WIS-CS-168-73
University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

FUZZY PLANNER

Computing Inexactness in a Procedural Problem-Solving Language

by

Rob Kling

Technical Report #168
February 1973

Received January 30, 1973

ABSTRACT

FUZZY PLANNER

Computing Inexactness
in a Procedural Problem-Solving Language

All contemporary deductive problem-solving paradigms deal with a world in which assertions are true (false) and action-rules valid (invalid). This simplified situation is inadequate for realistic applications which include inexact information. This report describes a precise computationally specific method for coupling two different many-valued logic with a procedural problem-solving system (PLANNER). Solutions to deductive problems can be found which meet specific criteria of validity. This particular scheme enables the system to dynamically compute the truth-value of a subgoal during the search process. Thus, the validity of a subgoal may be used to direct the heuristic search procedure.

Fuzzy PLANNER is a promising medium for experimenting with different many-valued logics to find the ones most appropriate for different problem domains.

FUZZY PLANNER

I. Introduction

A robot that reasons out actions to manipulate the world around him deals in a world of some imprecision. He does not always perceive a scene with complete accuracy; when he does, it still may change. The causal laws he uses in reasoning about actions may have some ambiguity. Unfortunately, all the deductive problem solving systems (e.g., QA3, STRIPS, PLANNER) that have been developed to aid a robot assume that all the information is exact. These systems are restricted by the conventions of classical two-valued logics. Here I will discuss how a deductive problem solver can deal with certain kinds of inexactness by redesigning a particular paradigm (PLANNER) to allow the use of a many-valued ("fuzzy") logic.

In many realistic settings a two-valued logic may be too constrictive a model. For example, suppose we wish to describe the intensity of a person's pain for purposes of medical diagnosis. We could have a large set of predicates which uniquely describe each gradation of pain. A practical axiom system utilizing such a large set of predicates in many similar inferences would be unwieldy. Alternately, we could allow a single predicate PAIN [X] to be interpreted as X is in pain. The truth value of PAIN[X] (T[PAIN[X]]) could be used as an index of intensity. Then T[PAIN[JOHN]] = .6 could indicate that John is in mild pain. T[PAIN[JOHN]] = .9 would indicate that John is in acute pain. Many descriptions in the medical literature (Mellinkoff [1959]) utilize qualifiers such as mild, acute severe, predominant, marked, slight, etc. We would like a diagnostic system which uses deductive inference to deal with such inexactness in a meaningful and precise way.

In a less dramatic setting, consider a robot which is asked to bring us three "tasty" apples. Tasty is not a precise description. We would really like the robot to bring us the three tastiest apples it can find (nearby). We would like our robot to solve problems which involve imprecision with human-like flexibility.

In the preceding discussion I suggested that statements assume truth-values on the interval [0,1]. The multi-valued logics or these statements may be good candidates for expressing inexact concepts and making inferences with them. At this time we have few analytically precise alternatives. This paper outlines a computationally precise scheme for a procedural problem-solver (PLANNER) to use a many-valued logic. For brevity, I will assume that the reader has some familiarity with the class of robot manipulation problems current in artificial intelligence (Raphael [1970]), has some acquaintance with fuzzy sets (Zadeh [1965]) and is conversant with the notions of procedural problem solving represented by PLANNER (Hewitt [1971], Winograd [1972]).

Each author proposes his particular logic as <u>the</u> "fuzzy-logic" (Goguen [1968], Lee [1972]). Actually, many distinct consistent fuzzy logics are possible. The appropriate meta-theory--the theory of multi-valued logics--is introduced in the next section. This section may be skipped on a first reading by the less mathematically inclined reader.

II. A Brief Introduction to Multi-Valued Logics

Multi-valued logics deal with statements which can assume truth-values on the interval [0,1]. We can construct such a calculus analogous to the first-order predicate calculus by creating a set of primitive atomic statements (P_i) and (2) a set of function of statements $\{F_i\}$:

