July 2015

LAWGICAL An Approach to Computer-Aided Legal Analysis

John T. Welch

LAWGICAL: AN APPROACH TO COMPUTER-AIDED LEGAL ANALYSIS

JOHN T. WELCH*

I. INTRODUCTION

LAWGICAL is a system for computer-based information management designed to aid in legal analysis. The phrase "computer-aided legal analysis" used in the title of this article should not be interpreted here to imply to any degree the takeover of the legal analyst's task. LAWGICAL is intended, rather, as a practical tool of limited scope which enhances, but does not change, existing analysis technique. It is not an application of artificial intelligence. It is an application of computer technology on the same order as legal retrieval services or word processing equipment.

The problem which LAWGICAL addresses is the inherent logical complexity of legal analysis material. LAWGICAL hides and reveals, calculates and simplifies, in order to provide a sequential, visible route through complex data to make it more understandable and useable. An overriding consideration in LAWGICAL's design was that the system not add to the difficulty of the analysis. Consequently the system follows an unobstrusive, undemanding course of action. LAWGICAL does not assume control of the user's exploration, but allows him freedom of choice as it implements his decisions.

The LAWGICAL system incorporates legal codes and derived interpretations into a structure of related definitions and statements, each of which has an associated true or false value. In the form of its data base and some of its processing, the system mimics JUDITH, Popp and Schlink's classic legal analysis program.[1] During a LAWGICAL interactive session, a user selects, from a menu[2] presented at his computer terminal, a structure representing a body of legal analysis material of apparent relevance to his case. The system exposes the structure gradually, working from general to specific statements. The user controls the direction of the exposition and may bypass the exploration of parts of the structure not of immediate interest. As he progresses, the user may enter information matching facts of his case to the structure. This process results in reductions in the user's copy of the structure,[3] reductions which eliminate matched information

*B.S., M.S., Ph.D., North Carolina State University; Associate Professor of Electrical Engineering, University of Akron; Member, ACM, IEEE Computer Society.

[1] Popp & Schlink, *JUDITH, a Computer Program to Advise Lawyers in Reasoning a Case*, 15 JURIMETRICS J. 303 (1975).

[2] A menu is a displayed set of tabulated choices. The term implies that some mechanism is available to select one or more items from the display. Later, we will use the verb "to prompt," meaning to display a request for a response and to receive the response as input.

[3] Reduction is a feature of LAWGICAL, but not of JUDITH.

from the structure, leaving a simplified statement of consequences in terms of facts not yet matched with the users' case. To the user, the reduced structure can represent possible avenues to pursue or facts to gather in applying the chosen body of legal principles to his case.

LAWGICAL is implemented as of this writing as a set of programs and an on-line data base on the University of Akron central computer facility. It is available for interactive use by law faculty and students through the university time-sharing system.[4] The implementation language is APL,[5] a system which lends itself to flexibility in program modification and enhancement. Existence of working programs means that explorations of structures of considerable size can be directly experienced, evaluated, and improved. The LAWGICAL design does not depend on APL features, however, and implementations in other computing environments can be considered, once data base and usage parameters can be estimated.

The reminder of this article will address the LAWGICAL data base and the system's behavior during an interactive session, will consider the nature of logical complexity in legal analysis and the rationale of the assistance provided by LAWGICAL, and will explore finally the features chosen to support the type of assistance provided by LAWGICAL as compared to those of other computer-based systems. This comparison will also afford an opportunity to mention LAWGICAL features which are secondary but essential to the primary goal of managing logical complexity.

In companion articles, Professor Finan[6] considers in general the problem of accurate representation of legal material in the LAWGICAL data base format, and Professor Kovach[7] wrestles with the problem of computer applications in the logically complex area of tax research.

II. THE HIERARCHIAL IMPLICATION STRUCTURE

For purposes of describing the LAWGICAL data base, we use the term *implication* to mean a logical expression and an associated conclusion, written as:

logical expression → conclusion

or as:

IF logical expression THEN conclusion.

The interpretation of an implication is that if the logical expression, which can

[4] VS Personal Computing, available with the IBM Operating System OS/VS.

[5] APL stands for A Programming Language. APL is widely used for interactive applications, developed by Kenneth Iverson. Programs are called functions in APL terminology, but the text will continue to use the more general term.

[6] Finan, *LAWGICAL: Jurisprudential and Logical Considerations*, 15 AKRON L. REV. 675 (1982).

[7] Kovach, *Application of Computer-Assisted Analysis Techniques to Taxation*, 15 AKRON L. REV. 713 (1982).

be evaluated to be true or false, is evaluated to be true, the conclusion is then assigned a true value.

LAWGICAL processes logical expressions made up of premises, logical operators and parentheses. Premises, like conclusions, are statements which can be assigned true or false values. A sample implication would be:

$$(B \vee C) \wedge (D \vee E) \wedge \sim (F \vee G) \rightarrow A$$

Premises and conclusions are represented here by one-letter names. The symbols $\vee$, $\wedge$, and $\sim$ stand for logical connectives OR, AND, and NOT,[8] respectively. Parentheses control the order of operations in the evaluation process, in that values are assigned to parenthesized sub-expressions first.

