# DILEMMA INFERENCE ON SNePS SEMANTIC NETWORK SYSTEM

Han Yong You

# DILEMMA INFERENCE ON SNePS SEMANTIC NETWORK SYSTEM

## I. INTRODUCTION

This paper presents a package of LISP functions collectively called DILEMMA defined within the frame of MULTI [McKay & Shapiro (1980)], adding an extra bit to the existing capability of inference process possessed by SNePS semantic network processing system [Shapiro (1979a)].

Current inference mechanisms implemented on virtulally any semantic network systems are driven by essentially two basic operations: <u>matching</u> and <u>data extension</u>. Matching operation is to discover instances out of the database which will prove or disprove the queried theorem, while data extension operation is to expand the database through legitimate applications of inference rules upon the data available at the given moment in order to feed the matching operation with more data. We can thus view that inferencing is a series of computations in which these two basic operations are taking place repeatedly until <u>either</u> the theorem is proved or disproved <u>or</u> no more data extension is possible with the theorem yet undecided when the available data is exhausted.

So far, a disjunctively asserted data items like

(1) P v Q

are regarded useless as data items in the database even if P and Q are individually a molecular constant. For instance, when one knows that "Tom is either a first-year graduate student or a senior undergraduate student of this department.", the current situation is such that that knowledge is not usable for answering such a question like "Is Tom a student of this department?".

The goal of this research is to add an extra set of control mechanisms to the present SNePS inference system so that it can now utilize such disjunctively asserted data as data item, not only as a rule if such data shows a possiblity of proving or disproving the queried theorem in one way or another. This technic of inference, which is traditionally called "dilemma inference", shows itself, in fact, not so rare in an actual human reasoning.

This paper has two folds. The first section is devoted to an examination of the anatomy of dilemma inference in terms of formal logic, and the latter section presents the new package as a program object describing its major components with some discussions on a few technical issues related to this implementation.

## II. THE LOGICAL ASPECT OF DILEMMA INFERENCE

It looks like to me that the intrinsic mechanism of inference processes implemented on virtually all systems including SNePS is the so-called "disjunctive syllogism" which is depicted by (2):

```
(2)      # P  v  Q.
         # ~P.
       ______________
         + Q.          ,
```

where the meta-symbols have the following interpretations:

```
        # X.:  "It is asserted that X is logically valid.";
        + X.:  "It is derivable that X is logically valid.";

        .  :  (indicates the range of the last meta-symbol
                # or +);
       ____:  (indicates whatever above this is in database).
```

If we introduce another literal P' which is logically complementary to P (i.e., P' = ~P), then (2) can be rewritten into (3):

```
(3)      # ~P' v  Q.
         # P'.
       ______________
         + Q.          ,
```

which is the canonical notation to express both syllogisms of <u>modus ponens</u> and <u>modus tollens</u> which are each conventionally represented by (4) and (5), respectively.

```
(4)      # P' =>  Q.
```

$\# \; P'.$

---

$+ \; Q.$

(5) $\# \; P' \Rightarrow Q.$

$\# \; {\sim}Q.$

---

$+ \; {\sim}P' \; ( = P).$

But, the constraint that the number of the terms appearing in the major premise has to be two seems uselessly too strong. And similarly, the number of expressions in the minor premise does not have to be one either (here, I will exclusively mean by "terms" the top-level disjuncts within a disjunctive proposition, and by "expressions" the top-level conjuncts within a conjunctive proposition). Thus, a more generalized disjunctive syllogism is formalized as (6):

(6) $\# \; P_1 \vee P_2 \vee \ldots\ldots \vee P_n.$

$\# \; {\sim}Q_1 \wedge {\sim}Q_2 \wedge \ldots\ldots \wedge {\sim}Q_m.$

---

$+ \; R_1 \vee R_2 \vee \ldots\ldots \vee R_{n-m}.$

where $Q_i \in A = \{P_j\}$,

$R_k \in A - \{Q_i\}$,

$m < n, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n, \quad 1 \leq k \leq n - m.$

It is immediatly recognized that representing (6) in an implication format will be extremely cumbersome and wasty because there exist roughly $2^n$ number of different ways of grouping the terms appearing in the major premise into two partitioned sets (one for the antecedent and another for the consequent), and every different partition will require each unique implication of its own. A bi-directional use of one implication can reduce that explosive number by one half only.

We now clearly see that the operation of data extension in a deductive inference process conventionally implemented on most systems using the material implication is a proper subset of this generalized disjunctive syllogism. The underlying idea of a deductive inference is that, if we design a database in a certain way abiding some constraint(s), then, when a part of the data turns out to have a certain logical value, may we possibly predict the logical value of some part of the rest of the data. Two different constraints very likely useful in constructing a database come to our mind. One constraint is to keep the overall logical value of the whole database being <u>inconsistant</u>, while another is to keep it being <u>valid</u>. A system which maintains the overall logical value to be valid is called a "truth-maintenance system" (TMS). And a system which maintains the overall logical value to be inconsistant may be analogously called an "inconsistant-maintenance system" (IMS). A TMS system will crash with even a single invalid expression connected by a logical AND on the top level. Similarly, an IMS system will crash with even a single consistant term connected by a logical OR on the top level. Therefore, if the whole database is going to be decomposed into a number of independent partitions connected all together by an appropriate logical connective on the top level (AND for a TMS system and OR for an IMS system), then it becomes essential to assure that the logical value of each partitioned data component is also valid for a TMS system and inconsistant for an IMS system. If a TMS system consisted of a top-level OR, then any one valid component will make the rest of the whole database uninteresting, and thus we can no longer find independence relation among the top-level components. A similar claim is also applicable to an IMS system as to top-level AND connection. It becomes thus clear why an appropriate top-level logical connective to integrate the whole database must be chosen so as to make the database non-trivially useful depending on the orientation of the system's truth-value maintenance. Thus, it becomes clear that the top-level data structure of a TMS system is a conjunction of disjunctions, while the top-level data structure of an IMS would be, if it be ever tried, a disjunction of conjunctions. And, we further see that, in a TMS system, disjunctive syllogism is the fundamental tool for inference.

Then, we may easily imagine another type of tool for inference, namely, conjunctive syllogism on an IMS system which is depicted by (7).