$$F_1(S_1,\ldots,S_{a1})$$

$$F_2(S_1,\ldots,S_{a2})$$

$$F_b(S_1,\ldots,S_{ab}) \qquad b\geq 1$$

$$a_i\geq 1$$

Then, for $1<i\leq b$, and $S_1,\ldots,S_{a_i}$ are statements, $F(P_1,\ldots P_{a_i})$ is a statement. Common examples of F_i are ~ , ⇒, and V . Thirdly, we divide the interval [0,1] into $[0,\alpha)$ and $[\alpha,1]$. If $x\varepsilon[0,\alpha)$ x is called "undesignated" and corresponds to falsity in the two valued calculus. Likewise, if $x\varepsilon[\alpha,1]$, x is "designated" and corresponds to "truth". Lastly, we assign a truth function $f_i[s_1,\ldots,s_{a_i}]$ to each $F_i[S_1,\ldots,S_{a_i}]$ such that if $T[S_i]=s_i$ (the truth-value of S_i), then, $T[F_i[S_1,\ldots,S_n]] = f_i[s_1,\ldots,s_a]$ ε [0,1] . We can then select connectives for the F_i such as ~ and V , associate f_i with these, and develop a scheme for assigning truth-values to any woff. Then we set up axioms schema for our logic and explore a multivalued logic by considering its tantologies. This approach is the typical axiomatic development of multi-valued logic (Rosser and Turquette [1952], Rose and Rosser [1958]. This approach allows considerable laxity in selecting the statement functions F_i and their associated truth-valuesfunctions f_i .

For example, the "standard condition" for "S_1 and S_2" is designated iff $T[S_1]$ and $T[s_2]$ are both designated. (Rosser and Turquette [1952].)

The "standard conditions" allow us tremendous freedom in choosing truth-value functions.

Let $T[F_1[S_i;S_2]]=f_1[s_1;s_2]= \min[s_1;s_2]$ and $T[F_2[S_1,S_2]]=f_2[s_1;s_2]=\max[0;s_1+s_2-1]$. The reader can verify that both F_1 and F_2 are analogous to conjunction in two-valued logic. Either is an acceptable candidate for conjunction in multi-valued logic. The remaining standard conditions are:

(2) "S_1 or S_2" is undesignated iff s_1 and s_2 are both undesignated.

(3) "S_1 implies S_2" is undesignated iff s_1 is designated and s_2 is undesignated.

(4) "not S" is designated iff s is designated.

For example, the set of truth-value functions proposed by Lee for "fuzzy resolution" (Lee [1972]) constitute a special many-valued logic in which $\alpha=.5$

(1) $T(S) = T(A)$ if $S=A$ and A is fully initiated.

(2) $T(S) = 1-T(R)$ if $S= R$

(3) $T(S) = \min[T(S_1),T(S_2)]$ if $S=S_1\ S_2$

(4) $T(S) = \max[T(S_1),T(S_2)]$ if $S=S_1\ S_2$

(5) $T(S) = \inf[T[B(x)]\ x\ D]$ if $S= xB$ and D is the domain of x

(6) $T(S) = \sup[T[B(x)],x\ \ D]$ if $S= xB$ and D is the domain of x .

(7) $T[S] = \max[1-T[S_1];\ T[S_2]$ if $S=S_1 \Rightarrow S_2$

(We will call any many-valued logic that satisfies (3) and (4) a "fuzzy logic".) Notice that Lee's valuation of implication can be rewritten as the following ALGOL-like expression:

$$(8)\quad T[S] := \textbf{if}\ T[S_1] \geq .5\ \textbf{then}\ T[S_2]\ \underline{\textbf{else}}\ \max[1-T[S_1], T[S_2]]$$

In a practical deductive system we need a rule for detatchment that will enable us to compute the truth-values of consequences of our premises. Given S_1, $S_1 \Rightarrow S_2$, $T[S_1]$ and $T[S_1 \Rightarrow S_2]$ we want to compute $T[S_2]$. We seek a binary operation $*$ such that:

$$(9)\quad T[S_1] * T[S_1 \Rightarrow S_2] \leq T[S_2]$$

We do not want the truth value of the consequence we deduce to exceed $T[S_2]$. On the other hand, we would like to have it as large as possible. Most accounts of many-valued logic neglect this issue since logicians are more concerned with the tautologies and axiomatic basis of their logics than with providing computationally specific deductive procedures. Lee was interested in providing a computationally specific marriage of "fuzzy logic" with resolution. For his definition of $T[S_1 \Rightarrow S_2]$ ((7) above) he proved the weak result:

$$(10)\quad \min[T[S_1], T[S_1 \Rightarrow S_2]] \leq T[S_2] \leq \max[T[S_1], T[S_1 \Rightarrow S_2]]$$

Thus, min can be used to estimate $*$ and provide a (conservative) lower bound for the truth-value of a consequent.

