we try for an immediate solution with the entire goal list, i.e., imsln $[(\int x(x + 1)dx)]$. Steps "a" through "d" are uneventful. However, in step "e" algorithm-like transformation "g" generates a new problem $\int (x^2 + x)\, dx$ which is appended to the goal list. The original problem is killed. Next we try to obtain imsln $[(\int (x^2 + x)dx)]$. Again steps "a" through "d" are uneventful, but, in step "e", algorithm-like transformation

"c" generates two new problems, $\int x^2 dx$ and $\int x dx$ which are added to the goal list. The goal, $\int (x^2 + x)dx$, is killed. Next we consider imsln [($\int x^2 dx$, $\int x dx$)]. This time, steps "a" through "c" are uninteresting. In step "d", we find that $\int x^2 dx$ is directly achievable since it is of standard form. We achieve it, i.e., $\int x^2 dx = 1/3\ x^3$. Since prune [$\int x^2 dx$] is FALSE (because the other half of the AND has not yet been achieved), we next compute imsln [($\int x dx$)]. Again steps "a" through "c" are uninteresting but, again according to step "d", we achieve the goal $\int x dx = \frac{1}{2} x^2$. Since prune [$\int x dx$] is TRUE the value of the entire imsln procedure is TRUE. The original goal is then easily achieved, $\int x(x + 1)dx = \frac{1}{3} x^3 + \frac{1}{2} x^2$.

14. Executive organization

Precisely how all the various elements 1 through 13 are pieced together to form an integration procedure is described below. The original goal is given as a triplet, namely, the integrand, the variable of integration and the resource allotment. The procedure is as follows:

a. If a try for an immediate solution with the original goal is successful, return with the answer, the actual indefinite integral.


ORIGINAL GOAL
a.
IMMEDIATE SOLUTION
SUCCEED
SUCCEED
FAIL
(MAY ADD GOALS TO TEMPORARY GOAL LIST)
b.
RESOURCE ALLOTMENT TEST
OUT OF RESOURCES
FAIL
RESOURCES
REMAIN
c.
COMPUTE CHARACTERS AND RELATIVE COSTS OF GOALS ON TEMPORARY GOAL LIST
INSERT GOALS ON HEURISTIC GOAL LIST AND ERASE TEMPORARY GOAL LIST
HEURISTIC GOAL LIST EMPTY?
YES
NO
d.
TAKE NEXT GOAL FROM HEURISTIC GOAL LIST
e.
IS A HEURISTIC TRANSFORMATION APPLICABLE?
NO
YES
f.
APPLY NEXT APPLICABLE HEURISTIC TRANSFORMATION
ADD NEW GOAL TO GOAL LIST
TRY IMMEDIATE SOLUTION
ORIGINAL GOAL ACHIEVED
SUCCEED
FAIL
HEURISTIC GOAL ACHIEVED

b. If the resource allotment has been exceeded, report failure.

c. Obtain and associate with each goal on the temporary goal list, its character and relative cost estimate. Take the goals off the temporary goal list, and insert each one in the heuristic goal list according to its relative cost estimate. If no goals remain on the heuristic goal list, report failure.

d. Take the next goal g_i off the heuristic goal list, and let it be the goal under consideration in the following inner loop.

e. If no heuristic transformations applicable to g_i remain, go to step "b".

f. Apply the next heuristic transformation applicable to g_i. As soon as a new goal g is so generated, add it to the goal list, and try for an immediate solution with g. Then there are three cases. If this try achieves the original goal, return with the answer. Failing this, if g_i is achieved, go to step "b". Otherwise go to step "e".

15. An example

In this section the integration procedure is illustrated by working an example in detail; the reader is urged to apply the procedure to solve some other problems, e.g.,

$\int \sin^2 x \cos x \, dx$ and $\int \frac{x^2 + x}{\sqrt{x}} \, dx$. Suppose we wish to find $\int \frac{x^4}{(1 - x^2)^{5/2}} \, dx$ (2 May '57, 3c). As in step 14a, after the goal list becomes $(\int \frac{x^4}{(1 - x^2)^{5/2}} \, dx)$ a try is made for an immediate solution with respect to the goal list.

In finding imsln $[(\int \frac{x^4}{(1 - x^2)^{5/2}} \, dx)]$, steps 13a to 13e are uneventful. As directed in step 13f, this original goal is made the first (and only) member of the temporary goal list. Since the rest of the goal list is empty, the value of imsln $[(\int \frac{x^4}{(1 - x^2)^{5/2}} \, dx)]$, is the same as the value of imsln $[(\,)]$ which, by step 13a, is FALSE.

Hence we enter the loop starting with step 14b, which step is uneventful. As directed in step 14c, we obtain and associate with the goal $\int \frac{x^4}{(1 - x^2)^{5/2}} \, dx$, its character, namely (5, 3, FALSE, TRUE, NIL, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE) and its relative cost estimate, namely, 5. This goal is then taken off the temporary goal list and made the first (and only) member of the heuristic goal list. As directed in step 14d, this goal is now taken off the heuristic goal list for consideration in the following inner loop. Step 14e is uneventful. As directed in step 14f, heuristic transformation g(2), "trigonometric substitution", suggests the substitution $y = \arcsin x$ which transforms the original goal into

$\int \frac{\sin^4 y}{\cos^4 y}\, dy$ which is appended to the goal list and with respect to which an immediate solution is attempted.

In attempting imsln [($\int \frac{\sin^4 y}{\cos^4 y}\, dy$)] steps 13a to 13e are uneventful. As directed in step 13f, this goal is made the first (and now the only) member of the temporary goal list. Since the rest of the goal list is empty, the value of imsln [($\int \frac{\sin^4 y}{\cos^4 y} dy$)] is the same as the value of imsln [( )] which is FALSE.

Accordingly, as further directed in step 14f, since we are in the third case, we go to step 14e. Since no heuristic transformations applicable to the original goal remain, we go to step 14b, which is uneventful. As directed in step 14c, we obtain and associate with the new goal, its character and relative cost estimate. This new goal is then taken off the temporary goal list and made the first (and now the only) member of the heuristic goal list. This goal is then taken off the heuristic goal list and is the goal under consideration in the following inner loop. Step 14e is uneventful. As directed in step 14f, heuristic transformation a(2), "transformation of an elementary function of trigonometric functions", transforms $\int \frac{\sin^4 y}{\cos^4 y}\, dy$ into $\int \tan^4 y\, dy$ with respect to which an immediate solution is tried. This attempt fails, and this goal is made

the first (and now the only) member of the temporary goal list. Hence, as further directed in step 14f, we go to step 14e, which is uneventful; hence we return to step 14f. This time heuristic transformation, a(3) "transformation of an elementary function of trigonometric functions" transforms $\int \frac{\sin^4 y}{\cos^4 y}\, dy$ into $\int \cot^{-4} y\, dy$ which is made the second member of the temporary goal list. We come a third (and final) time through the inner loop back to step 14f. This time, heuristic transformation "j", "rational function of sines and cosines", suggests the substitution, $z = \tan \frac{y}{2}$ which transforms $\int \frac{\sin^4 y}{\cos^4 y}\, dy$ into $\int 32 \frac{z^4}{(1 + z^2)(1 - z^2)^4}\, dz$ with respect to which an immediate solution is attempted.