(7)

$$\begin{array}{l} \mid\# \; P_1 \;\hat{}\; P_2 \;\hat{}\; \ldots\ldots \;\hat{}\; P_n. \\ \mid\# \; {\sim}Q_1 \;\text{v}\; {\sim}Q_2 \;\text{v}\; \ldots\ldots \;\text{v}\; {\sim}Q_m. \\ \hline \mid+ \; R_1 \;\hat{}\; R_2 \;\hat{}\; \ldots\ldots \;\hat{}\; R_{n-m}. \end{array}$$

where the meta-symbols have the following meanings:

¦# X.: "It is asserted that X is inconsistant.";

¦+ X.: "It is derivable that X is inconsistant.";

and all the others rest the same as in (6).

However, if we pretend that "consistant" and "valid" are equivalent, and "inconsistant" and "invalid" are also equivalent (as is in a monotonic logic), then, by replacing each of the meta-symbols |# and |+ by a # and +, respectively, together with a negation associated to every of them, we can equivalently transform (7) into (8):

$$
\begin{array}{ll}
(8) & \# \sim[ P_1 \wedge P_2 \wedge \ldots\ldots \wedge P_n ]. \\
 & \# \sim[ \sim Q_1 \vee \sim Q_2 \vee \ldots\ldots \vee \sim Q_m ]. \\
\hline
 & + \sim [ R_1 \wedge R_2 \wedge \ldots\ldots \wedge R_{n-m} ].
\end{array}
$$

where all the conditions rest the same as in (7).

Using de Morgan's law, (8) is rewritten into (9):

$$
\begin{array}{ll}
(9) & \# \sim P_1 \vee \sim P_2 \vee \ldots\ldots \vee \sim P_n. \\
 & \# Q_1 \wedge Q_2 \wedge \ldots\ldots \wedge Q_m. \\
\hline
 & + \sim R_1 \vee \sim R_2 \vee \ldots\ldots \vee \sim R_{n-m}.
\end{array}
$$

where all the conditions rest the same as in (8).

If we introduce new atoms, which are each a logically complement of a negated atom in the set, and are each named by adding a prime (diacritic) to the name of its complementary original atom, then we can rewrite all the propositions using these newly introduced atoms only, getting (10) from (9).

$$
\begin{array}{ll}
(10) & \# P'_1 \vee P'_2 \vee \ldots\ldots \vee P'_n. \\
 & \# \sim Q'_1 \wedge \sim Q'_2 \wedge \ldots\ldots \wedge \sim Q'_m. \\
\hline
 & + R'_1 \vee R'_2 \vee \ldots\ldots \vee R'_{n-m}.
\end{array}
$$

where, if the complementary relation

is appropriately considered, all the conditions rest analogously the same as in (9).

We see that (10) is syntactically identical to (6), which is a formalization of a disjunctive syllogism on a TMS system. Hence, we now realize that disjunctive syllogisms and conjunctive syllogisms are isomorphic to one another in a monotonic logic. To put this in another way, the duality of a logical model enables us to construct an equally healthy model (as much healthy as the original) by exclusively exchanging all conjunctions with disjunctions and vice versa, all assertions with negations and vice versa, and all interpretation of "valid" with "inconsistant" and vice versa [Kleene (1967:22)]. Thus so far, we have shown that disjunctive syllogism may be adopted as a cannonical notation for all inference mechanism, and that any significant development of inference technic will probably be captured by this cannonical representation inference mechanism.

But, I would like to point out here that it is not our original purpose just to claim that the disjunctive syllogism-like notation can be a cannonical representation of any inference mechanisms. We rather want to look into the behavior of conventional inference mechanism with the aid of this cannonical representation, and attempt to find out any possible factors that we may be able to improve.

I think there are at least two heels of Achiles embedded in most of the currently running inference systems. One is that it is always assumed that the negation of "valid" is necessarily equivalent to "inconsistant", and vice versa, so that a system hardly knows how it can be modest by saying "Sorry, I don't really know. I can't really say anything.". Thus, one possible revolt can be not granting the monotonicity of the interpretation of a system's logical value. How to construct a system that computes deductive inferences on a non-monotonic logic appears to be an interesting worthy problem.

But, granting the monotonicity of logic, and thus pretending that the frame of disjunctive syllogism is the basic tool for all deductive inferences, another taboo which was not challenged by current systems appears to be the assumption that a minor premise is necessarily a single term. What kind of inference can be performed if a minor premise turns out to be a disjunction of syllogisms, the conclusion of a unit syllogism is supposed to be a disjunction of more than one term ?

Let $D_1$ and $D_2$ be two independently valid disjunctive syllogisms like the one whose schemata was defined in (6), and M(D), m(D), and C(D) respectively refer to major premise, minor premise, and conclusion of a disjunctive syllogism D. And let us define an operation * that applies on two disjunctive syllogisms such that $D_1 * D_2$ represents a new disjunctive syllogism which is a composition of $D_1$ and $D_2$ as described in (11).

$$
(11) \qquad D_1 * D_2 \quad <=> \quad \begin{array}{l} \#\ \ M(D_1) \ \ \wedge \ \ M(D_2). \\ \#\ \ m(D_1) \ \ \vee \ \ m(D_2). \\ \hline +\ \ C(D_1) \ \ \vee \ \ C(D_2) \qquad . \end{array}
$$

A careful calculation will prove that $D_1 * D_2$ is also valid if $D_1$ and $D_2$ are valid individually. Further more, it can be also shown that * operation is associative and commutative and can be applied abitrarily many times whose result is eventually given as shown in (12).

(12)

$$
\begin{array}{l}
\# \ X_1 \ \wedge \ X_2 \ \wedge \ \ldots. \ \wedge \ X_m. \\
\# \ Y_1 \ v \ Y_2 \ v \ \ldots. \ v \ Y_m. \\
\hline
+ \ Z_1 \ v \ Z_2 \ v \ \ldots. \ v \ Z_m. \quad ,
\end{array}
$$

where $X_i = P_{i1} \ v \ P_{i2} \ v \ \ldots. \ v \ P_{im<i>}$ ,

$Y_j = \sim P_{j1} \wedge \sim P_{j2} \wedge \ldots. \wedge \sim P_{jn<j>}$ ,

$Z_j = Q_{j1} \ v \ Q_{j2} \ v \ \ldots. \ v \ Q_{j(m<j>-n<j>)}$ ,

where $Q_{jt}$ e $A_j$ ,

$t = m<j> - n<j>$,

$A_j = \{P_{j1}, P_{j2}, \ldots. , P_{jm<j>}\} - B_j$ ,

$B_j = \{P_{jk}\}$ ,

$1 =< k =< n<j>$ ,

$1 =< i,j =< m$ , $\quad n<j> =< m<j>$ ,

A simple example of this expanded disjunctive syllogism is given by (13).