Alternate methods for computing $T[P \Rightarrow Q]$ and $*$ can be generated by studying the algebraic properties of the lattice of propositions generated by our logic. In his study of a variant fuzzy logic which satisfies (1)-(5) and in which $\alpha=0$, Goguen created algebraic constraints on $*$ and developed a related definition for $T[P \Rightarrow Q]$ (Goguen [1968]). He argues that we would like $T[S_1 \Rightarrow S_2]$ as large as possible, but subject to the constraint of (9). Then, if we know both $T[S_1]$ and $T[S_2]$,

$$(11)\quad T[S_1 \Rightarrow S_2] = \sup_x \{X \mid T[S_1]*x \leq T[S_2]\} .$$

Furthermore, if we want $(S_1 \Rightarrow S_2) \wedge (S_2 \Rightarrow S_3) \Rightarrow (S_1 \Rightarrow S_3)$ then $*$ should be associative. Goguen adds several additional constraints on $*$ and suggests that a good interpretation of the algebra of propositions is a complete-lattice-ordered semigroup (Birkhoff [1967]). Then, he argues that multiplication is a good candidate for $*$. It follows that:

$$(12)\quad T[S_1 \Rightarrow S_2] = \begin{cases} \dfrac{T[S_2]}{T[S_1]} & \text{if } T[S_1] \geq T[S_2] \\ 1 & \text{otherwise.} \end{cases}$$

One interesting property of Goguen's development is that a chain of nearly valid implications decreases as the chain increases in length. In contrast, Lee's fuzzy logic bounds the validity of a chain of implications by the truth-value of the least valid element in the chain.

The point of this development is to show that we have a great deal of freedom in choosing $*$ and $T[P \Rightarrow Q]$ in a fuzzy-logic. Our particular choice may well depend upon the kind of reasoning we are trying to model. Unfortunately, Lee's logic is not of much use in the context (Resolution logic) for which he developed it. The failure of his logic in that setting and our desire to find a more tractable setting for its use motivated this development of Fuzzy PLANNER. Both Lee's and Goguen's developments are viable logics and either one can be successfully embedded in Fuzzy PLANNER.

III. Deductions with Fuzzy-Resolution

Lee [1972] clarifies the relationship between fuzzy-logic and resolution. He proves the weak result that the truth-value of a resolvent is bounded (above and below) by the truth-values of its parent clauses. Each time we generate an inference we would like to know its truth-value. Then we could formulate strategies which would allow us to give priority to making deductions from the "truest" inferences first. Unfortunately Lee's result doesn't allow us to use resolution easily that way. Suppose we wish to find a ripe apple. We could phrase our request as the following theorem (Green and Raphael [1969]).

$$S_1 : \exists\, x\ \mathrm{apple}[x] \wedge \mathrm{ripe}[x]$$

In a resolution theorem prover (using T-support; Wos [1965]) we would start resolving S_1 with axioms $C_1, C_2, \ldots, C_k$.

Now, what are the truth-values of the resolvents $\{R_j\}$?

$$R_1 = R(S_1, C_1)$$

$$R_2 = (R\ S_1, C_2)$$

$$\vdots$$

$$R_j = R(S_1, C_j)$$

Let $T[C]$ be the truth value of clause C . Then, by Lee's result:

$$\min[T[S_1], T[C_j]] \leq T[R(S_1, C_j)] \leq \max[T[S_1], T[C_j]]$$

Suppose we know the truth-value of each axiom C_j . We do not know the truth-value of S_1: that will depend upon whether there is an x that satisfies $S_1[x]$. If there is such an x , e.g., a ripe apple, the truth-value of $S[x]$ will depend upon which ripe apple we choose for x . The truth-value of S_1 is unknown. Thus, the truth-value of S_1 is also unknown, a "?".

Thus, $\min[?,T[C_j]] \leq T[R_j] \leq \max[?,T[C_j]]$. $T[R_j]$ is rather uncertain. We might attempt to estimate it by $T[C_j]$. But there is still some uncertainty since we are unsure whether $T[C_j]$ is an upper or lower bound on $T[R_j]$. These considerations motivate us to select another system as a candidate for creating a computationally attractive "fuzzy" problem solver.

IV. Truth-Values in Fuzzy PLANNER

A PLANNER assertion, e.g., (RIPE APPLE7) is true if it appears in the data base. In Fuzzy PLANNER, we want to associate a truth-value $\tau \in [0,1]$ with each assertion. Thus (THASSERT (RIPE APPLE7) .9) means APPLE7 is RIPE with truth-value .9 . If an expression e as truth-value T[e] , we can interpret T[e] as the (fuzzy) membership function of e (Zadeh [1965]). If T[(RIPE APPLE9)] = .8 , we can say that APPLE9 belongs .8 to the set of RIPE things. Fuzzy truth-values are not probabilities. We are describing deterministic events.