In imsln $[(\int 32 \frac{z^4}{(1 + z^2)(1 - z^2)^4}\, dz)]$, steps 13a to 13d are uneventful. However, in step 13e, algorithm-like transformation "a", "factor constant", factors out the constant, 32, and generates a new goal $\int \frac{z^4}{(1 + z^2)(1 - z^2)^4}\, dz$ which is added to the goal list and with respect to which an immediate solution is attempted. Steps 13a to 13e are uneventful. As directed in step 13f, this newest goal is added to the temporary goal list. Since the rest of the goal list is empty, the value of imsln $[(\int 32 \frac{z^4}{(1 + z^2)(1 - z^2)^4}\, dz)]$ is FALSE.

As further directed in step 14f, we go to step 14e. Since no heuristic transformations applicable to the goal, $\int \frac{\sin^4 y}{\cos^4 y}\, dy$ remain, we go to step 14b, which is uneventful. As directed in step 14c, we obtain and associate with each of the three goals on the temporary goal list, its character and relative cost estimate. The three goals are taken off the temporary goal list, and each is inserted in the heuristic goal list according to its relative cost estimate. Since the relative cost estimates are 2, 2 and 5 respectively, it turns out that the heuristic goal list is the same as the temporary goal list had been. As directed in step 14d, we take the goal, $\int \tan^4 y\, dy$ off the heuristic goal list and consider it in the following inner loop. As directed in step 14f, heuristic transformation b(3) "substitution for a trigonometric function", suggests the substitution $z = \tan y$ which transforms $\int \tan^4 y\, dy$ into $\int \frac{z^4}{1 + z^2}\, dz$ for which an immediate solution is attempted.

In imsln [$(\int \frac{z^4}{1 + z^2}\, dz)$], algorithm-like transformation "g", "divide polynomials", generates the new goal, $\int (-1 + z^2 + \frac{1}{1 + z^2})\, dz$ which, in turn, is transformed by algorithm-like transformation "c", "decompose", into three goals, namely, $\int -dz$, $\int z^2 dz$, $\int \frac{dz}{1 + z^2}$.

The first two of these are of standard form and are integrated. Pruning is unsuccessful. The third is made the first (and now the only) member of the temporary goal list. The attempt for an immediate solution has failed.

Hence, as further directed in step 14f, we go to step 14e, and since no more heuristic transformations applicable to $\int \tan^4 y\, dy$ remain, we go to step 14b, which is uneventful. As directed in step 14c, we obtain and associate with $\int \frac{dz}{1 + z^2}$ its character and relative cost estimate, namely, 3. This goal is taken off the temporary goal list and inserted between the two members of the heuristic goal list, since their relative cost estimates are 2 and 5 respectively. As directed in step 14d, we take the goal, $\int \cot^{-4} y\, dy$ off the heuristic goal list for consideration in the following inner loop. In step 14f, heuristic transformation b(4), "substitution for a trigonometric function" suggests the substitution z = cot y which yields $\int - \frac{dz}{z^4(1 + z^2)}$ for which an immediate solution is attempted. In this unsuccessful attempt is generated a new goal, $\int \frac{dz}{z^4(1 + z^2)}$ which is made the first member of the temporary goal list. Now as directed in step 14e, since no heuristic transformations applicable to $\int \cot^{-4} y\, dy$ remain, we go to step 14b, which is uneventful. As directed in step 14c, we obtain and associate

with $\int \frac{1}{z^4(1 + z^2)}\, dz$ its character and relative cost estimate, namely, 4. We take this goal off the temporary goal list and insert it between the two members of the heuristic goal list since their relative cost estimates are 3 and 5 respectively. As directed in step 14d, we take $\int \frac{dz}{1 + z^2}$ off the heuristic goal list for consideration in the following inner loop. Step 14e is uneventful. As directed in step 14f, heuristic transformation g(1), "trigonometric substitution" suggests the substitution w = arctan z which transforms the goal into $\int dw$ for which an immediate solution is attempted. In imsln $[(\int dw)]$, since the goal is of standard form, it is achieved, i.e., $\int dw = w$. Pruning with respect to this newly achieved goal successively achieves goals until the original goal is achieved, namely, $\int \frac{x^4}{(1 - x^2)^{5/2}}\, dx = \arcsin x + \frac{1}{3} \tan^3 \arcsin x - \tan \arcsin x$.

B. Procedure for Learning a Standard Form

There is a procedure for adding, to the standard forms list, a standard form, i.e., the form of an integral and the form of its corresponding solution. This procedure can be used directly, but it is more important when it is used in a procedure for learning a standard form. This latter procedure integrates an expression (which may be a form) as in Section A and, if the integral is found, the former procedure is used to add the result

to the standard forms list. In particular, the form may be an ordinary elementary expression, in which case the procedure has merely "memorized" the answer to a particular problem. Section V F has an example of a learned standard form.

C. Definite Integration Procedure

SAINT can perform some definite integrations by first finding the corresponding indefinite integrals. Thus, for example, for the problem, $\int_0^3 x\sqrt{x^2+16}\,dx$ (1 Jan. '60, 13) SAINT first finds the indefinite integral,

$$\int x\sqrt{x^2+16}\,dx = \frac{1}{3}(x^2+16)^{3/2}$$

SAINT substitutes the limits and obtains the answer $\frac{61}{3}$.

D. Multiple Integration Procedure

SAINT can perform multiple integration when it can perform the required definite integrations, e.g., $\int_{-1}^{1}\int_{y^2}^{2-y^2} dx\,dy$ (1 Jan. '55, 6).

E. Realization of SAINT on a Computer

For the most part, the implementation of the integration procedure described above is straightforward though very lengthy. The programming language used is LISP, as implemented on the I.B.M. 709 and 7090 digital computers. About a third of the 32,768 register memory of the computer is occupied by the LISP

system, which includes many general purpose programs written by others (Chapter II). Another third is occupied by prerequisite programs, such as, those discussed in Chapter III. The remaining third is occupied by the SAINT program. Since the program is so large, only about three thousand registers are available for working space in the free storage list, despite great effort to make this list as large as possible. The overall procedure is embodied in a LISP function with three arguments, integral [integrand; variable; resource allotment] where the second argument denotes the variable with respect to which the integration is to be performed. The value of the function is either the solution to the integral or NIL, which denotes failure. Each goal is represented by an object in the sense of LISP. When a new goal is created, a unique print name such as, G0002, is assigned. In addition to its print name, the association list of a goal contains or may contain:

1. ALIVE

If ALIVE occurs on the association list of a goal, that goal is alive; otherwise it is dead.

2. Two consecutive elements, INTEGRAND and the integrand.

3. Two consecutive elements, CHILDREN and the list of children. Some goals are childless, in which case these two elements do not appear.

4. Two consecutive elements, PARENTS and a list of pairs;

the first member of each pair is the name of a parent. The second member of the pair describes how the solution to the parent problem is related to the solution of the problem, and is either:

a. COMPONENT, which denotes that the parent integrand is a sum which was decomposed.

b. An expression which represents a function which, when applied to the solution of this goal, will have for its value the solution of the parent goal.