(B V C) $\wedge$ (D V E) $\wedge$ $\sim$ (F V G) $\rightarrow$ A
WHERE
B: THE INSTRUMENT PAID IS A NON-NEGOTIABLE INSTRUMENT COVERED BY UCC ARTICLE 3 (SECTION 3-305)
C: THE INSTRUMENT PAID IS A NEGOTIABLE INSTRUMENT WITHIN THE MEANING OF UCC 3-104 (SEE ANALYSIS F2)
D: THE PERSON PAID HOLDS IN DUE COURSE (SEE ANALYSIS H1)
E: THE PERSON WHO OBTAINED PAYMENT HAS IN GOOD FAITH CHANGED HIS POSITION IN RELIANCE ON THE PAYMENT
F: THE PAYOR BANK MAY REVOKE THE SETTLEMENT FOR THE INSTRUMENT GIVEN TO THE PERSON PAID AND/OR RECOVER ANY PAYMENT MADE UNDER UCC 4-301 (SEE ANALYSIS F3)
G: THE PERSON WHO OBTAINED PAYMENT HAS BREACHED A WARRANTY UNDER UCC 3-417(1)
CONTINUE WITH EX, TF, OR RE.

Figure 1. An Initial Display Frame[9]

The system displays an implication along with text explaining each component premise, illustrated as in Figure 1. We shall call each such display a frame. One frame shows the immediate strategies available for establishing the conclusion, by assigning true or false values to premises of the expression. In the example shown in Figure 1, it is evident that if B is known to be true, C does not have to be established to reach the

---

[8] Representing true by the numeral 1 and false by the numeral 0, we can summarize the effect of logical connectors as:

$0 \vee 0 \rightarrow 0$	$0 \wedge 0 \rightarrow 0$	$\sim 0 \rightarrow 1$
$0 \vee 1 \rightarrow 1$	$0 \wedge 1 \rightarrow 0$	$\sim 1 \rightarrow 0$
$1 \vee 0 \rightarrow 1$	$1 \wedge 0 \rightarrow 0$	
$1 \vee 1 \rightarrow 1$	$1 \wedge 1 \rightarrow 1$	

The OR (V) connective is also termed an inclusive OR. The operator call the Exclusive OR (V) is not provided directly, but can be expressed as $A \vee B = (A \wedge (\sim B) \vee ((\sim A) \wedge B)$.

[9] This example is based on an analysis in preparation by Finan, *supra* note 6, (modified by this writer for purposes of illustration) for "Payment made by a payor bank is final within the meaning of UCC 3-418."

conclusion. Evident too is the dependence of the conclusion on a logical chain of four links:

(B V C), (D V E), ~ F and ~ G.

A user who is uncertain about true or false values would probably seek information first about the link that appears to him to be the weakest.

LAWGICAL accepts true or false value assignments and carries out expression evaluation. If user assignments indicate that an expression is true, then the conclusion is assigned a true value. If the expression evaluates as false, no assignment can be made to the conclusion. This system displays a message to indicate that the implication has failed. To establish that a conclusion is false, and therefore to cause the assignment of a false value, the data base must have available a separate implication for the complement (falseness) of the conclusion. The LAWGICAL implication structure provides for complement conclusions.

User assignments to premises result in a reduction of an implication to a simpler form, when they do not imply a value of the logical expression. For example, the assignments

B ← true, D ← false, and F ← false

reduce the sample implication of Figure 1 to

$E \wedge \sim G \rightarrow A.$

The reduction applies only to the user's copy of the implication, and does not affect the system data base, which is protected from any user alteration. The display frame of the reduced implication omits the text for the premises which have been removed. To start a session, a user selects a set of implications by name, from a menu frame giving a description of many such sets. The implications of a set form a linked structure, in that all conclusions but one are premises of other implications in the set. Such sets of implications will be called "implication structures," illustrated by the example set forth in Figure 2. The conclusion which is not a premise in any other implication represents a goal conclusion for all assignments, and is the first conclusion displayed. The text explaining the goal conclusion appears in the structure menu.

Implication structures are of indefinite size and they routinely exceed the amount of memory made immediately available to an interactive user. In the APL implementation, this immediately available memory is the user's personally assigned workspace. Figure 2 represents a part of the associated implication structure that is loaded into the user's workspace along with the implication of Figure 1. LAWGICAL displays refer to such a portion of an implication structure as an "analysis," as illustrated in Figure 1.

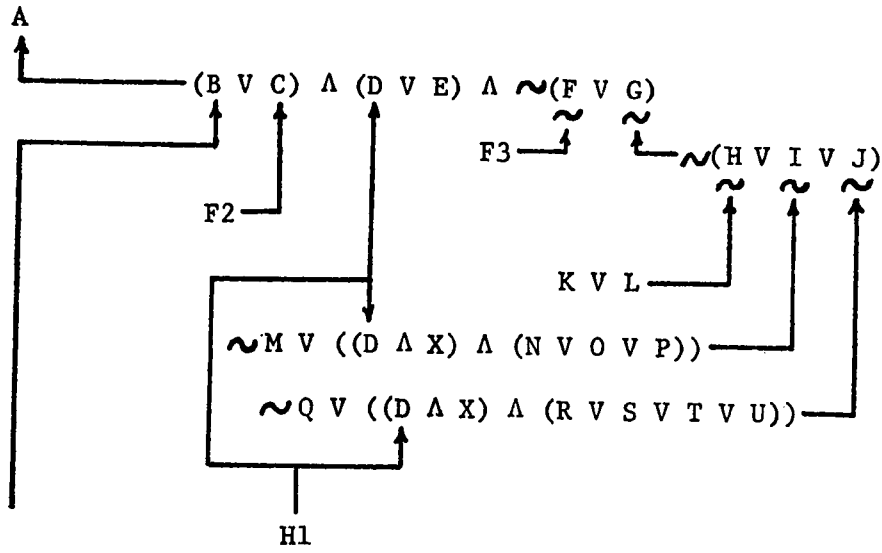

Figure 2. Part of an Implication Structure.