PLANNER's means of accessing an assertion is the THGOAL statement. (THGOAL (RIPE X)) will get me a RIPE thing. We should be able to ask for a "very" RIPE thing, e.g., (THGOAL (RIPE X) τ) . This goal should be satisfied by an X s.t. T[(RIPE X)] $\geq \tau$. In "classical" PLANNER , expressions are either True (and THSUCCEED) or False (and THFAIL) .

Fuzzy PLANNER should allow a PLANNER statement to succeed or fail based on the truth-value of the expressions that match the statement compared to some threshold.

Lee selects $\alpha = .5$ as his lower bound for designated truth-values. Thus $T \in [0,.5)$ corresponds to "false" in two-valued logic. Any $\alpha < 1$ is formally adequate; the virtue of $\alpha = .5$ is the symettry it provides. In the following discussion we will assume $\alpha = .5$, but it may be changed without loss of generality.

V. Truth-Values for Primitive PLANNER Expressions

Let's now consider the truth-values of the PLANNER primitives:

(1) T[THSUCCEED] = 1

(2) T[THFAIL] = 0

(3) T[(THGOAL e τ)] = T[A] if (i) A matches e
(ii) A is THASSERTED
(iii) T[A] $\geq \tau$
$1-\tau$† otherwise (we will consider goals that are satisfied by THCONSES later)

(4) T[(THAND e_1 e_2... e_n)] = min[T[e_1],...T[e_n]]

(5) T[(THOR e_1,e_2...e_n)] = max[T[e_1],...T[e_n]

(6) T[(THNOT e)] = 1-T[e]†

(7) Let e = (THCOND (p_1 e_{11} e_{12}...e_{1n})(p_2 e_{21}...e_{2n}) .... (p_m e_{m1}...e_{mm})) .

Then e can be rewritten as (THOR (THAND p_1 e_{11} e_{12}...e_{1n}) (THAND p_2 e_{21}...e_{2n})... (THAND p_m e_{m1}...e_{mn}))

Thus T[e] = max[min[T[p_1],T[e_{11}]...T[e_{1n}]],min[T[p_2],T[e_{21}] . T[e_{2n}]]....min[T[p_m]T[e_{m1}]...T[e_{mn})]]

(8) T[(THFIND ALL x e)] = $\min_x$ T[e] (x is a list of all objects that satisfy e)

(9) T[(THFIND n x e)] = $\min_x$ T[e] (x binds a list of n objects that "best" satisfy e)

(10) T[(THPROG (X) e)] = T[e] since THPROG acts like a THFIND that binds one element which satisfies e to x We will consider THPROGS with loops later on.

Several PLANNER primitives, such as THFIND, can allow several expressions to be evaluated. Thus

---

† The evaluations are consistent with both Lee's and Goguen's logic. Other forms of negation may be used. See Goguen [1968].

(THFIND ALL (X) (COLOR X RED) (TYPE X APPLE)) will return a list of all the red apples. There is an implicit THPROG in this and similar statements which must be considered in computing T[e] as in (8) and (9) above.

Lastly, we come to THCONSE, the "backwards" implication of PLANNER. If we want to say $p(x) \Rightarrow q(x)$, we write (THCONSE (X) (Q X)) (THGOAL (P X)) . If we satisfy (P X) , with X=A then we infer (Q A) . This is much like:

S_1: p(x)

S_2:p(x) $\Rightarrow$ q(x)

$T[q(a)] \geq T[S_1]*T[S_2]$ (The operation * was introduced in Section II.)