The original problem has no parents. The parent and children lists fully specify the goal tree, (and are the latter's only representation); operations such as pruning on the goal tree are performed by operating on these lists.

5. An element may be present to indicate that this goal was generated by heuristic transformation 11a, "elementary function of trigonometric functions".

6. Four elements, CHARACTER, the character, RELATIVECOSTESTIMATE and the relative cost estimate. These four elements are associated with the goal only if it is put on the heuristic goal list.

7. Two elements INTEGRAL and the solution to the problem. These elements are associated with the goal only if it has been achieved.

As an illustration, here is the final segment of an association list of a typical goal:

(ALIVE, INTEGRAND, (POWER, (PLUS, 1.0, (POWER, X, 2.0)), -1.0), CHILDREN, (G0008), PARENTS, ((G0004, COMPONENT)), CHARACTER, (3, 3, TRUE, TRUE, NIL, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE), RELATIVECOSTESTIMATE, 3).

Chapter V

Experiments and Findings with SAINT

This chapter describes some of SAINT's typical observed behavior and how some modifications change its behavior. The experiments to measure SAINT's behavior involve 86 problems. Largely for the purposes of debugging, 32 of the problems were selected or constructed by the author, who fully expected SAINT to solve them all. More objectively, the remaining 54 problems were selected from M.I.T. freshman calculus final examinations by my assistant with instructions to select the more diverse and difficult problems provided only that the method of partial fractions was not needed for the solution. More specific information about how the problems were selected for each experiment is discussed with that experiment. The description of each of the six experiments is immediately followed by some observations concerning behavior. The various versions of SAINT discussed in Sections A, B, ..., F in this chapter are designated SAINT, BSAINT, ..., FSAINT respectively. The measures of behavior that we use are:

1. Power

The power of a version of SAINT refers to the size of the class of problems that it can solve.

2. Time and the number of constructs

All the times mentioned in this chapter refer to the I.B.M. 7090 computer at M.I.T.'s Lincoln Laboratory. However, some of the experiments were performed on the I.B.M. 709 at the M.I.T. Computation Center. To obtain the approximately equivalent 709 time, multiply the 7090 time by 5.15, which was determined by running the same problem on both machines. We denote 22,300 constructs by 22.3 kcons. For a few problems, their unrecorded time or number of constructs was estimated from the observed fact that one minute of computation represented approximately 22.3 kcons.

3. Number of subgoals and unused subgoals

The original goal is not included in the number of subgoals. An unused subgoal is a subgoal which is not needed in the solution chain.

4. Level and superfluous level of a solution

The level of a solution is the maximum level at which a used subgoal occurs in the goal tree during that solution. A solution has a superfluous level of s if and only if the maximum unused subgoal level s exceeds the level of the solution. Thus, all other things being equal, a version of SAINT is best if it generates the fewest solutions with a superfluous level.

5. Heuristic level and superfluous heuristic level of a solution

These two measures are similar to those in the preceding paragraph except that only the goal tree branches representing

heuristic transformations are considered rather than all the branches representing algorithm-like or heuristic transformations.

Chapter VI contains some conclusions based on these six experiments.

A. Unmodified SAINT

The SAINT program described in the preceding chapter tried to solve all 86 problems selected by the author and his assistant. In this attempt, the computer spent about half of its time in reclaiming abandoned memory registers for reuse. Approximately half of the remaining time was spent in pattern recognition, namely, in finding characters and in recognizing when an integrand is of standard form or amenable to an algorithm-like or heuristic transformation. As the author expected, SAINT solved all 32 of his problems. Of the 54 M.I.T. problems, SAINT solved 52 (96%) and quickly (in less than a minute) reported failure for the other two problems, namely, $\int x\sqrt{1 + x}\ dx$ (2 Aug. '56, 2) and $\int \cos\sqrt{x}\ dx$ (2 June '59, 2b). If SAINT were extended as suggested in Chapter VII, it could solve both of these problems. Both of the failures are excluded from the averages in table 5.1, which summarizes SAINT's average performance.

TABLE 5.1

SAINT's Average Performance

	minutes	kcons	sub-goals	unused sub-goals	level	super-fluous level	heu-ristic level	super-fluous heuristic level
32 Author Problems	3.3	77	6.4	2.0	3.5	0.3	1.0	0.01
52 M.I.T. Problems	2.0	44	4.7	0.8	2.9		0.8	
All 84 Problems	2.4	56	5.3	1.25	3.0	0.01	0.9	0.004

In this paragraph we complement the tabulation of SAINT's average performance on M.I.T. problems with some examples of SAINT's extreme behavior. For this purpose, only M.I.T. problems are considered since they were selected more objectively. SAINT seemed to find $\int_1^2 \frac{dx}{x}$ (1 Jan. '61, 22) the easiest problem since it generated no subgoals at all and took the least time and fewest constructs, namely, 0.03 minutes and 0.6 kcons. SAINT took the most time and constructs (18 minutes and 370 kcons) for $\int \frac{\sec^2 t}{1 + \sec^2 t - 3 \tan t}\, dt$ (2 June '59, 1b), whose solution ties for the maximum heuristic level

of four. The other problem whose SAINT solution has a heuristic level of four is $\int \frac{x^4}{(1 - x^2)^{5/2}}\,dx$ (2 May '57, 3c). The maximum heuristic level obtained by the unmodified LOGIC THEORIST is two, which occurred for two of the 38 solutions. SAINT generated the most subgoals (18) and had the maximum level (8) for $\int (\sin x + \cos x)^2 dx$ (1 June '60, 6b). The maximum ratio of the number of unused subgoals to the number of subgoals occurs in two problems in which two of the three subgoals are unused; the two problems are $\int_0^{\frac{\pi}{3}} \tan x \sec^2 x\,dx$ (2 June '60, 3) and $\int_0^{\frac{\pi}{4}} \sin x \cos x\,dx$ (1 Jan. '60, 14). In 37 of the 52 problems, SAINT generated only subgoals that were needed in the solution chain. In this regard, SAINT registered its best performance on one of these 37 problems, $\int_{5/2}^{4} \int_{x(11 - 2x)}^{4(x - 1)(5 - x)} dy\,dx$ (1 June '60, 7), for which SAINT generated 16 subgoals, all of which were needed in the solution chain.

B. BSAINT, i.e., SAINT When It Had a Faulty Heuristic Transformation

Instead of algorithm-like transformation IV A 4d, "linear substitution", and heuristic transformation IV A 11c, "substitution for a subexpression whose derivative divides the integrand", BSAINT had heuristic transformation c' which is the same as

heuristic transformation "c" except that the substitution for s (v) was made so long as the number of factors in $\frac{g(v)}{s'(v)}$ did not exceed the number of factors of g(v). The problems in table 5.2 were selected because they were the only problems that showed promise of causing a significant quantitative difference in behavior between SAINT and BSAINT. For the second and third problems in the table, heuristic transformation c' suggested fruitless substitutions which seriously damaged BSAINT's performance. Hence it was replaced by heuristic transformation "c", which among other things excludes linear substitutions. Algorithm-like transformation "d" was added in order to catch appropriate linear substitutions. The difference in behavior for the first and fourth problems in the table arises from the fact that an appropriate linear substitution is made by a transformation which is algorithm-like rather than heuristic. BSAINT is slightly more powerful than SAINT, e.g., it can solve $\int x\sqrt{1 + x}\ dx$ (2 Aug. '56, 2) for which heuristic transformation c' will appropriately suggest the substitution $y = \sqrt{1 + x}$. However, chapter VII shows how SAINT can be extended to solve this problem and hence there is no need to accept the disadvantages of BSAINT. Table 5.2 shows how SAINT dominates BSAINT in all other measures of performance.