(13)

$$
\begin{array}{l}
\# \ (P_{11} \ v \ P_{13}) \ \wedge \ (P_{21} \ v \ P_{22}). \\
\# \ \sim P_{11} \ v \ \sim P_{21}. \\
\hline
+ \ P_{12} \ v \ P_{22}. \quad .
\end{array}
$$

Here in this example, we can easyily see that the major premise has two expressions and both the minor premise and the conclusion also have two terms, and that the terms there are each one-to-one associated with each expression in the major premise.

An eyeball examination tells us that (13) is nothing else but the canonical notation of a simplex dilemma as shown by (14).

(14) # $(P'_{11} \Rightarrow P_{12}) \wedge (P'_{21} \Rightarrow P_{22})$.

# $P'_{11} \vee P'_{21}$.

---

\+ $P_{12} \vee P_{22}$. ,

where $P'_{ij} = \sim P_{ij}$.

Since old Greek sophists paid them a great deal of attention, dilemmas (<u>di</u>- for <u>two</u> and <u>lemma</u> for <u>assumption</u> or <u>proposition</u>) have been long regarded as one of the most powerful tools for a debate and has attracted much of logicians' interest. To describe it verbally, the format of a dilemma is that the major premise is a conjunction of arbitrary number of implications, and the minor premise is either a disjunction of every antecedent or a disjunction of the negation of every consequent. Of course, the conclusion is either a disjunction of every consequent or a disjunction of the negation of every antecedent. Traditionally, the terms listed in the monor premise were called "horns" of the dilemma, and completing an exhaustive list of all possible horns were known as an ultimate technic of developing strong arguments using a dilemma. Two well-known standard technics of attacking a dilemma argument were known to be either to seek any propositions not listed in the major premise that may produce a couter-argument unfavorable to the dilemma argument (-- "to take horns"), or to seek some missing horns that may provide a counter-argument unfavorable to the dilemma argument (-- to evade between horns). Depending on whether a dilemma is run in modus ponens or nodus tollens, it was called "constructive" or "destructive", respectively. And further, whether the conclusion of dilemma was merged into one single term or not, the distinction between a "simplex" and a "complex" dilemma was also made.

To my knowledge, no system has attempted to implement this most general schemata of disjunctive syllogism which seems to me to be able to accomodate any kind of inference mechanisms including simple disjunctive syllogisms.

## III. IMPLEMENTATION OF DILEMMA INFERENCE

### 1. Motivation

We pointed out already that dilemma is a special type of inference mechanism which may be viewed as a set of parallelly processed simplex disjunctive syllogisms among which a special inter-relationship * operation holds as defined. In most situation, however, we would not really appreciate this too complex process just in order to obtain a set of disjunctive conclusion with which one can hardly do anything. But, if in some situation, the conclusion happens to be merged into one single term (which is the case of a simplex dilemma), then we find that having the dilemma inference machanism available will sometimes enable a system to derive a conclusion which one would normally want but could not get. Two examples of typical benefits we can get through this dilemma inference are shown by (15) and by (16).

(15)

$$\# \; (\forall x)(Animal(x) \Rightarrow Breathe(x))$$
$$\# \; (\forall y)(Plant(x) \Rightarrow Breathe(x))$$
$$\# \; (\forall z)(Alive(z) \Rightarrow Animal(z) \vee Plant(z))$$

---

$$+ \; (\forall w)(Alive(w) \Rightarrow Breathe(w))$$

(16)

$$\# \; On(block1, block2).$$
$$\# \; On(block2, block3).$$
$$\# \; Red(block1).$$
$$\# \; Red(block3).$$
$$\# \; Red(block2) \vee Blue(block3).$$

---

$$+ \; (\exists x \exists y)(Red(x) \wedge Blue(y) \wedge On(x,y)).$$

where all the predicates in (15) and (16) are dummy symbols for some appropriate n-ary predicates (But, readers are not discouraged to make a possible association for each of these with any tangible interpretation they may like to imagine).

The example (15) shows that dilemma inference can be used to derive new inference rules ("patterns" in a network term) which is not supprising since dilemma inference is a meta-mechanism which is ablout rule schematas. The example shown by (16), which is one of the famous problems in AI community known as "three-box problem" sheds another interesting point. The conclusion in the process (16) asserts that there exist some x and some y such that the formula in the conclusion has an actual instanciation. Usually, a formula with unbound variable or a disjunctively asserted assertion is worthless for a final use. However, we notice that the conclusion drawn in (16) is already usable enough to answer a query that simply asks whether or not there exists such an instance not necessarily demanding to learn the exact binding situation. However, there may be someone who may wonder if (16), with a disjunctive assertion still in the conclusion, is within the frame of a simplex constructive dilemma which requires its conclusion to be a merged single term. We believe that it is so, but in one level higher order, though. To prove it, let us define P and E such that

(17) $P(x,y) \Leftrightarrow On(x,y) \wedge Red(x) \wedge Blue(y)$ , and

$E(q) \Leftrightarrow (\exists x \exists y)(q(x,y))$.

Certainly, E(P) will have an interpretation that says "there is at least one instance of P, a binary predicate.". With this definition, we can then rewrite (16) into (18).

(18) # P(block1, block2) => E(P).

\# P(block2, block3) => E(P).

\# P(block1, block2) v P(block2, block3).

---

\+ E(P).

Notice that (18) conforms precisely the schemata of simplex constructive dilemma, but where a predicate P is used as an argument of a new higher order predicate E.

## 2. Solution on SNePS

First of all, it will be a good procedure to describe very briefly the present status and the nature of SNePS semantic network procesing system being maintained at the Department of Computer Science at State University of New York at Buffalo. SNePS can hold semantic information represented in the net as a set of associated network, and can perform backward inference for a specific theorem queried while a limited depth of forward inference may be also done. Inference mechanisms being used includes resource limited inference and ANDOR computation [Shapiro (1979b)] on top of the standard material implication. These various inference mechanisms are run in multi-processing mode in which a maximum data sharing becomes possible.

One of the conceivable ways of approaching to the implementation of dilemma inference to SNePS could be a utilization of already available inference mechanisms with a little addition of extra control. Among others, AND-IMPLICATION mechanism may appears to be very plausible. It proves a given theorem only when all the antecedents are proved. Considering that a dilemma inference proves a theorem only when the theorem is proved with every horn, we can realize that there is an AND-IMPLICATION's nature in a dilemma inference. From this, we can obviously derive a rule (19) which says:

(19) $( H \underset{\wedge}{\Rightarrow} T ) \Rightarrow T.$ ,

where H is a set of all horns, and T is a queried theorem. This rule represented by (19) is a tautology no matter what T happens to be. Thus, adding this rule to the database does not cause any problem as far as the truth-maintaining business is concerned. Due to this rule, then we may initiate AND-IMPLICATION mechanism, and wait until this process returns an answer? If AND-IMPLICATION proves T, then by the rule (19), we do prove T. But, a close examination shows us that this is not a solution by any means. Because the disjunctively asserted data will not be utilized by the AND-IMPLICATION process, the rule has no significance at all as far as dilemma inference is concerned. It is just like your saying that if you prove it then it is proved.