From our discussion of many-valued logics in Section II we know that we have quite a bit of freedom in choosing * and $T[P \Rightarrow Q]$. The operation * for detatchment corresponds to the truth-value function we select for THCONSE . While our truth-value function for $\Rightarrow$ will be reflected by our computation of $T[P \Rightarrow Q]$, e.g., $T[S_2]$. At present we will not select specific computations. Rather, we will leave * unspecified and assume that we have some way of assigning truth-values to conditionals like S_2 . Let us denote a THCONSE with its assigned truth-value τ_0 as (THCONSE vars τ_0 e_0 e'). In this format e_0 represents an expression which can match a THGOAL statement and e' is the THCONSE body (with an implicit THPROG) which will be evaluated. Then,

(12) T[(THGOAL e τ)] = T[A] if (1) A is in the data base
(2) A matches e

	(3) $T[A] \geq \tau$
$T[(\text{THGOAL } e\ \tau)] \geq T[e']*\tau_0$ THCONSE theorem (THCONSE	if there is a vars τ_0 e_0 e') s.t.
	(1) e_0 matches e
	(2) this theorem THSUCCEEDS.
	(3) $\tau_0 > \tau$ and $T[e']>\tau$

This brief discussion outlines the assignment of truth values to most of the PLANNER primitives whose execution will result in returning an expression with some associated truth-value. Note in passing that certain PLANNER primitives such as THGO do not have meaningful truth-values.

Now let's return to THGOAL. A THGOAL may be satisfied by directly matching some item in the data base or by triggering a THCONSE theorem which THSUCCEEDS. We would like to prefer THCONSES which have adequately high truth-values to allow success. We could order these by their truth-values and use truth-value as an estimate of utility.

```
TH1:  (THCONSE (X) .6 (TASTY X)
                      (THGOAL (APPLE X) 1)
                      (THGOAL (RED X) .7))

TH2:  (THCONSE (X) .9 (TASTY X)
                      (THGOAL (RIPE X) .95))
```

Suppose we want a tasty apple: The appropriate PLANNER expression is:

```
(THPROG (Z) (THGOAL (APPLE X) 1)
            (THGOAL (TASTY Z) τ) )
```

If we want any tasty apple, let $\tau = \alpha$. Suppose PLANNER satisfies (THGOAL (APPLE Z) 1) for Z = APPLE7, but must invoke a THCONSE theorem to satisfy (THGOAL (TASTY APPLE7) .5) . Then PLANNER should try TH2 before attempting TH1 .

Suppose $*[x;y] = \min[x;y]$. $T[(\text{THGOAL } e\ \tau] \geq \min[\tau_0;T[e']]$. If $\tau > \tau_0$, then a THCONSE with truth-value τ_0 will be useless in satisfying such a goal. Such THCONSES should be rejected as candidates. Then, if we want a very tasty apple and set $\tau = .85$, then we should try only TH2 since $T[(\text{TASTY APPLE7})] \leq .6$ with TH1 . If we are very picky and ask for an extremely tasty apple $\tau = .95$, then neither TH2 nor TH1 should be invoked.

VI. Evaluation of Fuzzy PLANNER Expressions

In the preceding section, I have implied the evaluation procedure for THGOAL and hinted at one for other PLANNER primitives. Some of the underlying issues will be treated for THAND and extended later. They include:

(1) Threshold
(2) Back-track
(3) Loops
(4) Satisficing.

A PLANNER (THAND $e_1 \ldots e_n$) is analogous to LISP (AND $e_1 \ldots e_n$) in that it will evaluate the e_j's until one of them THFAILS (in LISP e_j=NIL). Otherwise it THSUCCEEDS. (LISP 1.5 returns T)

With fuzzy-truth how do we know when to fail?

Let e_0 = (THAND e_1 e_2 e_3 e_4 e_5 e_6)

$T[e_1] = .9 \qquad T[e_2] = .7 \qquad T[e_3] = .6$

$T[e_4] = .4 \qquad T[e_5] = .2 \qquad T[e_6] = .8$

Now $T[e_0] = \min_{1<j<6} T[e_j] = .2$

In fuzzy-logic, $T[e] \geq \alpha \Rightarrow T[e] \approx$ True. How many of the e_j should PLANNER evaluate before quitting? Usually we want to stop when we find the first "false" e_j, e.g., $T[e_j] < \alpha$. Suppose we use Lee's fuzzy-logic with $\alpha = .5$. In this example, $T[e_0] = .4$ and we would quit after evaluating e_4. Then, we would estimate $T[e_0]$ by .4. Alternately we may want to stop if the reliability of the e_j's drops below some threshold k. Thus, if $k = .7$, we will stop evaluating e_0 after we find $T[e_3] = .6$. At that point, $T[e_0] < .7$ and we might want to say $T[e_0] = 1-k = .3$. While this scheme is appealing, it seems superfluous. Usually the e_i in a THAND are THGOAL statements. Their truth values may be controlled by setting their truth-value thresholds to the appropriate level ($\geq k$) . Thus, an additional parameter seems unnecessary. Our two choices are

(1) Evaluate all the e_j and compute the value of THAND by formula 4 given in the preceding section

(2) If $T[e_j] < .5$, stop evaluation and assign a truth-value of $T[e_j]$.