TABLE 5.2

Performance of BSAINT, i.e., SAINT When It Had a Faulty Heuristic Transformation

Problem	Minutes BS	Minutes S	Kcons BS	Kcons S	Subgoals BS	Subgoals S	Unused Subgoals BS	Unused Subgoals S	Level BS	Level S	Superfluous Level BS	Superfluous Level S	Heuristic Level BS	Heuristic Level S
$\int \frac{dx}{\sec^2 x}$	7.4	2.1	165	45	12	7	5	0	6	6			2	1
$\int \frac{x^2+1}{\sqrt{x}}\,dx$	4.5	1.7	95	32	10	3	7	0	2	2	3		1	1
$\int \frac{x\,dx}{\sqrt{x^2+2x+5}}$	19.4	16	430	314	18	14	10	6	5	5			3	3
$\int \frac{e^x dx}{1+e^x}$	2.9	1.7	66	36	8	2	1	0	5	2			1	1

C. CSAINT, i.e., SAINT When It Computed Unnecessary Characters

Instead of adding a new goal to the temporary goal list, CSAINT immediately computed the goal's character and inserted the goal in its place on the heuristic goal list. For a few problems such as those selected for table 5.3, some subgoal characters were never needed so that the time and space used in computing these characters was wasted. SAINT avoids this waste.

The qualitative differences in behavior are easily anticipated. For the two versions, the time and number of constructs are not significantly changed except in some problems such as those in table 5.3 in which CSAINT uses significantly more time and constructs than SAINT does. No differences can occur for the other measures of behavior.

TABLE 5.3

Performance of CSAINT, i.e., SAINT When It Computed Unnecessary Characters

Problem	Minutes CS	Minutes S	Kcons CS	Kcons S
$\int \sin^2 x \cos x \, dx$	4.4	2.0	90	58
$\int (\sin^2 x + 1)^2 \cos x \, dx$	6.5	3.8	152	88
$\int \frac{e^{2x} dx}{1 + e^x}$	5.6	3.7	129	84
$\int \frac{dx}{1 - \cos x}$	5.5	2.0	127	73
$\int_0^{\frac{\pi}{3}} \tan x \sec^2 x \, dx$ (2, June '60, 3)	4.3	2.6	97	62
$\int_0^1 x(\ln x) \, dx$ (2, June '60, 11)	2.7	2.2	62	46
$\int_0^{\frac{\pi}{6}} \sin x \cos x \, dx$ (1, Jan. '60, 14)	3.3	2.6	89	58

D. DSAINT, i.e., SAINT When It Had a Faulty Heuristic Transformation and Computed Unnecessary Characters

DSAINT combined the defects of BSAINT and CSAINT, i.e., contained the faulty heuristic mentioned in section B and computed unnecessary characters as the version in Section C. The problem in table 5.4 was selected so that both defects would damage DSAINT's performance.

Table 5.3

Performance of DSAINT, i.e., SAINT When It Had a Faulty Heuristic Transformation and Computed Unnecessary Characters

Problem	Minutes DS	Minutes S	Kcons DS	Kcons S	Sub-goals DS	Sub-goals S	Unused Sub-goals DS	Unused Sub-goals S	Level DS	Level S	Super-fluous Level DS	Super-fluous Level S	Heu-ristic Level DS	Heu-ristic Level S	Superfluous Heuristic Level DS	Superfluous Heuristic Level S
$\int \frac{x+1}{\sqrt{2x - x^2}}\, dx$	17.5	9.6	317	214	19	13	15	9	3	3	4	4	2	2	3	3

E. ESAINT, i.e., SAINT Trying Heuristic Goals in Order of Generation

Instead of trying heuristic goals in order of increasing depth, ESAINT tries heuristic goals merely in the order in which they were generated. A few problems such as those selected for table 5.5 caused a difference in behavior between SAINT and ESAINT. Table 5.5 shows how SAINT was superior in the last three problems, but, curiously, inferior in the first problem. The measures of behavior critical in this experiment are discussed below.

1. Power

Given sufficient time and space, ESAINT is at least as powerful as SAINT since ESAINT will eventually try every transformation that SAINT does. In fact, I think that, given sufficient time and space, ESAINT is more powerful than SAINT. However, as will appear below, this added power, if any, would be purchased at a high price. Within more practical limits of time and space, SAINT appears to be more powerful than ESAINT, e.g., SAINT succeeded with the fourth problem in table 5.5 whereas ESAINT failed for lack of space (out of push-down list).

2. Time and number of constructs

Given enough time and space, ESAINT would have solved the fourth problem in table 5.5 but it would have used considerably

more time and constructs than SAINT did for the same problem. Thus it is fair to say that SAINT outperforms ESAINT in three of the four problems as measured by time and number of constructs.

3. Unused subgoals

In the second problem, ESAINT generated exactly the same subgoals as SAINT but seriously degraded its performance by an unproductive attempt to apply its heuristic transformations to the unused subgoal. In the other three problems, the difference in performance between SAINT and ESAINT can be traced directly to the difference in the number of unused subgoals.

4. Superfluous heuristic level

Of the author's 32 problems, the first problem in table 5.5 is the only one to have a superfluous heuristic level; none of the 54 M.I.T. problems had a superfluous heuristic level. An ESAINT solution cannot have a superfluous heuristic level. SAINT was inferior to ESAINT for this problem due to the fact that SAINT's solution was exceptional in that it had a superfluous heuristic level.

TABLE 5.5

Performance of ESAINT, i.e., SAINT Trying Heuristic Goals in Order of Generation

Problem	Minutes ES	Minutes S	Kcons ES	Kcons S	Sub-goals ES	Sub-goals S	Unused Sub-goals ES	Unused Sub-goals S	Level ES	Level S	Super-fluous Level ES	Super-fluous Level S	Heu-ristic Level ES	Heu-ristic Level S	Remarks
$\int \frac{x+1}{\sqrt{2x-x^2}}\,dx$	8.2	9.6	169	214	11	13	7	9	3	3	4	4	2	2	SAINT had Superflous Heuristic Level: 3
$\int \frac{2e^x}{2+3e^{2x}}\,dx$ (2 March '60, 4)	6.7	6.0	165	146	4	4	1	1	3	3			2	2	
$\int \frac{x^4 dx}{(1-x^2)^{5/2}}$ (2 May '57, 3c)	17	11	343	307	15	13	7	5	6	6			4	4	
$\int \frac{\sec^2 t\,dt}{1+\sec^2 t-3\tan t}$ (2 June '59, 1b)	17.1	18	352	370	6	9	5	4	3	5			2	4	ESAINT died in effort

F. FSAINT, i.e., SAINT With a Learned Standard Form

Using the procedure described in Section IV B, SAINT solved $\int \frac{dv_1}{p_1 + p_2 v_1^2}$ and added to the standard forms list the answer it found, namely, $\frac{1}{\sqrt{p_1 p_2}} \arctan\left(\sqrt{\frac{p_2}{p_1}}\, v_1\right)$ FSAINT is SAINT with this learned standard form. The presence of the standard form did not significantly affect performance except when a subgoal is an instance of the learned form as in the problems in table 5.6, in which case performance is significantly improved.