The arrows in Figure 2 represent implication. LAWGICAL displays represent complement conclusions by showing the NOT logical operator at the head of the arrow, as in:

$$\sim (H \vee I \vee J) \rightarrow \sim G$$

The same convention is used in Figure 2. There is nothing to restrict the number of times a premise may be used in an implication or in the implication structure. A true or false value assignment to a premise takes effect at every occurrence of the premise in the structure. It is also possible to have both a premise and its complement in the same implication, as in:

$$(X \wedge P) \vee (\sim X \wedge Q) \rightarrow Y$$

This implication might express different circumstances which exist, depending on X. Or it may express contingencies to be covered, such as:

If the court chooses to make interpretation X THEN P; ELSE Q.[10]

As we have described, the user may assign values to reduce the implication frame. If instead, he indicates that he wishes to expose a more detailed level of the structure, LAWGICAL responds with a list of frame premises which are conclusions of available implications. The user chooses one, which is then displayed as the current frame. This process, which will be called expansion, provides the user with access to the whole structure.

At a given point in a LAWGICAL session, a user by his expansion decisions, has defined a path of incompletely reduced implications from the goal implication to the current frame. LAWGICAL provides for the user's return along this path to higher levels, and permits selection of alternate expansion paths from any level by choice of alternate premises for

[10] *See* Finan, *supra* note 6, at 688-90, for a discussion of the "else" clause.

expansion. The user's return along a path is initiated by an implication assignment or failure (false expression value), or by a user's direct request for a return. Implication and premise assignments are carried back along a path, and are automatically applied to further reduce previously encountered implications, perhaps triggering further automatic assignments, failures and reductions. Previous assignments are also carried forward along new expansion paths, allowing the user to see the consequences of the known facts of his case, without requiring that the information be reentered.

LAWGICAL asks no questions about the user's case. The user types TF when he wants to assign true or false values to premises, and LAWGICAL asks for the list of names to be assigned true values, then the list to be assigned false values. When he wants to expand on the implication of the current frame, the user types EX. LAWGICAL replies with a menu of conclusions and complement conclusions left among premises in the implication of the current frame. If no expansions are available, LAWGICAL says so; if only one expansion is left, LAWGICAL chooses it automatically. If the user wishes to return without assignment or expansion, he types RE. The responses to these user inputs are almost immediate. After each response the user may save the reduced structure and sign off, to resume the exploration at a later time, at the point he left it.

Figure 3 illustrates many of the behaviors just described, with an example based on the structure of Figure 2. The session is assumed to be a continuation, immediately following Figure 1. In Figure 3, user responses are in parenthesis and text has been removed.

(TF)
TYPE THE NAMES IN (B V C) $\wedge$ (D V E) $\wedge$ $\sim$ (F V G)
FOR PREMISES KNOWN TO BE TRUE: (D)
FOR PREMISES KNOWN TO BE FALSE: ( )
(B V C) $\wedge$ (D V E) $\wedge$ $\sim$ (F V G)
IS REDUCED TO:
(B V C) $\wedge$ $\sim$ (F V G) $\rightarrow$ A
WHERE:
B: THE INSTRUMENT PAID IS A NON-NEGOTIABLE INSTRUMENT COVERED BY UCC ARTICLE 3 (SECTION 3-305)
C: THE INSTRUMENT PAID IS A NEGOTIABLE INSTRUMENT WITHIN THE MEANING OF UCC 3-104 (SEE ANALYSIS F2)
F: THE PAYOR BANK MAY REVOKE THE SETTLEMENT FOR THE INSTRUMENT GIVEN TO THE PERSON PAID AND/OR RECOVER ANY PAYMENT MADE UNDER UCC 4-301 (SEE ANALYSIS F3)
G: THE PERSON WHO OBTAINED PAYMENT HAS BREACHED A WARRANTY UNDER UCC 3-417(1)

CONTINUE WITH EX, TF, OR RE.

(EX)
SELECT A PREMISE BY TYPING ONE OF THE FOLLOWING:
B $\sim$ G
($\sim$ G)
$\sim$ (H V I V J) $\rightarrow$ $\sim$ G

WHERE
. . .
CONTINUE WITH EX, TF, OR RE.

(EX)
SELECT A PREMISE BY TYPING ONE OF THE FOLLOWING:
~ H ~ I ~ J
(~ J)
~ Q V ((D ∧ X) ∧ (R V S V T V U))
IS REDUCED TO:
~ Q V ((X) ∧ (R V S V T V U)) → ~ J
WHERE
. . .
CONTINUE WITH EX, TF, OR RE.

(TF)
TYPE THE NAMES IN ~ Q V ((X) ∧ (R V S V T V U))
FOR THE PREMISES KNOWN TO BE TRUE: (X)
FOR THE PREMISES KNOWN TO BE FALSE: (Q)
J IS REJECTED
RETURNING TO G
~ (H V I V J)
IS REDUCED TO:
~ (H V I) → ~ G
WHERE
. . .
CONTINUE WITH EX, TF, OR RE.

(EX)
SELECT A PREMISE BY TYPING ONE OF THE FOLLOWING:
~ H ~ I
(~ I)
~ M V ((D ∧ X) ∧ (N V O V P))
IS REDUCED TO:
~ M V ((N V O V P)) → ~ I
WHERE
. . .
CONTINUE WITH EX, TF, OR RE.