The second solution is the most efficient, although it yields an estimate of the truth-value of the THAND.

Consider asking PLANNER to find a ripe apple: (THAND(THGOAL(RIPE X))(THGOAL(APPLE X))) . Suppose we have THASSERTED:

e_1: (RIPE APPLE1)	$T[e_7] = .85$
e_8: (RIPE APPLE75)	$T[e_8] = .9$
e_9: (RIPE BX75)	$T[e_9] = .95$
e_{10}: (APPLE APPLE1]	$T[e_{10}] = 1.0$
e_{11}: (APPLE APPLE75)	$T[e_{11}] = 1.0$
e_{12}: (BANANA BX75)	$T[e_{12}] = 1.0$

The PLANNER evaluator will make a list of all the candidates for (RIPE X) , e.g., (e_9, e_8, e_7) . Later if one THFAILS, it will attempt the next choice on the list. In this example, if e_9 is chosen first with X=BX75 , PLANNER will attempt to satisfy (THGOAL (APPLE BX75)) .

This will THFAIL and PLANNER will backup to the place it made its last choice and try again with e_8 , (X=APPLE75) . In classical PLANNER this list is randomly ordered. In Fuzzy PLANNER, this list should be ordered by truth-values of the candidates assertions that can be used to satisfy a THGOAL. In brief, we should use a simple heuristic "Try the truest choices first."

How shall we keep track of the partial evaluation of the truth-value of an expression to allow for back-tracking? Suppose that e_0 = (THAND e_1 e_2 e_3 e_4 e_5 e_6) and a variable X is bound to $\{x_1, x_2, x_3\}$ in e_3 .

At the time that we evaluate e_4 and e_5 with $X=X_1$, we have evaluated $T[e_1]$, $T[e_2]$, $T[e_3(x_1)]$, $T[e_4(x_1)]$ and $T[e_5(x_1)]$.

We know: $T[e_0] \leq \min[T[e_1], T[e_2], T[e_3(x_1)], T[e_5(x_1)]$.

We would like to carry along our partial evaluation of $T[e_0]$ based upon the e_i we have already evaluated. If it falls below α (for a THAND) PLANNER should stop evaluating any of further e_i . Suppose $[e_6(x_1)] = .2$. Now PLANNER should THFAIL, and back up to its last choice point e_3 and try $X=x_2$. What about our partial evaluation? The evaluations of e_3 , e_4 and e_5 based upon x , are no longer relevant while $T[e_1]$ and $T[e_2]$ are still valid and useful. To keep track of such partial evaluations, PLANNER needs to keep track of the partial evaluations that are made up to each choice point. Here we would store $\min[T[e_1]$, $T[e_2]]$ with the choices for x at e_3 . If any expression $e_i (i>3)$ THFAILS back to e_3 we can choose a new value for x and have a valid partial evaluation for e_0 . The value of the THAND should be $\min T[e_i]$

for the first set of substitutions which satisfy all the e_i . Otherwise, it should be $\min_i T[e_i]$ for the last set of which are tried and THFAIL.

In a THPROG with loops, the PLANNER executive allows backup to proceed properly "through the loops." If partial-evaluations are stored at each choice point, the evaluation of the truth-value of a THPROG can proceed much as in a THAND.

Suppose we add the following assertions to the data base we have just developed:

e_{13}: (FIRM APPLE75) $T[e_{13}] = .8$

e_{14}: (FIRM APPLE1) $T[e_{14}] = .95$

Now suppose we ask for a firm, ripe apple with credi- bilities = .8 (THAND (THGOAL (RIPE X) .7)(THGOAL (FIRM X).8) (THGOAL (APPLE X)1)) . Then if the PLANNER executive orders its choices for (RIPE X) by truth- value, it will again attempt e_9 e_8 e_7 with X=BX75, APPLE75,APPLE1 in that order. PLANNER cannot satisfy (THGOAL (FIRM BX75)) , backtracks, and selects e_8 to match (THGOAL RIPE X) .7) . e_8 satisfies each THGOAL in the THAND with truth values .9 , .8 and 1 respectively. X=APPLE75 with truth-values of .8 is FUZZY PLANNER's response. Had PLANNER chosen APPLE1 instead of APPLE75, it would have THSUCCEEDED through the THGOALS with truth-values of .85, .95 and 1. The resultant truth-value of the THAND is .85. In an exhaustive search, APPLE1 would be preferred to APPLE75. Here we are willing to accept a suboptimal solution that satisfies our minimal truth-value since it takes less search to find than in seeking the truly best choice.