TABLE 5.6

Performance of FSAINT, i.e., SAINT With a Learned Standard Form

Problem	Minutes FS	Minutes S	Kcons FS	Kcons S	Sub-goals FS	Sub-goals S	Unused Sub-goals FS	Unused Sub-goals S	Level FS	Level S	Heu-ristic Level FS	Heu-ristic Level S
$\int \frac{e^{6x}}{e^{4x}+1}\, dx$	5.3	8.5	142	192	9	10	3	3	5	6	1	2
$\int \frac{x^4 dx}{(1-x^2)^{5/2}}$ (2 May '57, 3c)	9.0	11	218	307	10	13	3	5	5	6	3	4
$\int \ln(2+3x^2)dx$ (2 June '59, 2a)	3.5	6.5	99	141	8	10	0	2	6	7	1	2

Chapter VI

Conclusions

Corresponding to the purposes avowed in chapter I, this chapter presents certain conclusions drawn from the SAINT project and, to an extent, from the findings of other workers in the field. Although many of the conclusions are stated flatly and in general terms, no claim is made that the last word has been said concerning intelligent problem solving or, even concerning symbolic integration by machines. However the conclusions are based on experience, namely, the experiments described in the preceding chapter and the author's experience concerning the creation, structure and performance of SAINT. Throughout this chapter, a parenthesized mention of an experiment is an appeal for support of a conclusion to an experiment described in chapter V. The subjects of the conclusions conveniently fall into three categories, namely, intelligence (both natural and artificial), natural intelligence and artificial intelligence.

A. Intelligence (Both Natural and Artificial)

Under intelligence, are listed four subjects (pattern recognition, algorithms and heuristics, problem solving and learning) for which certain conclusions can be made.

1. Pattern recognition

Certain conclusions about pattern recognition can be drawn from SAINT's behavior in recursive matching and character finding.

a. A suitably programmed computer can recognize when a symbolic expression is an instance of a useful pattern, including recursively generated patterns, such as, "rational function of sines and cosines". The idea of character and characteristics, due to Minsky, provides a convenient framework for some successful pattern recognition.

b. In symbolic integration, pattern recognition plays a very important part in three senses.

(1) Pattern recognition consumes much of the program and programming effort.

(2) In SAINT, recursive matching is used frequently and with great variety, e.g., in determinations involving standard forms, algorithm-like and heuristic transformations and relative cost estimates.

(3) Pattern recognition consumes much of the time in problem solving (experiment A). The recursive matching in SAINT is serial, i.e., the recursion is performed sequentially.

c. The time consuming nature of serial pattern recognition points to a parallel (simultaneous) process in three senses.

(1) The fact that humans are so quick to recognize some patterns indicates parallel pattern recognition is present as an important mechanism and skill.

(2) The acquisition of skill in recognizing patterns in a parallel (rather than in a serial) manner is desirable for both men and machines.

(3) Mechanisms for parallel pattern recognition by machines are desirable.

d. Especially until machines can perform effective parallel pattern recognition, great care should be taken not to recognize superfluous patterns (experiment C).

e. A machine can learn to recognize a new pattern for recursive matching which is very useful for suitable problems (experiment F).

2. Algorithms and heuristics

a. A heuristic program can perform well (experiment A) in a domain where there is no totally effective procedure [10].

b. The practical consideration of limited time and space may sometimes justify using a heuristic program rather than another program which may be more powerful if given sufficient time and space (experiment E).

c. While trying to achieve a goal, a heuristic program should use its algorithm-like transformations before it resorts to its heuristic transformations.

d. To avoid many blind alleys, heuristics that suggest transformations should be made as sharp as possible, i.e., a large portion of the suggestions should be good (experiment B).

3. Problem solving

a. Some heuristics are valuable in problem solving (experiment E).

b. The tripartite division of methods into standard forms, algorithm-like transformations and heuristic transformations is very useful in problem solving. Standard forms in SAINT and "substitution" in the LOGIC THEORIST may be instances of an "immediately achieve" procedure which seems to be a basic component of a goal achieving scheme. The input to the procedure is a goal. The output is NIL or one or more of the following three items, namely, TRUE (the goal is achieved), how to achieve the goal or the achievement of the goal. In each domain, the procedure for immediately achieving a goal must be supplied anew and, since it is a very frequently used procedure, should operate very rapidly. The algorithm-like transformations also seem to be a basic component of a goal achieving scheme, but this remains to be seen since they are not present in all schemes, e.g., the LOGIC THEORIST. The organization of SAINT's heuristic transformations (corresponding to that of the LOGIC THEORIST's methods of detachment, forward chaining and backward

chaining) seems to be an often convenient but not a basic component of a goal achieving scheme.

c. A machine can manifest intelligent problem solving behavior, i.e., behavior which, if performed by people, would be called intelligent (experiment A).

d. General problem solving systems and systems that learn effectively from their experience should be developed. Since much of SAINT's goal achieving structure is general or could be learned, these facilities alone would enable a project such as SAINT to be carried out with a small fraction of the effort, and other much more powerful heuristic programs would become easily feasible.

e. Soon, a man machine combination will be solving problems.

f. Soon, intelligent problem solving machines will far surpass man, at first, in real time applications, then as a matter of economics and finally, I think, as a matter of power. The present speed of SAINT compares very favorably with the speed of the average college freshman (experiment A). With a now commercially available large high speed digital computer, such as the I.B.M. 7030 (STRETCH), a compiled but otherwise unimproved SAINT program would run eight hundred times faster, which would far surpass in speed even the most gifted of

mathematicians at this task. At present commercial rates, an I.B.M. 7090 SAINT solution of an average M.I.T. final exam problem costs about fifteen dollars, far more expensive than a human solution. However, a STRETCH SAINT solution would cost only about four dollars or if compiled, only about four cents. This rapidly decreasing cost trend in computers, not to mention possible improvements in the SAINT program will result in solutions which are far cheaper by machine than by man. My conjecture is that, with a greatly increased memory and by sheer speed of calculation and accumulated cleverness, a machine will solve symbolic integration problems beyond the scope of any mathematician. The gist of these remarks applies equally to many domains of intelligent problem solving besides symbolic integration.

4. Learning

A suitably programmed computer can learn by rote or, in fact, can learn a new standard form which dramatically improves its performance for suitable problems (experiment F). SAINT also learns what to do with a goal and uses this learning when it generates a new subgoal which is the same as some previous goal.