(TF)
TYPE THE NAMES IN ~ M V (N V O V P))
FOR PREMISES KNOWN TO BE TRUE: ( )
FOR PREMISES KNOWN TO BE FALSE: (M)
I IS REJECTED
RETURNING TO G
~ (H V I)
IS REDUCED TO:
~ (H) → ~ G
WHERE
CONTINUE WITH EX, TF, OR RE.

(EX)
K V L → ~ H
WHERE
. . .
CONTINUE WITH EX, TF, OR RE.

(EX)
NO EXPANSIONS ARE AVAILABLE. CONTINUE BY TYPING: TF
OR BY TYPING: RE

```
      (TF)
TYPE THE NAMES IN K V L
FOR PREMISES KNOWN TO BE TRUE:     (L)
FOR PREMISES KNOWN TO BE FALSE:    (K)
H IS REJECTED
RETURNING TO G
G IS REJECTED
RETURNING TO A
(B V C) ∧ ~ (F V G)
IS REDUCED TO:
(B V C) ∧ ~ (F) → A
WHERE
    . . .
```

Figure 3. A Sample LAWGICAL Session.

III. COMPLEXITY IN LEGAL ANALYSIS

We use the term "complex" in a restricted sense, to mean "having more than an easily manageable number of related parts." This definition leaves aside other reasons why a legal text can be difficult to understand, such as ambiguity or convoluted structure. However, this special use of the term is not unconventional. Computer science uses the term "complexity" to mean numbers of operations, or numbers of memory cells, required to solve a problem.

Complexity in the above-mentioned sense is inherent in legal analysis material. Acceptably accurate rules require finely drawn categories and specific, multi-faceted definitions. Codes economically organized around a consistent set of principles become cluttered by exceptions and adjustments. Alternatively possible court interpretations of a code add to the combinations relevant to the analysis. Hornbook analysis adds information, and along with it, complexity.

The difficulty we have with legal complexity appears to reflect limitations in brain function, the inability to process more than a small number of elements simultaneously. The use of visual models to present complex material is a universally adopted strategy in the war on complexity. The success of visual models may be due to highly efficient visual scanning "programs" in the brain which process visual information in volume.

Visual models are scant in the legal data base. The programmer has his flowchart, the engineer, his diagram. The student of the law has tables of contents, section numbers, paragraphs, and italics. Specifically, the lack of visual representation of logical elements in legal text has been recognized as a serious problem. In proposing "normalized" legal drafting techniques, Allen[11] seeks, along with clarity of intent, a remedy for this problem. Allen proposes that variously expressed logical connectives give way to a small standard set. He suggests special, easily recognized symbols for the con-

[11] Allen, *Symbolic Logic: A Razor Edged Tool for Drafting and Interpreting Legal Documents*, 66 YALE L.J. 833 (1957).

nectives and a prescribed arrangement of statements, making the logical structure of a statute more explicit visually.

The LAWGICAL display carries Allen's proposals a step further. The text is separated from the logical connectives. Each textual premise is replaced in the implication by its name. The logical structure thereby becomes a visual pattern. Were it not necessary to keep the text close by for the separate mental act of interpreting meaning, the logical structure of an entire analysis might be displayed at once. Perhaps it should be anyway, without a nearby explanation, to give the user a visual "radar picture" of complexity turbulence ahead that he might wish to bypass. The action of LAWGICAL reduction has a highly visible effect on an implication, removing from view the elements of an implication which do not have to be dealt with in the user's particular case. In short, the LAWGICAL interactive session exploits visual processing abilities to combat logical complexity.

IV. LAWGICAL COMPARED TO OTHER SYSTEMS

A. *Purpose of the Exercise*

We have considered the primary functions of structure traversal and reduction performed by LAWGICAL, and the pervasive problem that the system design addresses, logical complexity. These matters have presented little opportunity for an exploration of alternatives to the LAWGICAL design, and thereby for an attempt to justify the methods, as well as the objectives, of the system. Comparisons help to clarify issues, and such clarification is especially desirable in an interdisciplinary field such as computer-assisted legal analysis. This introduction to LAWGICAL will therefore conclude with a comparison of the system with contemporary systems whose descriptions are readily accessible. Points of comparison have been selected in order to clarify differences in method, not to evaluate systems. A discussion in depth of the various contemporary systems is beyond the scope of this article.[12]

Figure 4 contains a chart which summarizes in very brief terms the common elements of the systems we will be comparing. The descriptions in the chart are our own interpretations of reports on these systems, worded for purposes of comparison to LAWGICAL, so that we may undertake a separate comparison of the features of each of these systems with those of LAWGICAL.

B. *JUDITH*

As mentioned earlier, several elements of the LAWGICAL design are identical, insofar as information content is concerned, with correspond-

[12] *See* Comment, *Emerging Computer-Assisted Legal Analysis Systems*, 1980 B.Y.U. L. REV. 116.

ing elements of JUDITH. The JUDITH construction file[13] is an implication structure. JUDITH expands a premise when requested to, evaluates logical expressions, makes implication assignments, and returns to higher level implications along a previously traversed path, applying assigned values. In spite of these similarities, the "conversation" one has with JUDITH is quite different from a LAWGICAL session. JUDITH is tight-lipped, telling you nothing you do not ask for. She keeps the direction and intent of her questions to herself. She tells you when you have establised something, but not what routes are available to establish it.