Discussing AND-OR tree problem solving technic, Nilsson (1980) illustrated that the three-box problem is a very peculiar kind of a graph-like AND tree problem where there are two roots at both of which the control paths should communicate. One root is the node representing the given goal while another is the node that provides the disjunctive instance leading to a solution via "reasoning-by-cases" strategy. It is an AND tree because every branch of the disjunctive instance must succeed in proving the theorem in order to prove the theorem. In an ordinary AND-OR tree problem solving, the goal node creates descendent AND or OR nodes, and lets them run to see who brings back which anwer. However, in this peculiar situation that he called the case of reasoning-by-cases strategy, an AND tree is created from the node that represents the disjunctive assertion, and the goal node can make a decion when the AND tree created by someone else reports to him that every branch has been successfully terminated. Thus, he stated that the key to the solution of this strategy is somehow to make it possible for the two critical nodes to communicate. This matter could be resolved pretty easily on SNePS running on MULTI. When a top level inference mechanism finds a disjunctive assertion which seemingly has a potentiality of proving the theorem in a reasoning-by-cases fashion, the top level inference driver sets up a specialist who will take care of the dilemma inference case. Upon being set up, the specialist creates each case handler for every horn resolution, and instructs each of them to report to himself, and then waits until all horns sends him a message of success.

The real harder problem of a dilemma inference rather lies in the way how each of the horn attackers can resolves his own problem. Each horn attacker is of course expected to call for the help of inference specialist forming a daisy-chain recursion in order to solve his horn resolution problem with his particular horn assumption. In a data-sharing system like present SNePS running on MULTI, a horn assumption may not safely be added to the database as if it were real to everybody since it will mess up all other innocent processes sharing the data.

Thus, the way how the horn assumption is handled for each horn resolution seems to be the real core of the solution to dilemma inference (or reasoning-by-cases).

The solution which is adopted by this implimentation is to let the horn attacker reshape his own theorem derived from the original theorem that the dilemma process boss has been asked about. Each horn attacker is asked to prove the grand theoren with one's own horn assumption taken granted. A different assumption leaves a different subtheorem to be proved. For instance, if the horn assumption were the original theorem itself, then that particular horn attacker does not have to do anything. His duty is none from the very beginning. All he has to do is to report that with his horn assumption, the theorem is proved. Reshaping of theorem clearly does not affect the database, and makes a change onto the theorem ultimately to be proved, thus raising the possibility of the theorem's being proved.

## 3. Program description

On top of number of supporting lisp functions, this program package consists of four newly defined MULTI processes and two pre-existing processes slightly modified so that this package can be coupled to the present SNePS inference package. Each process is described as follows:

### INFER

This is a pre-existing process which cranks the main piston of inference machinary. This was modified so that it collects available and relevant horn sets for a given theorem to be proved. Dilemma inference is triggered only when the global switch <DILEMMA> is set to T, which is the default value at the top-most top level. If any horn sets are collected, INFER creates D-INFER and D-ANSCAT with the help of a lisp function <dilemma-infer>.

### TOPMOST-TOPINF

This pre-existing process was modified so that it can receive disjunctive answers obtained throuhgh dilemma inference via a different process registor. The reason it does not use the normal message channel is in order not to cause the disjunctive answer to be permenantly built as other type of answers are. The deep reason why a disjunctive answer must not be permenantly built is that a dilemma inference cannot determine the number value for MAX arc. The routine assigns the maximum value for MAX for the sake of the largest generality, but certainly the system does not want it to serve as inference rules for any further inference.

D-INFER and D-ANSCAT

D-INFER creates appropriately many horn attackers ATTACK.HORN and one D-ANSCAT. ATTACK.HORN's are each given a horn to be disjunctively proved, and D-ANSCAT collects the answers coming from individual ATTAC.HORN's. When all horn aresuccessfully finished, D-ANSCAT send the anwer (disjunctively bound binging set) to whoever ordered the dilemma inference work.

ATTACK.HORN AND HORN-ANSCAT

ATTACK.HORN reshapes the local theorem related to the horn assumption, and initiate INFER recursively to resolve the horn. In this embedded call to INFER, the switch <dilemma> is set to NIL such that too much costing dilemma inference may not be triggered within the embedded level. HORN-ANSCAT catches answers from ATTACK.HORN's clients and sends it to D-ANSCAT.

# References

Kleene, S. C. (1967) Mathematical Logic. New York: Jojn Wiley.

McKay, D. P. & Shapiro, S. C. (1980) MULTI -- A LISP based multiprocessing system. Dept./Comp. Sci. SUNY at Buffalo. Technical Report #164.

Nilsson, N. J. (1980) Principles of Artificial Intelligence. Palo Alto: CA: Tioga.

Shapiro, S. C. (1979a) The SNePS semantic network processing system. In Findler ed., Associative Networks: Representation and Use of Knowledge by Computers: 179-203. New York: Academic Press.