B. Natural Intelligence

1. Models

A partial model of intelligent human problem solving in the domain of symbolic integration has been constructed. Symbolic integration problems can be solved by this model, which consists of a computer with the SAINT program which is a hierarchical organization of elementary procedures. The model can be readily modified (experiments B, C, D, E and F).

2. Pedagogy

In addition to the many conclusions from section A relevant to teaching we have the following:

a. The Integral Calculus teacher should teach in detail each of the algorithm-like and heuristic transformations which prove so useful in SAINT (experiment A). First the student should learn the algorithm-like transformations which he then could apply to solve suitable problems. Then the student should learn and use the heuristic transformations. In fact, I think that the SAINT procedure is better than the usual general approaches taught to students for doing symbolic integration. Further research on symbolic integration programs would doubtless reveal still better procedures, e.g., better relative cost estimates and heuristic transformations. Any proposed procedure can be evaluated by experimenting with it.

b. As much as possible, the teacher should teach creativity, i.e., the ability to perform tasks most difficult for a machine to perform, since soon machines will perform the more routine tasks.

c. The teacher should prepare the student to join in a man machine combination for solving problems.

d. Soon, a computer with a program embodying an Integral Calculus procedure will teach this procedure and more.

C. Artificial Intelligence

In addition to many of the conclusions in section A relevant to artificial intelligence, we have conclusions concerning the following:

1. Computer design

a. Computers should be still faster and less expensive.

b. For artificial intelligence applications, computers should have much larger memories. A fourfold increase in SAINT's memory size (now 32,768 registers) could have been readily converted into a hundredfold increase in speed since reclamation time which now accounts for about half of the running time would become insignificant and since a compiled program would run about fifty times faster.

c. Some computers should be designed with symbol manipulating applications in mind, e.g., much computer time and space could be saved if one computer instruction represented

the very frequently used symbol manipulating functions, such as, an f-r chain in LISP.

d. Some computers should be designed with time sharing and man machine combinations in mind.

2. Symbol manipulating language and compiler

Many of the features discussed below either have just been or will be included in LISP.

a. An experimental symbol manipulating program can often suggest improvements in the symbol manipulating language used.

b. A symbol manipulating language, such as, LISP, is a very good start towards a language for representing complex symbol manipulating problems to a computer.

c. A symbol manipulating language should efficiently handle objects that are integers or rational numbers. For symbolic integration, exact rational arithmetic rather than approximate arithmetic is appropriate.

d. A symbol manipulating language should be able to express in a convenient form the manipulation of many kinds of quantities besides list structures, including integer indexed arrays. In SAINT, the availability of integer indexed arrays would have allowed the efficient handling of matrices, convenient for the solution of simultaneous linear equations needed in the method of partial fractions.

e. The computer language should be directly expressible in the symbol manipulating language. Then, frequently used subprograms (inner loops) could be hand coded for increased speed.

f. For artificial intelligence, a symbol manipulating language should include a convenient representation for an executive procedure which can operate a hierarchy of procedures, roughly corresponding to the hierarchy of subgoals. Such an executive procedure can abandon or resume the operation of certain subprocedures, such as attempts to achieve particular subgoals.

g. For this improved symbol manipulating language, there is a need for a more efficient compiler, which, perhaps by heuristics, can do some optimizing in time and space.

3. Heuristic programming

a. The behavior of a heuristic program often suggests improvements in that program. The improvements mentioned in experiments B and C of chapter V were suggested in this way.

b. Some heuristics in a heuristic program can be easily tested experimentally (experiment E).

c. A heuristic program can easily include programs for handling an AND-OR goal tree (such as found in SAINT), which is often useful in complex goal achieving schemes.

Chapter VII

Suggestions for Future Work

Future work will doubtless reveal more and better conclusions about intelligence than those found in the preceding chapter. This, the final chapter, describes some of the forms that this work might take.

A. Direct Extensions to SAINT

An improved computer and symbol manipulating language would facilitate making extensions to SAINT. With his particular purposes in mind, the prospective programmer should select an improved computer and symbol manipulating language and then, revise and extend SAINT to take full advantage of the new machine and language. Below are listed some possible direct extensions to SAINT. A psychological study of human subjects would probably suggest other improvements in SAINT.

1. Hyperbolic functions

SAINT could be extended to handle hyperbolic functions by incorporating methods analogous to those used in handling trigonometric functions and by including a method to convert a hyperbolic function to its equivalent exponential form.

2. Relative cost estimate and subgoal selection

Estimates of relative cost that depend on several characteristics of the goal should be tried. In addition to the depth

of the integrand, these characteristics might include breadth, whether the integrand is algebraic but not rational, transcendental but not a rational function of sines and cosines, the number of occurrences of the variable and the number of noninteger powers of the variable. Following Samuel's successful generalization learning experiments, SAINT could discover the significant characteristics and learn good weights for each. Moreover, SAINT should extend the basis for subgoal selection from merely the relative cost estimate to include the apparent centrality of the subgoal, i.e., SAINT should prefer the subgoal whose achievement would contribute the most to the achieving of the original goal. For further discussion of the selection of subgoals and methods, see section B 4.

3. Solution by transposition

This and the next two sections describe some new methods which might be incorporated in SAINT. SAINT should have the trick of solving a problem by transposition, often preceded by a couple of applications of the method of integration by parts, e.g.,

$$\int e^x \cos x \, dx = e^x \cos x - \int e^x (-\sin x \, dx) = e^x \cos x +$$
$$\int e^x \sin x \, dx = e^x \cos x + e^x \sin x - \int e^x \cos x \, dx$$

Transposing the integral on the right gives

$$2\int e^x \cos x \, dx = e^x \cos x + e^x \sin x \quad \text{hence}$$
$$\int e^x \cos x \, dx = \tfrac{1}{2} e^x (\cos x + \sin x)$$

When SAINT finds that a

newly generated goal is the same as an old one, it could test for a possible solution by transposition.

4. Algorithm-like transformations

SAINT should have more algorithm-like transformations including certain reduction formulas, the Chebyshev integral, the binomial integral and substitution for a constant power of the variable of integration. Once SAINT has exact rational arithmetic, the last three of these four transformations can be conveniently added.

a. Reduction formulas

SAINT should have certain reduction formulas, e.g., especially useful for odd $n \geq 3$ is the reduction formula. $\int \sec^n v \, dv = \frac{1}{n-1} [\sec^{n-2} v \tan v + (n-2) \int \sec^{n-2} v \, dv]$ note that when n is even, heuristic transformation IV A 11 b (3), "substitution for a trigonometric function", suggests the substitution $u = \tan v$ which turns out to be preferable to the reduction formula. Equipped with this reduction formula, SAINT would quickly find the following solution: $\int \sec^3 x \, dx = \frac{1}{2} [\sec x \tan x + \int \sec x \, dx] = \frac{1}{2} [\sec x \tan x + \ln (\sec x + \tan x)]$. This problem can also be done directly in the same way that the reduction formula is derived - by doing an integration by parts (where $\sec^2 x$ is the part to be integrated), by using the secant tangent trigonometric identity and by finishing

with a solution by transposition similar to that described in Section 3. The fact that heuristics to motivate the latter procedure are hard to find probably explains why most people find this problem difficult. Similar remarks apply to cosecant, hyperbolic secant and hyperbolic cosecant. Other reduction formulas found to be useful may also be added as algorithm-like transformations.

b. Chebyshev integral

If the problem is of the form $\int v^{r_1}(c_1 + c_2 v)^{r_2} dv$ where c_1 and c_2 are nonzero constants, r_1 and r_2 are not both integers, r_1 is a rational nonzero constant and r_2 is a rational constant which, if an integer, is positive. Try the substitution in the first applicable case below:

(1) If r_1 is a positive integer, then substitute $u = c_1 + c_2 v$.