It is not necessary for JUDITH to be so inscrutable. The entire logical expression is available to JUDITH when she asks for the value of its first premise. Why is she so secretive? Perhaps it is a matter of viewpoint. Popp and Schlink regard JUDITH as an artificial intelligence program. Artificial intelligence has been broadly defined in terms of the generation of behavior indistinguishable from that of human consultant.[14] JUDITH can manage that, as long as she asks all the questions. But with similar restrictions, so can a tape recorder. JUDITH is not an artificial intelligence program, because she can evidence no behavior that has not been absolutely prescribed in the strictest sense in her data base. The claim that JUDITH "shows a remarkable resemblance to . . . Thorne McCarty's TAXMAN"[15] is remarkably inaccurate. As a laboratory demonstration, JUDITH impresses, but she is not destined to be "a tool for the practicing lawyer"[16] until she reveals her true personality as a data base management system and becomes a cooperative, information-sharing servant instead of a close-mouthed inquisitor.

---

[13] Popp & Schlink, *supra* note 1, at 307.

[14] *See* Turing, *Computing Machinery and Intelligence*, 59 MIND 433 (1950), *reprinted in* E. FEIGENBAUM & J. FELDMAN, COMPUTERS AND THOUGHT 11 (1965).

[15] Popp & Schlink, *supra* note 1, at 313-14.

[16] *Id.* at 314.

System	*Supporting Actions*	*User's Task*	*Processing*	*Data Base*
LAWGICAL	Prompt, store and deliver a visual form of logical analysis	Fit case into given analysis, identify strategies to establish an implication	Traverse and reduce implication structure	Implication structures premise texts
JUDITH	Conversational consultant	Same as LAWGICAL	Traverse structure, compute implications ask questions	Implication structures, index to structures, premise texts
ABF	Generate text Prompt facts for each blank	Produce a legal form	Prompt questions and query programs Prompt fill-ins and execute query programs	Document forms fill-in queries, procedures
LOGIC	Minimize logical connectives in one implication	Find "simplest" logical expression with one level level of parentheses	Gate minimization methods of switching theory	Truth table description of implication
SEARCH/ CORPTAX	Gather data, classify, logical and arithmetic calculations	Classify a case, obtain numerical results	Execute analysis built into program, prompt user input.	Source program
TAXMAN	Represent conceptual model, answer unanticipated questions	Find cases, classify a case	Semantic structure searching	Semantic data base: values associated through lists

Figure 4. A Comparison Chart for Computer-Assisted Legal Analysis Programs

Akron Law Review, Vol. 15 [1982], Iss. 4, Art. 3

C. *ABF*

Sprowl's ABF system[17] is very dissimilar from LAWGICAL in purpose, but ABF is what LAWGICAL hopes to become, a successful example of ordinary computer technology being put to practical use.[18] ABF assists a contributor in the task of adding a document form to the data base and interprets this information later to assist a user. Along with a document text, the contributor enters marked descriptions, which the system plays back to the user as questions, whose responses will fill blanks on the form. For some blanks, the contributor enters short program procedures, in a simple language developed for the system, which will prompt information from the user to select which questions to ask. Sprowl's system, like LAWGICAL, exhibits widely different behaviors as it presents to a user different portions of the data base.

LAWGICAL also helps contributors add implication structures to the data base. Facilities include a prompting program, and programs for packing the analysis and adding it to the LAWGICAL analysis file. The analysis file can be read, but not rewritten, by all users. The prompting program asks for logical expressions and text by following checklists supplied by the contributor. The prompting program provides an interface for the contributor who is not an APL programmer.

D. *LOGIC*

The digital logic minimization program LOGIC[19] is included in this survey because its recent citation[20] as a tool for legal drafting may be cause for serious misunderstanding about the LAWGICAL logical expressions. Among other functions, LOGIC determines the logical expression with the minimum number of AND ($\wedge$) and OR ($\vee$) connectives,[21] given that any premise appears but once in the expression and that parentheses are not found within other parentheses.[22] These restrictions are not at all appropriate for implication expressions, or for legal drafting, for that matter, and nothing like them is required for LAWGICAL.

Logic minimization as performed in LOGIC bears no relation to LAWGICAL reduction or evaluation of logical expressions. Minimization is a much more difficult problem, in which the number of required oper-

---

[17] Sprowl, *Automating the Legal Reasoning Process: A Computer that Uses Regulations and Statutes to Draft Legal Documents*, 1979 AM. B. FOUND. RESEARCH J. 3.

[18] The title of Sprowl's American Bar Foundation report, *supra* note 17, unfortunately is overstated beyond comprehension.

[19] R. CROES & C. ROTH, LOGIC USER'S MANUAL (Courtesy of Professor C.H. Roth, Department of Electrical Engineering. The University of Texas at Austin. On file at AKRON LAW REVIEW).

[20] Edwards & Barber, *A Computer Method for Legal Drafting Using Propositional Logic*, 53 TEX. L. REV. 965 (1975).

[21] R. CROES & C. ROTH, *supra* note 19, at 1.

[22] A characterization, in terms of the LAWGICAL expression, of the two-level gate function.

ations is an exponential function of the number of premises. LAWGICAL reduction and evaluation are based on a character-matching process in which the number of operations is proportional to the length of the logical expression, and these computations can never become burdensome.