Shapiro, S. C. (1979b) Using non-standard connectives and quantifiers for representing deduction rules in a semantic network. Presented at "Current Aspects of AI research", a seminar at Electrotechical Lab. Tokyo. Aug. 1979.

```
 ENTERING ECHO     DECEMBER  22, 1981     2:48 AM

 ?(INPUT DILMA)
(INFER GATHER.HORNS WORTH-DIL? D-HORN? NO-FREEVAR? NON-NILS ATOMOLECULE?
 DILEMMA-INFER DILEMMA D-INFER ALL-HORNS? ATTACK.HORN HORN-TORN-CQ UNTOR
N-CQ SELECT.HORNS SORT.HORNS PACK-IN PUTIN-BASKET SET.PREG EQUISET HORN-
ANSCAT D-ANSCAT RECORD.DANS DILEM-RPT DRAFT.D-ANS TOPMOST-TOPINF DEDUCE*
 D-SEND)
 ?(INSYS MEMO)
(SNEPS FILE LOADED)
 ?DILEMMA
T
 ?(SNEPS)
SNEPS
**(DESCRIBE (M1 M2 M3 M4 M5 M6  M7 M8 M9 M10 M11))
(M1 (OBJ (BLOCK2)) (SUBJ (BLOCK1)) (REL (ON)))
(M2 (OBJ (BLOCK3)) (SUBJ (BLOCK2)) (REL (ON)))
(M3 (OBJ (BLOCK12)) (SUBJ (BLOCK11)) (REL (ON)))
(M4 (COLOR (RED)) (P.OWN (BLOCK1)))
(M5 (COLOR (BLUE)) (P.OWN (BLOCK3)))
(M6 (COLOR (RED)) (P.OWN (BLOCK2)))
(M7 (COLOR (BLUE)) (P.OWN (BLOCK2)))
(M8 (ARG (M7 (COLOR (BLUE)) (P.OWN (BLOCK2)))
         (M6 (COLOR (RED)) (P.OWN (BLOCK2))))
    (MAX (1))
    (MIN (1)))
(M9 (COLOR (BLUE)) (P.OWN (BLOCK11)))
(M10 (COLOR (RED)) (P.OWN (BLOCK12)))
(M11)
(DUMPED)
629 MSECS

**(DEDUCE  P.OWN BLOCK2 COLOR %X)

 FOR A DILEMMA INFERENCE,
  WE KNOW
(M8 (ARG (M7 (COLOR (BLUE)) (P.OWN (BLOCK2)))
         (M6 (COLOR (RED)) (P.OWN (BLOCK2))))
    (MAX (1))
    (MIN (1)))

 HERE, WE INFER A DISJUNCTIVE ANSWER
(T87 (ARG (T86 (COLOR (BLUE)) (P.OWN (BLOCK2)))
          (T85 (COLOR (RED)) (P.OWN (BLOCK2))))
     (MIN (1))
     (MAX (2)))


NIL
882 MSECS

** (DEDUCE MIN 3 MAX 3 ARG (
*    (TBUILD SUBJ %X OBJ %Y REL ON)
*    (TBUILD P.OWN *X COLOR RED)
*    (TBUILD P.OWN *Y COLOR BLUE)))

 FOR A DILEMMA INFERENCE,
  WE KNOW
(M8 (ARG (M7 (COLOR (BLUE)) (P.OWN (BLOCK2)))
         (M6 (COLOR (RED)) (P.OWN (BLOCK2))))
    (MAX (1))
    (MIN (1)))
```

```
'A HORN TRIGGERS INFER TO PROVE  T113
 A HORN TRIGGERS INFER TO PROVE  T116
 HERE, WE INFER A DISJUNCTIVE ANSWER
(T153
 (:SVAR (Q100 (:VAR (T))) (Q101 (:VAR (T))))
 (ARG
  (T152
   (ARG
    (T151 (REL (ON))
          (OBJ (BLOCK2))
          (:SVAR (Q100 (:VAR (T))))
          (SUBJ (Q100 (:VAR (T)))))
    (T150 (COLOR (RED)) (:SVAR (Q100 (:VAR (T)))) (P.OWN (Q100 (:VAR (T)
    ))))
    (T149 (COLOR (BLUE)) (P.OWN (BLOCK2))))
   (:SVAR (Q100 (:VAR (T))))
   (MAX (3))
   (MIN (3)))
  (T148
   (:SVAR (Q101 (:VAR (T))))
   (ARG
    (T147 (REL (ON))
          (:SVAR (Q101 (:VAR (T))))
          (OBJ (Q101 (:VAR (T))))
          (SUBJ (BLOCK2)))
    (T146 (COLOR (RED)) (P.OWN (BLOCK2)))
    (T145 (COLOR (BLUE)) (:SVAR (Q101 (:VAR (T)))) (P.OWN (Q101 (:VAR (T
    ))))))
   (MAX (3))
   (MIN (3))))
 (MIN (1))
 (MAX (2)))


NIL
2259 MSECS

** (LISP)
END SNEPS
 ?(GRIND DILMA XREF ALPHA 90)



         DILMA                22 DECEMBER 1981  2.52.24

         CREATED:             14 DECEMBER 1981  21.31.24
         LAST MODIFIED:       22 DECEMBER 1981  2.15.28
         CHANGES MADE TO:     DEDUCE*  D-ANSCAT ATTACK.HORN      ALL-HORNS?        INFER


 **********************************************************************
 * [ALL-HORNS?] TESTS IF OR NOT EACH HORN INFERENCE HAS BEEN FINISHED
 *   WITH EVERY HORN IN THE HORN-SET BEING MADE AS AN ASSUMPTION.
 *   THE ASKED QUERY IS DISJUNCTIVELY ANSWERED ONLY WHEN THE INFERENCE FOR EVERY
 *   HORN AS AN ASSUMPTION HOLDS.
 *                                HYY -- 12/21/81


    ALL-HORNS?
VALUE
(LAMBDA (REG NHORN) (AND (EQ (LENGTH REG) NHORN) (NON-NILS (MAPCAR REG CDR))))

PLIST
NIL

 *****************************************************************
```

```
 * [ATOMOLECULE?] ASKS IF OR NOT A GIVEN NODE <NDE> DOMINATES ATOMIC NODES ONLY.
 *                                HYY -- 12/21/81


     ATOMOLECULE?
VALUE
(LAMBDA (NDE)
   (NON-NILS (MAPCAR (DOWNSET NDE)
                     (LAMBDA (ARGT)
                        (OR (NUMBERP (CDR ARGT)) (NULL (DOWNSET (CDR ARGT))))))
)

PLIST
NIL

 **********************************************************************
 * PROCESS [ATTACK.HORN] TAKES CARE OF THE INFERENCE OF THE GIVEN CQ WITH
 *   THE ASSUMPTION THAT THE HORN IS TRUE.  THUS, THE NEW CQ' TO BE PROVED
 *   IS "CQ - HORN" WITH THE BINGING PROVIDED BY THE HORN ASSUMPTION.
 *                                HYY -- 12/21/81


     ATTACK.HORN
VALUE
(LAMBDA (NAME: CLINK: CQ: BNDG:)
   (COND ((NULL CQ:) (SEND (LIST T) CLINK:))
         (T (PRIN3 <> " A HORN TRIGGERS INFER TO PROVE " * CQ:)
            (NEW-OLD-INFER CQ: BNDG: CLINK:))))

PLIST
(LREGS: (NAME: CLINK: CQ: BNDG:))

 *******************************************************************
 * [D-ANSCAT] PROCESS CATCHES THE ANSWERS FOR EVERY HORN INFERENCE, AND KEEPS
 *   CHECKING IF ALL HORNS PRODUCE EACH A DISJUNCTIVE ANSWER.  IF SO, THEN THIS
 *   PROCESS REPORTS THE ASNWER TO CLINK:.  <NHORN:> REMEMEMBERS THE NUMBER OF