(2) If r_2 is a (negative) integer, then substitute $u^d = v$ where d is the denominator of r_1.

(3) If r_1 is a negative integer then substitute $u^d = c_1 + c_2 v$ where d is the denominator of r_2.

(4) If $r_1 + r_2$ is an integer, then substitute $u^d = \frac{c_1 + c_2 v}{v}$ where d is the denominator of r_1 and r_2.

Equipped with this transformation, SAINT would be able to solve a problem in which it previously failed. In

$\int x\sqrt{1 + x}\ dx$ (2 Aug. '56, 2), the first case of this transformation gives the substitution, $y = 1 + x$, which leads to an easy solution.

c. Binomial integral

If the problem is of the form $\int v^p (c_1 + c_2 v^q)^r\ dv$ where c_1 and c_2 are nonzero constants, not all three of p, q and r are integers, p is a constant, q is a constant which is neither zero nor unity and r is a rational constant which, if an integer, is positive, then substitute $u = v^q$. The result of this transformation is often a Chebyshev integral.

d. Substitution for a constant power of the variable of integration

The transformation which will be described in this section will enable SAINT to solve many more problems including the following two, the first of which is one of the two problems on which SAINT failed.

In $\int \cos\sqrt{x}\ dx$ (2 June '59, 2b), SAINT should substitute $y = \sqrt{x}$. In $\int \frac{dx}{x\ (ax^n + c)}$, SAINT should substitute $y = x^n$. The way that SAINT should discover these and other similar good substitutions may be seen from the following consideration. Every integral is of the form $\int v^c$ elf $\{v^{c_1}, \ldots, v^{c_k}\}\ dv$ where $c, c_1, \ldots, c_k$ are constants. For any nonzero constant d,

the substitution $u = v^d$ transforms the above integral into $\int \frac{1}{d} u^{\frac{c+1}{d} - 1}$ elf $\{ u^{\frac{c_1}{d}}, \ldots, u^{\frac{c_k}{d}} \}$ du. From this it is easily seen that d should be chosen so as to make the ratios $\frac{c+1}{d}$, $\frac{c_1}{d}$, ..., $\frac{c_k}{d}$ as simple as possible, which occurs when d is sort of a generalized greatest common divisor. Let C be the set which results from deleting the zero elements from the set $\{c + 1, c_1, \ldots, c_k\}$. Then finding a good d involves finding the common divisor (including the greatest common divisor of the integers) of the numerators in C and in finding the common multiple (including the least common multiple of the integers) of the denominators in C. If the d which is so found is not unity, substitute $u = v^d$.

5. Heuristic transformations

To SAINT's heuristic transformations we have the following list of changes and additions which should improve its performance.

a. Rationalizing the denominator

SAINT should try the method of rationalizing the denominator. Thus, in $\int \frac{dx}{\sqrt{x^2 + a^2} + x}$, SAINT would successfully try multiplying the numerator and denominator by $\sqrt{x^2 + a^2} - x$.

b. Transformation of an elementary function of trigonometric or hyperbolic functions

Heuristic transformation IV A 11a, "transformation of an elementary function of trigonometric functions" should be extended to the hyperbolic case. In addition, some work should be done to reduce in number the many bad suggestions which this transformation makes.

c. Integration by parts

SAINT should give priority to partitions of the integrand which result in differentiating (rather than integrating) expressions that are logarithmic, inverse hyperbolic, inverse trigonometric, or positive integer powers of the variable of integration.

d. Method of partial fractions

SAINT should have the method of partial fractions which requires, among other things, the ability to factor polynomials and to solve simultaneous linear equations with symbolic coefficients. Designing a procedure for factoring polynomials is a fairly interesting project in itself. LISP has just provided for the convenient representation and rapid manipulation of integer indexed arrays of symbolic expressions. This facility provides the convenient framework for the matrix manipulations used in solving simultaneous linear equations with symbolic coefficients.

6. Generalization

Some work should be done to give SAINT the ability to generalize a problem into plausible related forms. By using methods similar to those that solved the problem, SAINT could try to integrate each such form. If this attempt is successful and if the result is useful, SAINT could incorporate such generalizations in its own pattern recognition procedures.

7. Approximate definite integration

SAINT should be given a procedure for performing approximate definite integration with a symbolic input. When its exact symbolic procedures fail on a definite integration problem, SAINT could resort to this approximate method, which, under appropriate circumstances, might be applied to one or more subproblems rather than to the original problem directly.

8. Differential equations

SAINT could be extended to the symbolic solution of some differential equations thus, the present SAINT procedure could be called upon when the problem of solving a differential equation has been reduced to that of solving an integral. More drastic but probably better would be the opposite embedding, i.e., to embed the techniques for solving differential equations into the goal achieving structure of SAINT.

9. Machine as a mathematical assistant

In addition to the ability to solve differential equations, many other mathematical skills may be added to SAINT, e.g., the series expansion in orthogonal functions, such as Fourier series. The machine would gradually accumulate a wide variety of procedures for performing commonly encountered mathematical tasks. Probably a good way to use such a machine is as follows. The machine would be directly connected to many remote consoles. From time to time, the human user at his console types a request for the performance of some elementary mathematical skill, e.g., symbolic integration. The machine types back the required answer.

10. Machine as a mathematician

As time goes on, the procedures which the machine can perform will increase in number and complexity. It seems quite possible that such a machine will eventually equal and then surpass the mathematician at his own game.

B. Some General Areas Requiring Investigation

This section contains a few of the author's thoughts about some of the areas which must be understood before machines can solve really difficult problems. See also in this connection reference [7]. Most of the questions discussed in this section are essentially open.

1. Patterns

More work should be done to enable machines to construct, use and evaluate patterns.

2. Learning

The machine should be able to discover new methods and adjust old ones. It should also be able to learn which methods are appropriate for goals of a certain character.

3. Structure among goals

Work should be done to develop a system to handle a very general type of goal tree independently of any particular problem domain. In addition to the AND and OR relationship, this system should handle many other common relationships among goals, e.g.,

a. MODEL relationship

A goal m is said to be a model of a goal g when the achieving of m will furnish a plan (tentative sequence of methods) which has a better than random chance to achieve g.

b. VEILING relationship

An example of this relationship occurs in integration by parts, e.g., when the goal is $\int x \sec^2 x \tan x \, dx$, then $\int \sec^2 x \tan x \, dx$ is a veiling subgoal for, when it is achieved ($\int \sec^2 x \tan x \, dx = \frac{1}{2} \sec^2 x$), it reveals another subgoal, $\int \frac{1}{2} \sec^2 x \, dx$, of the original goal.

c. SIMULTANEOUS relationship

In this relationship, not only must all the subgoals be achieved (as in the AND relationship), but also all of them must be achieved at the same time.