E. *CORPTAX*

Hellawell's CORPTAX[23] program establishes the logical implication that a stock redemption "qualifies for favorable tax treatment under the 'substantially disproportionate' provisions of the Internal Revenue Code."[24] This classification is made by "asking the user questions and recording his answers, almost always 'yes', or 'no', or a number."[25] CORPTAX also prompts numerical information in order to follow "a chain of attribution from trust to corporation to estate, and so on."[26] CORPTAX must determine the route of stock attribution "which yields the highest total attribution of ownership to the taxpayer."[27]

LAWGICAL is a facility for exploring analyses. CORPTAX, like its successor SEARCH, is an analysis in the form of a computer program. For convenience we will call such a program a programmed analysis. Hellawell's programs represent a valuable line of research, and offer much beyond the programmed analysis. Programmed analysis as a general strategy, however, has serious disadvantages. A computer program defines a fixed structure, allowing little variation in the sequence of actions. Significantly more program instructions are necessary to permit expansions and anything more than an "inquisitor" interface with the user. In the programmed analysis, reduction of the structure based on user responses is impractical. Besides these performance restrictions, there are also practical disadvantages. The builder of the data base must be a computer programmer, rather than another type of user. Hellawell expresses well-founded concern about the manpower and funds required to build and update programs like CORPTAX.[28] In the programmed analysis, the extra coding required for a friendly and capable interaction with the user is done repeatedly, rather than once, as in LAWGICAL. Dispersion of programmed analyses over many computer languages and operating systems will mean that few are accessible to many. Storage and retrieval of logical structure in the form of sequences

---

[23] Hellawell, *A Computer Program for Legal Planning and Analysis: Taxation of Stock Redemptions,* 80 COLUM. L. REV. 1363 (1980).

[24] *Id.* at 1363.

[25] *Id.*

[26] *Id.* at 1368.

[27] *Id.* Hellawell's latest program, SEARCH, also involves analysis of the problems of stock attribution under I.R.C. § 318, as modified by I.R.C. § 304. The description of CORPTAX applies equally in most respects to SEARCH. *See* Hellawell, *SEARCH: A Computer Program for Legal Problem Solving,* 15 AKRON L. REV. 635 (1982).

[28] Hellawell, *supra* note 23, at 1393.

of computer instructions will be very inefficient, compared to storage as strings of characters, the natural form of longer term computer storage.

The previous paragraph suggests that "yes and no" sequences in programs like CORPTAX are best provided in data management programs like LAWGICAL. But what about numerical computations such as those performed by CORPTAX in the area of stock attribution? For large data bases, it may be best to imbed such calculations in the implication structures. By an imbedded calculation, we mean that when some premise expansion is requested, a function is executed which interacts with the user in its own unLAWGICAL way, prompting numerical data and displaying calculated results. The imbedded function would be part of the structure and not part of the program which processes structures, so it will be taking up fast access, expensive memory only when its host structure is being explored. The ABF system mentioned earlier uses this method, imbedding prompting functions in document forms.

The function imbedding capability just described has been provided in the APL version of LAWGICAL now available, permitting applications like CORPTAX to be imbedded. It is necessary that the functions be defined as APL functions which return a true or false value and that these functions be in the contributor's workspace when the implication structure and text are packed for long-term storage.

Before leaving the programmed analysis, we should examine the reasons for the above statements about its restricted range of interactive behavior. To make the point, it is necessary to illustrate the individual steps in the program, at the level of detail of individual statements in a computer language. This digression will involve programming details not of interest to every reader. Consider the simple implication

$$(A \vee B) \wedge \sim C \rightarrow D.$$

The flow chart in Figure 5(a) represents an "inquisitor" program that does not allow the user any escape from a "yes" or "no" response. The user who is certain of B and C, but unsure of A, is put to a disadvantage by this line of questioning. We can allow for uncertainty by permitting a "pass" reply, as in Figure 5(b). Tests must often be made for the occurrence of any passes within a series of ANDs ($\wedge$) or ORs ($\vee$). A pass within a series of ORs (alternatively, ANDs) means that we cannot be sure that the series is false (true). If there is at least one yes (no) in such a series, we can be sure that it is true (false), in spite of the pass. The test is not always needed, because there may be no value in determining that the series is false (true), but only in knowing that it has not yet been proven true (false). Handling of passes requires initialization and assignments of values to extra variables, and testing their values when the series of ANDs or ORs are completed.