 *   HORNS, <REG:> KEEPS ALL THE ANSWERS, <FLG:> SIGNAL GETS OFF AFTER ONE SET
 *   OF ANSWER IS SENT TO CLINK:.  BUT ALL THE ANSWER ARE CONTINUOUSLY DEPOSITED.
 *                                HYY -- 12/21/81


     D-ANSCAT
VALUE
(LAMBDA (NAME: CLINK: CQ: NHORN: REG: MBNDG: FLG: MSG:)
   (IF MSG:
       (MAPC MSG: (LAMBDA (MSG) (SETQ REG: (RECORD.DANS REG: MSG))))
       (SETQ MSG: NIL)
       (IF (AND FLG: (ALL-HORNS? REG: NHORN:))
           (COND
            ((EQ (REGFETCH CLINK: 'NAME:) 'TOPMOST-TOPINF)
             (D-SEND (DRAFT.D-ANS CQ: NHORN: REG:) CLINK:))
            (T (SEND (DRAFT.D-ANS CQ: NHORN: REG:) CLINK:)))
           (SETQ FLG: NIL)))
   (SET CURNT: (LIST NAME: CLINK: CQ: NHORN: REG: MBNDG: FLG: MSG:)))

PLIST
(LREGS: (NAME: CLINK: CQ: NHORN: REG: MBNDG: FLG: MSG:))

 ***************************************************************
 * [D-HORN?] ASKS IF OR NOT THE GIVEN NODE <DNE> IS A PPOTENTIALLY USEFUL
 *   HORN FOR A DILEMMA INFERENCE.  FOR THE TIME BEING, <DNE> IS REGARDED
 *   AS A CANDIDATE HORN SET ONLY WHEN IT IS A DISJUNCTIVELY ASSERTED CONTAINING
 *   NO VARIABLES.  FOR A FURTHER EXPANSION OF DILEMMA INFERENCE EVEN WITH RULE
 *   NODES, A RELAXATION OF THIS FUNCTION MUST BE APPROPRIATELY MADE.
 *                                HYY -- 12/21/81
```

```
     D-HORN?
VALUE
(LAMBDA (NDE)
   (PROG (MINI)
         (RETURN
          (AND (TOP? NDE)
               (NO-FREEVAR? NDE)
               (NON-NILS (MAPCAR (GET NDE 'ARG) ATOMOLECULE?))
               (SETQ MINI (CAR (GET NDE 'MIN)))
               (PLUSP (DIFF (LENGTH (GET NDE 'ARG)) MINI))))))

PLIST
NIL

 ***********************************************************************
 * [D-INFER] ITERATIVELY TRIES TO PROVE THE GIVEN CQ WITH EACH HORN BEING
 *   AN ASSUMPTION.
 *                                  HYY -- 12/21/81



     D-INFER
VALUE
(LAMBDA (NAME: CLINK: CQ: HORNSET: MBNDG:)
   (MAPC
    (CDR HORNSET:)
    (LAMBDA (HORN)
       (PROG (HP HC)
             (SETQ
              HP (NEW 'ATTACK.HORN
                      (SETQ HC (NEW 'HORN-ANSCAT CLINK: (ARGN HORN 2) NIL NIL T))
                      (HORN-TORN-CQ HORN CQ:)
                      (UNION-B (ARGN HORN 2) MBNDG:)))
             (REGSTORE CLINK: 'REG: (CONS (LIST HC) (REGFETCH CLINK: 'REG:)))
             (INITIATE HP)))))

PLIST
(LREGS: (NAME: CLINK: CQ: HORNSET: MBNDG:))

 ***********************************************************************
 * [D-SEND] IS A KLUDGE FOR SENDING AN ANSWER DERIVED THROUGH A D-INFERENCE
 *   TO [TOPMOST-TOPINF] PROCESS.  THE REASON FOR NOT USING NORMAL MESSAGE
 *   CHANNEL IS DESCRIBED IN [DEDUCE*] SECTION.  THIS MUST BE, THOUGH, ELIMINATED
 *   IN THE FUTURE BY CHANGING SOME CODE IN [TOPMOST-TOPINF] PROCESS, THROWING
 *   AWAY THE REGISTER <D-ANS:> EVENTUALLY.
 *                                  HYY -- 12/21/81



     D-SEND
VALUE
(LAMBDA (ANS BOSS)
   (SETQ ANS (CAR (ARGN ANS 2)))
   (REGSTORE BOSS 'D-ANS: ANS)
   (INITIATE BOSS))

PLIST
NIL

 ***********************************************************************
 * [DEDUCE*] MODIFIED BY HYY IN ORDER TO ADD ONE EXTRA REGISTOR TO THE PROCESS
 *   [TOPMOST-TOPINF].  THIS REGISTOR IS NEEDED TO GET AN ANSWER FROM THE
 *   DILEMMA INFERENCING PROCESS.  THE REASON WHY WE DO NOT USE THE NORMAL
 *   MESSAGE SENDING CHANNEL FOR THIS PURPOSE IS TO AVOID THE THEOREM PROVEN
 *   VIA D-INFERENCE BEING PERMENANTLY BUILT IN THE DATABASE.
 *                                  HYY -- 12/21/81
```

```
     DEDUCE*
VALUE
(LAMBDA (NUMFLD CQ)
   (PROG (TP RESULTS: INF %DILEMMA)
         (SETQ %DILEMMA DILEMMA)
         (PUT 'LASTINFER ':VAL NIL)
         (IF (NULL (FIRST-ATOM CQ)) (RETURN RESULTS:))
         (SETQ
          TP (NEW 'TOPMOST-TOPINF
                  NIL
                  (FIRST-ATOM CQ)
                  NIL
                  NIL
                  0
                  0
                  (IF (NUMBERP NUMFLD) NUMFLD)
                  (IF (NOT (ATOM NUMFLD)) (CAR NUMFLD))
                  (IF (NOT (ATOM NUMFLD)) (CADR NUMFLD))
                  NIL
                  NIL
                  (NEW 'I-MTR
                       NIL
                       (LIST (SETQ INF (NEW 'INFER NIL (FIRST-ATOM CQ) NIL NIL)))
                       NIL)
                  NIL))
         (REGSTORE INF 'CLINK: TP)
         (MULTIP (LIST (REGFETCH TP 'MTR:)))
         (PUT 'LASTINFER ':VAL (LIST TP))
         (TERPRI)
         (TERPRI)
         (RETURN RESULTS:)))

PLIST
NIL

 *************************************************************************
 * [DILEM-RPT] ISSUES A SNEPSUL USER READABLE MESSAGE FOR THE DILEMMA INFERENCE
 *    PROCESSING TAKEN.
 *                                       HYY -- 12/21/81


     DILEM-RPT
VALUE
(LAMBDA (HORNSET)
   (PRIN3 <> " FOR A DILEMMA INFERENCE,")