These four considerations, thresholding, ordering the candidates for backtracking by truth-value partial evaluation and satisficing apply to the execution of other PLANNER statements such as THFIND, THPROG and THOR.

VI. Control of Fuzzy-PLANNER Expressions

The primary control mecha-isms within PLANNER programs is the THCOND while THCONSE theorems are chosen to select programs that may satisfy a THGOAL. Both THCOND and THCONSE can pass control based upon the truth-values of Fuzzy-PLANNER expressions. Consider the PLANNER expression

$$\text{(THCOND } (p_1\ e_{11}\ e_{12}\ \cdots\ e_{1n} \\ \vdots \\ (p_n\ e_{n1}\ \cdots\cdots\ e_{nn}))\ .$$

When the first p_i THSUCCEEDs, the corresponding $e_{i1} \cdots e_{in}$ are evaluated.

Analogously, we would like to branch when we find the first p_i such that $T[p_i] \geq \alpha$. Consider the following program:

```
(THCOND (((THPROG (X) (THGOAL (RIPE X) .8)
                      (THGOAL (APPLE X) 1))
                                     (GIVE X "JOHN"))
        ((THPROG (X) (THGOAL (CHEESE X) 1)
                     (GREAT (WEIGHT X) 4 oz))
                                     (GIVE X "PETER"))
        (T (TELL "JOHN" "NO SNACKS"))
```

If a ripe apple $(\tau \geq .8)$ is found, then it will be given to John. If a piece of cheese is found, it will be given to Peter. Otherwise John will be told "no snacks." In executing this program, Fuzzy-PLANNER will satisfice with minimal search.

The truth-value necessary to satisfy a given THGOAL can be used to cut off ineffective search. Consider:

```
TH3:  (THCONSE (XY) .95 (TASTY X)
                (THGOAL (CREDIBLE Y))
                (THGOAL (SAYS Y (TASTY X)))
```

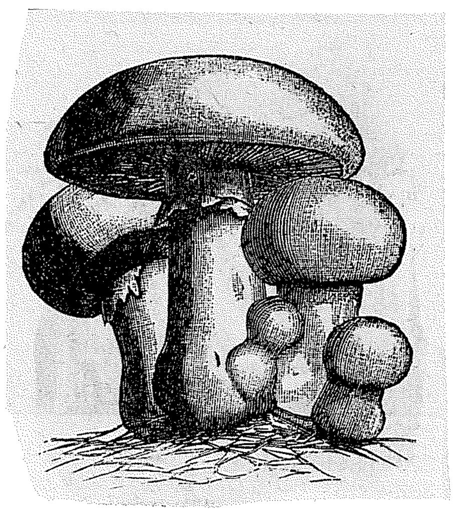
Suppose we wish a very tasty apple:

```
(THPROG (Z) (THGOAL (TASTY Z) .8)
            (THGOAL (APPLE Z)  1)
```

Furthermore let $*[x;y] = \min[x,y]$.

TH1, TH2 and TH3 are all candidate theorems to help us find a tasty apple. They will be ordered by their truth-values TH3, TH2, TH1 and attempted in order. If TH3 is invoked, we know that T[(THGOAL (CREDIBLE Y))] must be $\geq .8$. If it is not, then T[(THGOAL (TASTY Z))] $< .8$ and we cannot get an adequately tasty apple. In this example, .8 (denoted by β) provides a lower bound for the truth-value of any top-level expression in TH1, TH2 or TH3. More generally, suppose we are seeking to satisfy a THGOAL with truth-value $\geq \tau$ by invoking a particular THCONSE. Then the truth-value of each top-level expression in that THCONSE must exceed τ . If the truth-value of any top-level expression falls below τ , then we can generate a failure at that point since no further evaluation will return a truth-value $> \tau$ for that assignment of constants. These observations hold precisely for $*[x_i y] = \min[x_i y]$. For other $*$ functions, the β cutoff value will still exist, but need not be equal to the τ of the desired THGOAL. In Goguen's logic, $\beta = \frac{\tau}{\tau_0}$ where τ_0 is the truth-value specified in the desired THGOAL and τ_0 is the truth-value of the THCONSE.