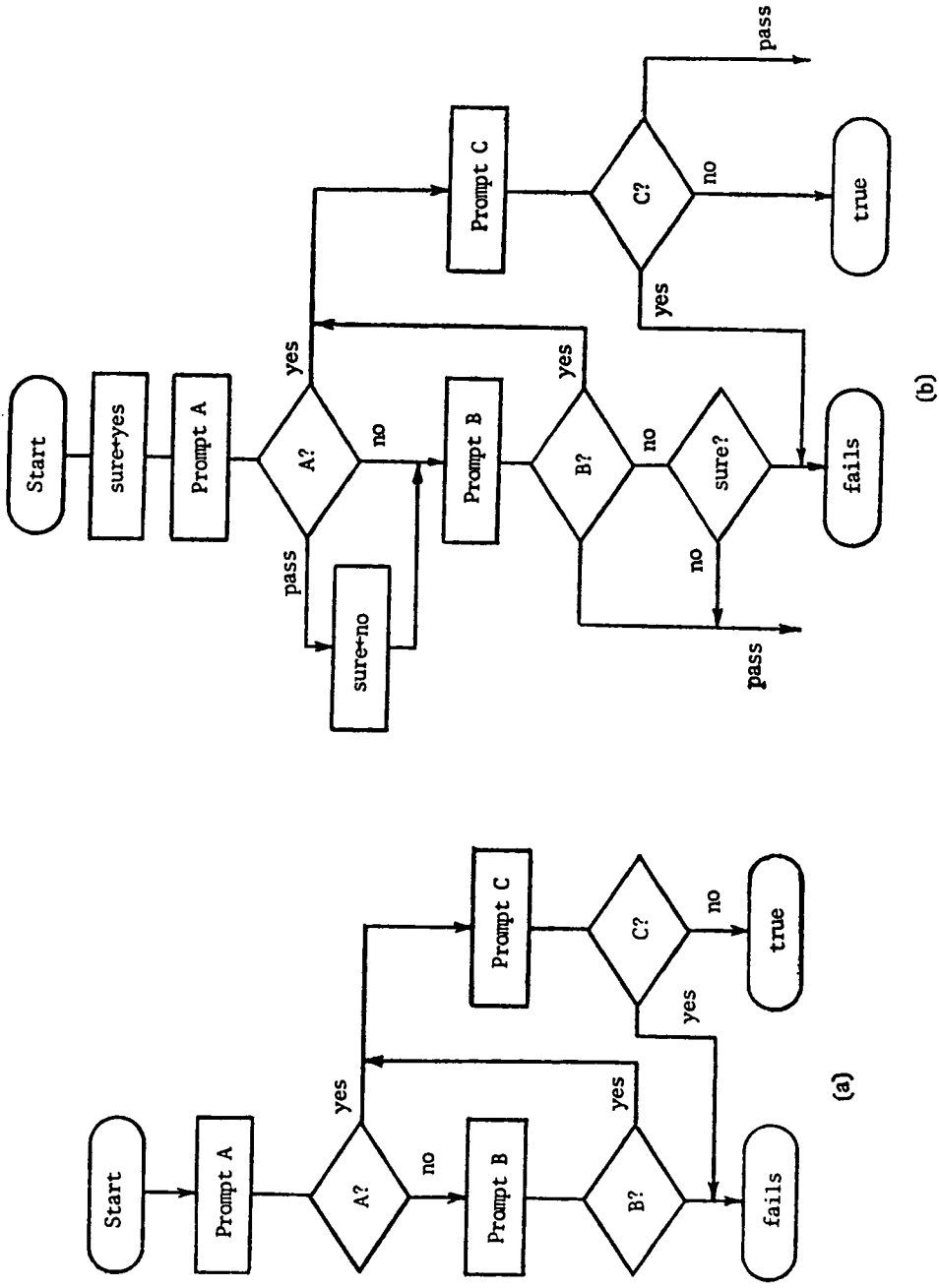

Figure 5. An Intolerant Inquisitor and a More Tolerant One.

Assignments to all premises of an implication at one time avoids imposing an order and it can be done as illustrated in Figure 6(a). This is not equivalent to the LAWGICAL TF prompt, because every premise must be given a value. LAWGICAL reduces the expression by what it is given, but in the programmed analysis, an expression must be evaluated fully. A separate list of values may be kept, so that the user has only to change the values previously assigned, to which he now takes exception.

An expansion capability comparable to LAWGICAL's can be provided by organizing the implication structure as a nest of subroutines. Under this plan, the flowchart of Figure 6(a) would represent a subroutine D which returns one of three values (yes, no, fails). The prompt and each decision would be coded to accommodate a fourth response for each premise which is a conclusion, and which therefore has an implication subroutine. Figure 6(b) shows how the decision about B would be coded for this case. One must allow for a direct yes, no, or pass, or call to the subroutine for expansion. If the implication for B were

$$E \wedge F \wedge G \rightarrow B,$$

the subroutine could be coded as shown in Figure 6(c). Clearly these methods permit no reduction capability, because it is impractical to have a program do major surgery on itself during execution.

In the previous paragraphs, we have outlined programming techniques which permit a programmed analysis some degree of LAWGICAL capability in exposing implication structures. We have not been describing methods used in LAWGICAL. The LAWGICAL processing routines operate on implications as data, giving the user the same options with respect to each. Our purpose has been to show that desirable interactive behavior in programmed analysis is purchased at considerable cost in terms of programmer time and memory for instructions.