   (COND ((EQ (REGFETCH BOSS 'NAME:) 'TOPMOST-TOPINF) (PRIN3 <> "  WE KNOW" <>))
         (T (PRIN3 <> "  SINCE" <>)))
   (DESCRIBE (^ (CAR HORNSET))))

PLIST
NIL

 ************************************************************
 * <DUKILEMMA> IS A GLOBAL SWITCH FOR DILEMMA INFERENCING.  DEFAULT IS T.
 *                                       HYY -- 12/21/81


     DILEMMA
VALUE
T

PLIST
NIL

 ***************************************************************************
```

```
 * [DILEMMA-INFER] ITERATIVELY TRIES EVERY HORNSET TAKEN FROM THE HORNPILE,
 *   SETTING UP [D-ANSCAT] FOR ANSER CATCHER AND [D-INFER] FOR A DISJUNCTIVE
 *   REASONING.
 *                                      HYY -- 12/21/81


     DILEMMA-INFER
VALUE
(LAMBDA (BOSS CQ MBNDG HORNPILE)
   (SETQ %DILEMMA NIL)
   (REPEAT NIL
     WHILE HORNPILE
           (DILEM-RPT (CAR HORNPILE))
           (INITIATE (NEW 'D-INFER
                          (NEW 'D-ANSCAT
                               BOSS
                               CQ:
                               (LENGTH (CDAR HORNPILE))
                               NIL
                               MBNDG
                               T
                               NIL)
                          CQ
                          (CAR HORNPILE)
                          MBNDG))
           (SETQ HORNPILE (CDR HORNPILE))))

PLIST
NIL

 ***************************************************************************
 * [DRAFT.D-ANS] IS A KLUDGE FOR SENDING AN ANSWER DUE TO DILEMMA INFERENCE
 * [DRAFT.D-ANS] DRAFTS THE FINAL ANSWER TO BE SENT TO [TOPMOST-TOPINF] WHEN
 *   A DILEMMA INFERENCE BRINGS UP WITH A DISJUNCTIVE ANSWER.  NOTE THAT THE
 *   ANSWER IS A TEMPORARY NODE.
 *                                      HYY -- 12/21/81


     DRAFT.D-ANS
VALUE
(LAMBDA (CQ MAXI REG)
   (LIST CQ
         (APPLY TBUILD
                (LIST 'MAX
                      MAXI
                      'MIN
                      1
                      'ARG
                      (MAPCAR REG (LAMBDA (D-ANS) (NBUILD CQ (CADR D-ANS) TBUILD)))))))

PLIST
NIL




     EQUISET
VALUE
(LAMBDA (L1 L2)
   (AND (EQ (LENGTH L1) (LENGTH L2))
        (NON-NILS (MAPCAR L1 (LAMBDA (LL) (MEMB LL L2))))))

PLIST
NIL

 ***************************************************************************
 *
```

```
* [GATHER.HORNS] GATHERS WORTHWHILE HORN-SETS FOR A DILEMMA INFERENCE OF THE
*   GIVEN CQ.  A HORN-SET IS A DISJUNCTIVELY ASSERTED STATEMENT IN WHICH ANY
*   SUBSET OF THE GIVEN CQ IS INCLUEDED AS ONE OF ITS DISJUNTS.
*   RETURNS (MI  (MJ BJ TJ) (MK BK TK) ... ),
*   WHERE MI IS THE NODE OF THE HORNSET FOUND IN THE DATABASE,
*          (MJ .. TJ) IS A DATA SET FOR EACH HORN,
*          WHERE, MJ IS THE NODE OF HORN DISJUNCT,
*                 BJ IS THE BINGING SATISFYING THE HORN AS AN ASSUMPTION,
*                 TJ IS THE SUB-PART OF CQ WHICH IS PROVED BY THE HORN ASSUMPTION.
*                                      HYY -- 12/21/81


     GATHER.HORNS
VALUE
(LAMBDA (CQ BNDG)
   (PROG (HRN)
         (MAPC (OR (GET CQ 'ARG) (LIST CQ))
               (LAMBDA (X)
                  (MAPC (MATCHI X BNDG)
                        (LAMBDA (Y) (SETQ HRN (CONS (APPEND Y (LIST X)) HRN))))))
         (RETURN (SORT.HORNS (PUTIN-BASKET (SELECT.HORNS HRN))))))

PLIST
NIL

******************************************************************
* [HORN-ANSCAT] CATCHES ANSWERS FOR A HORN AND SEND THE FIRST ANSWER TO
*   [D-ANSCAT].  RIGHT NOW, THE ANSWER IS SENT JUST ONCE.  IN THE FUTURE,
*   SOMEONE MAY ATTEMPT TO LET IT SEND ALL ANSWERS BACK.  BUT NOTICE THAT
*   A TAXONOMICALLY EMBEDDED DISJUNCTION OF DISJUNCTIONS ARE REALLY MESSY.
*                                         HYY -- 12/21/81


     HORN-ANSCAT
VALUE
(LAMBDA (NAME: CLINK: BNDG: REG: MSG: FLG:)
   (IF MSG: (SETQ REG: (APPEND REG: (CDR MSG:)) MSG: NIL))
   (IF FLG: (SEND (LIST CURNT: (UNION-B BNDG: (CAR REG:))) CLINK:) (SETQ FLG: NIL))
   (SET CURNT: (LIST NAME: CLINK: BNDG: REG: MSG: FLG:)))

PLIST
(LREGS: (NAME: CLINK: BNDG: REG: MSG: FLG:))

************************************************************************
* [HORN-TORN-CQ] GENERATES A NEW CQ TO BE PROVED FROM A GIVEN CQ WITH THE
*   ASSUMPTION THAT THE HORN IS TRUE.
*   RETURNS A TEMPORARY NODE NEWLY BUILT.
*                                      HYY -- 12/21/81


     HORN-TORN-CQ
VALUE
(LAMBDA (HORN CQ)
   (PROG (LCQ)
         (RETURN
          (COND
           ((OR (EQ CQ (ARGN HORN 3)) (EQ (SETQ LCQ (LENGTH (GET CQ 'ARG))) 1)) NIL)
           (T (FIRST-ATOM (APPLY TBUILD
                                 (LIST 'MAX
                                       (SUB1 LCQ)
                                       'MIN
                                       (SUB1 LCQ)
                                       'ARG
                                       (UNTORN-CQ (ARGN HORN 3) (GET CQ 'ARG)))))))))))

PLIST
```