4. Planning

Machines that can plan well must be developed. A machine must be able to formulate a new plan and to modify an old one which nearly worked. One way that a machine might get a plan is through a model, e.g., a plan for solving a problem may sometimes be constructed from a sequence of methods that solve a model of that problem. In addition, the machine must be able to make a sensibly balanced allocation of its rescurces (time and space) to the various phases of its task, e.g., goal and method selection,application of methods to goals, models and learning. Minsky in [7], discusses goal selection according to centrality and difficulty of the goal by methods which are either global or local and hereditary. A very general approach is to select a goal method pair according to the centrality of the goal and the "promise" of the goal method pair. In appropriate problem domains, the increased generality and flexibility of this scheme may well justify its use despite the increased complexity in the selection process compared to selecting the goal first and then the method. Two methods of selection with their accompanying advantages are:

a. Global or Gestalt methods

A global or Gestalt method of selection is one which, before deciding, considers all the "live" goals in the goal structure. From all the heuristic goals, SAINT selects the one whose integrand is the most shallow. The advantages of the global method are:

(1) The best available selection can be made.

(2) The selection criteria may be arbitrarily elaborate.

(3) It is easy to handle equivalent goals.

(4) Accurate prediction of a goal's heuristic level and powerful termination criteria (such as needed in local methods) are not required.

(5) The global method avoids premature termination of all lines of attack which may occur in a local method.

b. Local, hereditary, recursive or line of attack methods

Before deciding, such a selection method considers only some "live" subgoals (often only one); criteria are given for terminating a line of attack. The advantages of this scheme are:

(1) It is easy to pursue a promising line of attack.

(2) It is economical with time and space.

(3) Such a recursive procedure is easily represented in a language such as LISP.

(4) The relationship between parent and child goals is clear and easy to use.

C. Other Specific Projects in Artificial Intelligence

Projects from the following list should be undertaken. Many of them would be useful in themselves. All of them would shed new light on the general areas requiring investigation discussed in Section B which in turn would help in designing for the projects of this section.

1. The mathematics machine

First the machine would be a mathematical assistant and later a mathematician. Three projects which could be done right now are a trigonometric identity proving machine, a machine for checking proofs in formal systems and a four-dimensional geometry theorem proving machine. A machine would have an advantage over people with diagrams in four or more dimensions.

2. A general-purpose problem solver

As more is learned about how people and special-purpose machines (such as SAINT) solve problems, there will emerge additional general-purpose problem solving techniques including the ability to plan and learn. Many of these techniques should be incorporated in a heuristic program for a computer. After receiving a description of, and possibly some training in, some suitable particular problem domain, such a machine could solve problems in that domain.

3. The communicator

This machine should be able to communicate with other machines and people. To reduce the burden on humans, it should be able to communicate in a language easy for people to use. Thus, the inputs and outputs of the machine should be imperative, declarative, and interrogatory sentences. As in McCarthy's proposed advice-taker system [5] the machine should be able to exhibit "common sense", i.e., the ability to make simple deductions from premises. Probably a good way to use the communicator is as follows. A man would tell the machine the situation and make certain requests for action. Questions would be asked and answered back and forth. When the machine had the information it needed, it would carry out the task.

4. Induction and learning machines

An important thing to work on is the problem of getting machines to make inductions, i.e., to draw general conclusions from a finite amount of special data. The machine should be designed to discover scientific laws from data. A machine for exploring space must be able to make inductions about its environment both for more efficient communication to earth of its discoveries and for adapting itself to a possibly hostile environment. Turing has suggested that it would be easier to construct a "child" machine rather than an "adult" machine;

with an appropriate training sequence, this "child" machine could learn to become an "adult" machine.

5. The teaching machine

The student would sit at one of many consoles directly connected to the machine. The eventual goal is the creation of a machine which would act as a superb tutor, adapting itself to the requirements of each student.

6. The programming machine

Here will occur a progression through compilers which use heuristics until at a certain stage it could fairly be called a programming machine. Such a machine could program in a general-purpose programming language. A human would give the machine more or less informal specifications for a program. The machine and person would communicate until the machine decided that the specifications were consistent and sufficient to specify the program. Then the machine would write the program. Note that the programming machine could learn by incorporating in its own procedures a program which it has written. Note also that just as in the GEOMETRY THEOREM PROVING MACHINE [3] the machine would have available a convenient interpretation of a program, i.e., the execution of the program with legal inputs selected at random.

BIBLIOGRAPHY

1. Church, A. The Calculi of Lambda Conversion. Princeton, New Jersey: Princeton University Press, Second Printing, 1951. (Annals of Mathematics Studies, No. 6.)

2. Dinneen, G. P. "Programming Pattern Recognition." Proceedings of the Western Joint Computer Conference, (1955), pp. 94-100.

3. Gelernter, H., J. R. Hansen, and D. W. Loveland. "Empirical Explorations of the Geometry Theorem Machine." Proceedings of the Western Joint Computer Conference, (1960) pp. 143-147.

4. Hardy, G. H. The Integration of Functions of a Single Variable. Cambridge, England: Cambridge University Press, Second Edition, 1958. (Cambridge Tracts in Mathematics and Mathematical Physics.)

5. McCarthy, J. "Programs With Common Sense." Proceedings of the Symposium on Mechanisation of Thought Processes, Edited by D. V. Blake, and A. M. Uttley. 2 Volumes. London: Her Majesty's Stationery Office. (1959) pp. 75-84.

6. McCarthy, J. "Recursive Functions of Symbolic Expressions." Communications of the Association for Computing Machinery. (Preprints of Conference on Symbol Manipulation Programs, April 1960. Vol. 3) pp. 184-195.

7. Minsky, M. L. "Steps Toward Artificial Intelligence." Proceedings of the Institute of Radio Engineers, (Jan. 1961) pp. 8-30.

8. Minsky, M. L. "A Selected Descriptor-Indexed Bibliography to the Literature on Artificial Intelligence." IRE Transactions on Human Factors in Electronics. (March 1961) pp. 39-55.

9. Newell, A., J. C. Shaw, and H. A. Simon. "Empirical Explorations of the Logic Theory Machine." Proceedings of the Western Joint Computer Conference, (1957) pp. 218-239.

10. Ritt, J. Integration in Finite Terms, Liouville's Theory of Elementary Methods. New York: Columbia University Press, 1948.

11. Samuel, A. L. "Some Studies in Machine Learning Using the Game of Checkers." IBM Journal of Research and Development, Vol. 3 no. 3. (July 1959) pp. 210 - 229.

12. Selfridge, O. G. "Pattern Recognition and Modern Computers." Proceedings of the Western Joint Computer Conference, (1955) pp. 91-93.

13. Shannon, C. E. "Programming a Digital Computer for Playing Chess." Phil. Mag., Vol. 41. (March 1950) pp. 356-375.

14. Turing, A. M. "Computing Machinery and Intelligence." Mind, Vol. 59. (New Series 236, October 1950) pp. 433-460.