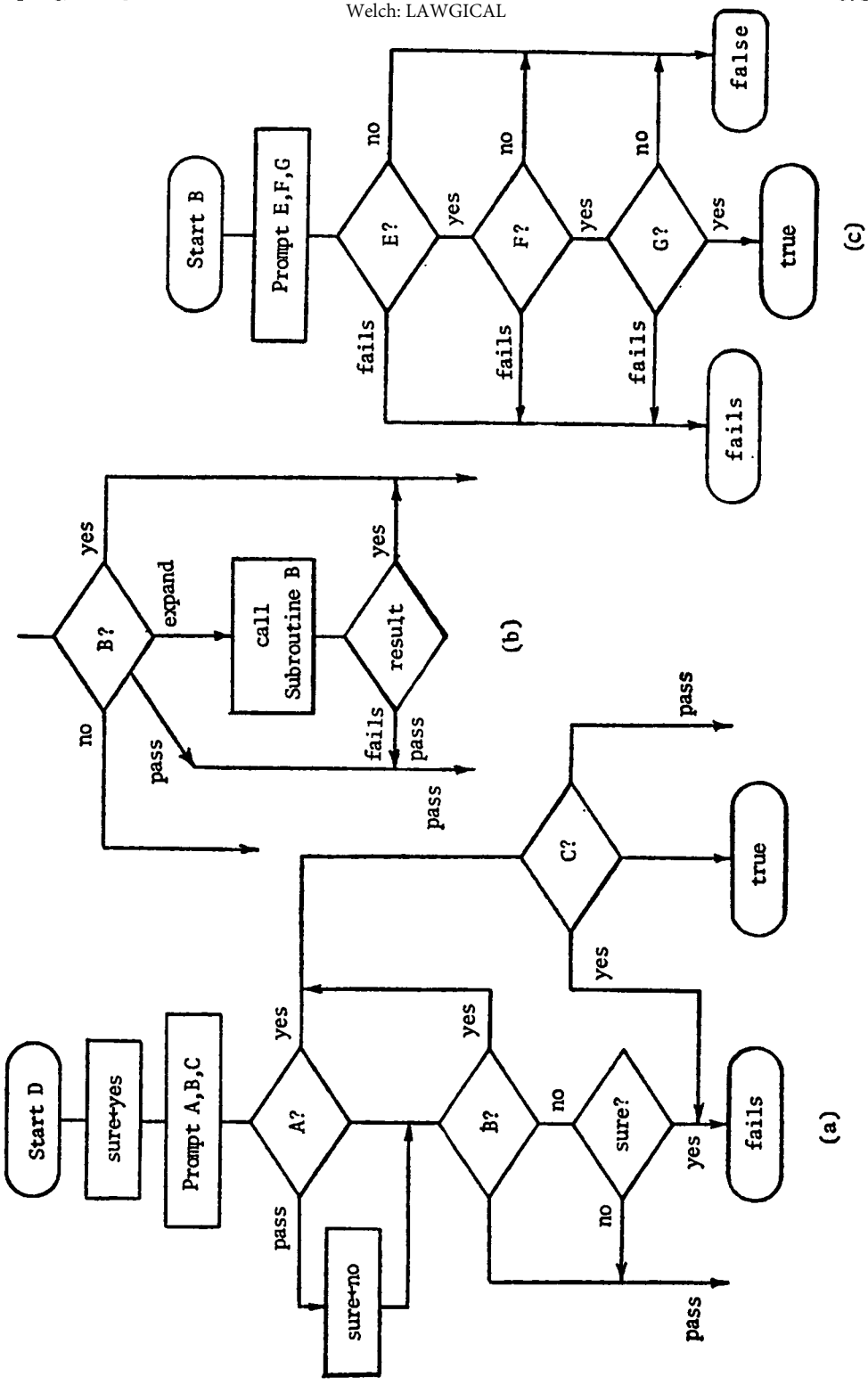

Figure 6. One-prompt Organization and Coding for Expansions.

F. *TAXMAN*

McCarty's TAXMAN[29] is an example of basic research in the application of artificial intelligence concepts to legal problems. Results of basic research are often unexpected, but McCarty sees two principal benefits: namely, the development of more powerful tools for legal analysis, and a more thorough understanding of legal reasoning. The successful extension of TAXMAN into a commercial system would make it possible to enter factual data and models of legal concepts in a straightforward manner, and to combine them in a wide variety of ways to answer questions unanticipated when the data was entered. Better understanding of legal reasoning could result from building, exercising and evaluating increasingly sophisticated models made possible by the semantic information processing technology on which TAXMAN is built.

LAWGICAL, in stark contrast, is not expected to add new elements to the practice of legal analysis. Greater productivity and accuracy could be expected, though, through the enhancement of the interface between human and elaborate logical structure. Every increment of that gain is achieved for the many through the considerable efforts of a few — the analyst contributors to the LAWGICAL data base, whose task is far from straightforward. The contributing analyst must anticipate, in a general way, all questions to be resolved. Whereas TAXMAN is a precursor of "intelligent" systems, LAWGICAL is a medium for the transmission of intelligence from contributing analyst to using analyst.

Can systems based on artificial intelligence concepts become power-tools in legal analysis? One may be hopeful that they can. Such systems may be destined, however, for small, specialized, highly organized data bases. Data base searching is the primary processing activity in artificial intelligence systems like TAXMAN, and, as the data base grows larger, the need for mechanisms to confine the search becomes more acute. It is not simply a matter of defining algorithms that work on a small data base, and then making routine adjustments for size as the data base grows.

V. CONCLUSION

As we have seen, LAWGICAL is an information management system designed to assist in legal analysis by enhancing the analyst's ability to deal with logical complexity. The system provides an undemanding, highly visual interface which permits the user to suit the exploration of the logical structure to his immediate needs. LAWGICAL processes an unrestricted form of implication structure as data, providing for traversal and reduction of the structure. Imbedded functions can provide extended capability

---

[29] McCarty, *Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning*, 90 HARV. L. REV. 305 (1977).

as needed. The system also aids contributors in adding implication structures to the data base.

Many aspects of the LAWGICAL system require investigation before its design can be considered complete. The data base structure is largely untried as a representation of many kinds of legal data. There is considerable choice in the packaging of data within the structure. Trials with different styles of packaging may be revealing. With more extensive use, faults may be found in the style and content of the LAWGICAL interactive session, and additional features may be found to be desirable.

Implementation in APL on a central time-sharing system is well-suited to the period of design refinement and data base building anticipated for the immediate future. However, since processing requirements for LAWGICAL are not high, the best configuration may be a decentralized one in which microcomputer systems work from implication structure libraries on floppy disks. Floppy disks might be distributed and routinely updated by an information service. The existence of such systems and services may affect legal drafting and legal analysis practice, but only as the LAWGICAL communication medium and LAWGICAL data bases find favor in the information marketplace.