Fuzzy PLANNER is upwards compatible with the standard PLANNER. If no truth-values are specified in THGOAL statements, then α is assumed. If at least some assertions are assigned fuzzy truth-values, then a Fuzzy PLANNER search will be conducted even though the program does not appear to specify fuzzy criteria.



VII. Open Issues

The preceding discussion has included the essential themes of Fuzzy PLANNER. Some details, such as the treatment of Fuzzy THANTE have been excluded for brevity. Certain issues are still unresolved (e.g., same syntax). Other new issues have been opened by this research.

A. Fuzzy PLANNER syntax: Occasionally one wishes to neglect an expression in computing truth-values. For example, (THSETQ X (CDR X)) is uninterpreted and should not interfere with the computation of Fuzzy truth. Classical PLANNER offers few means to iterate through a set. One mechanism entails using a THAMONG which selects elements from a set, followed by some operation on each element. This sequence is terminated by a THFAIL which backs up to the last choice point (THAMONG) to select a new element. This cumbersome PLANNER device uses a THFAIL for control purposes. But one does not want T[THFAIL] = 0 to be included in the evaluation of truth-values in such a process. Generally, we need a neat syntax for a Fuzzy PLANNER programmer to mask out selected expressions from being included in the evaluation of truth-values.

B. Computing Fuzzy Predicates: The existent literature on fuzzy sets (Zadeh [1965]) and fuzzy logic simply assume that one has a fuzzy membership function. The epistomological issue (how one gets such a function) is neglected. Now the epistomological issue can no longer be neglected. After all, if we wish a robot to bring us the "sturdiest" chair, we have to specify the computation "sturdiness" for chairs. In addition, the computation of fuzzy predicates may well depend upon their arguments. After all, a "large chair" is (typically) smaller than a "small room."

C. <u>Computing Truth Values for THCONSE Theorems:</u> Implication in Lee's logic and Goguen's logic have different algorithms. If we wish to know $T[p[x] \Rightarrow q[x]]$ in Lee's logic, we list all $q[a_j]$ such that $T[p[a_j]] \geq \alpha$. Then,

(1) $T[p[x] \Rightarrow q[x]] = \min_j \{T[q[a_j]] | T[p[a_j]] \geq \alpha\}$

In Goguen's logic:

(2) $T[p[x] \Rightarrow q[x]] = \min_j \{\frac{T[q[a_j]}{T[p[a_j]} \mid T[p[a_j] \geq T[q[a_j]]\}$

Both logics use min for $\forall$. Thus, one a_k such that $T[q[a_k]]$ is very low with respect to the other $T[q[a_j]]$ will lead to a low value for $T[p[x] \Rightarrow q[x]$. Integrating over $T[q[a_j]$ to obtain an "averaged" value violates our interpretation of $\forall$ as a condensed conjunction unless our truth-value for conjunction is changed. Then we are no longer on a lattice. This is now an open issue.

D. <u>Applicability of Different Fuzzy Logic:</u> The preceding discussion has described two distinctly different fuzzy logics. Truth-values of inferences are bounded below by the weakest piece of evidence in the derivation in Lee's logic. In contrast, Goguen's inferences decrease in truth value as the length of the deduction increases. Each is an interesting candidate logic. Which is more applicable to different situations, e.g., robot planning, medical diagnosis? We are now in a position where it makes sense to find out.

BIBLIOGRAPHY

Goguen, J. A. [1968] "The Logic of Inexact Concepts", _Synthese_ 19:325-373.

Green, C. and Raphael, B. [1969] "The Use of Theorem Proving Techniques in Question-Answering Systems" _Proc. ACM 23rd Natl. Conf._:169-181.

Hewitt, C. [1971] "Procedural Embedding of Knowledge in PLANNER" _Proc. 2nd International Joint Conference on Artificial Intelligence_:167-184.

Lee, R. C. T. [1972] "Fuzzy Logic and the Resolution Principle" _JACM_ 19(1):109-119.

Mellinkoff, S. [1959] _Differential Diagnosis of Abdominal Pain_ McGraw-Hill.

Raphael, B. [1970] "Robot Research at Stanford Research Institute" SRI Artificial Intelligence Center Technical Note 64.

Rosser, J. B. and Turquette, A. R. [1952] _Many-valued Logics_ North-Holland Publishing Co.

Winograd, T. [1972] _Understanding Natural Language_ Academic Press.

Zadeh, L. [1965] "Fuzzy Sets" _Information and Control_ 8(6):338-353.