NIL

```
 ***********************************************************************
 * [INFER] MODIFIED BY HYY 12/21/81
 *   IN ORDER TO COLLECT HORNSETS INTO A HORNPILE, AND CALL DILEMMA INFERENCING
 *   ROUTINE IF <DILEMMA> IS SET AND <HORNPILE> IS NOT EMPTY.  FOR THE TIME BEING
 *   <DILEMMA> IS SET TO NIL AT ALL EMBEDDED INFERENCE LEVELS.  HOWEVER, THIS MAY
 *   BE LIFTED IN THE FUTURE IF AN ENOUGH MOTIVATION JUSTIFIES TO DO SO.
 *   SWITCHING OF <DILEMMA> IS DONE IN [DILEMMA-INFER].
 *                                     HYY -- 12/21/81


     INFER
VALUE
(LAMBDA (NAME: CLINK: CQ: BNDG: MSG:)
   (PROG (WD HORNPILE)
         (COND
          ((AND (NULL MSG:) (GET CQ: 'NAME:))
           (INITIATE (NEW 'EVAL-FN CLINK: CQ: BNDG: NIL NIL)))
          (T
           (PROG (M A D)
                 (SETQ M (OR MSG: (MATCHI CQ: BNDG:)) MSG: NIL)
                 (SETQ WD (WORTH-DIL? CQ:))
                 (IF (AND %DILEMMA WD) (SETQ HORNPILE (GATHER.HORNS CQ: BNDG:)))
                 (REPEAT NIL
                   WHILE M
                         (COND
                          ((TOP? (TNODE (CAR M)))
                           (SETQ A (CONS (LIST (TNODE (CAR M)) (SBIND (CAR M))) A))
                           (IF (EQ (REGFETCH CLINK: 'NAME:) 'TOPMOST-TOPINF)
                               (INF-RPT (SBIND (CAR M)) NIL NIL (LIST CQ:))))
                          ((AND (OR (GET (TNODE (CAR M)) (CONV 'CQ))
                                    (GET (TNODE (CAR M)) (CONV 'ARG))
                                    (GET (TNODE (CAR M)) (CONV 'DCQ)))
                                (NOT
                                 (MEMBER
                                  NIL
                                  (MAPCAR (TBIND (CAR M))
                                          (LAMBDA (BP)
                                             (OR (VAR (CDR BP))
                                                 (NULL (GET (CAR BP) 'EVB-))))))))
                           (SETQ D (CONS (CAR M) D))))
                         (SETQ M (CDR M)))
                 (IF (OR A
                         (AND (NULL (REGFETCH CLINK: 'CLINK:))
                              (EQP (REGFETCH CLINK: 'TOT:) 0)))
                     (SEND (CONS CQ: A) CLINK:))
                 (IF (AND D (OR (NULL A) (WH-Q (SVAR CQ:) BNDG:)))
                     (MAPC D
                           (LAMBDA (MTCHD)
                              (INITIATE
                               (NEW 'GO-UP
                                    (NEW 'SWITCH CLINK: CQ: (SBIND MTCHD) NIL)
                                    (TNODE MTCHD)
                                    (TBIND MTCHD)))))))
           (IF (AND %DILEMMA WD HORNPILE) (DILEMMA-INFER CLINK: CQ: BNDG: HORNPILE))
           (COND
            ((GET CQ: 'MAX)
             (INITIATE
              (NEW 'VANDOR
                   CLINK:
                   CQ:
                   (CAR (GET CQ: 'MIN))
                   (CAR (GET CQ: 'MAX))
                   (LENGTH (GET CQ: 'ARG))
                   0
```

```
                  (GET CQ: 'ARG)
                  BNDG:
                  NIL
                  NIL)))
            ((GET CQ: 'I-//-) (INITIATE (NEW 'VI-//- CLINK: CQ: BNDG: NIL NIL)))
            ((GET CQ: 'UNK) (INITIATE (NEW 'V-UNK CLINK: CQ: BNDG: NIL NIL NIL NIL))))))
         (SET CURNT: (LIST NAME: CLINK: CQ: BNDG: MSG:))))

PLIST
(LREGS: (NAME: CLINK: CQ: BNDG: MSG:))



     NO-FREEVAR?
VALUE
(LAMBDA (NDE)
   (NON-NILS (MAPCAR (DOWNSET NDE) (LAMBDA (X) (NOT (VAR (CDR X)))))))

PLIST
NIL

 *************************************************************************
 * [NON-NILS] IS A HELP FUNCTION TESTING IF A LIST CONTAINS ANY TOP LEVEL NIL
 *    AS AN ELEMENT.  THIS IS USEFUL USED ASSOCIATED WITH A [MAPCAR] FUNCTION.
 *                                          HYY -- 12/21/81


     NON-NILS
VALUE
(LAMBDA (LST)
   (COND ((NULL LST) T)
         ((NULL (CAR LST)) NIL)
         (T (AND (CAR LST) (NON-NILS (CDR LST))))))

PLIST
NIL

 *********************************************************************
 * [PACK-IN] IS AN AID TO [PUTIN-BASKET]
 *                                          HYY -- 12/21  (PROG (TND)
         (RETURN
          (COND
           ((NULL HHEAP) NIL)
           ((AND (MEMB (SETQ TND (CAR (GET (CAAR HHEAP) (CONV 'ARG))))
                       (GET 'NODES ':VAL))
                 (D-HORN? TND))
            (CONS (LIST TND (CAAR HHEAP) (ARGN (CAR HHEAP) 3) (ARGN (CAR HHEAP) 4))
                  (SELECT.HORNS (CDR HHEAP))))
           (T (SELECT.HORNS (CDR HHEAP)))))))

PLIST
NIL



     SET.PREG
VALUE
(LAMBDA (FL)
   (MAPC FL (LAMBDA (F) (PUT F 'LREGS: (ARGN (EVAL F) 2))))
   (APPLY OUTPUT (LIST 'DILMA FL)))

PLIST
NIL

 ************************************************************************
 * [SORT.HORNS] PUTS ALL USEFUL HORNS SELECTED BY [SELECT.HORNS] INTO A
```

```
 *   CONVINIENT FORMAT AS DESCRIBED IN [GATHER.HORNS].
 *                                        HYY -- 12/21/81


     SORT.HORNS
VALUE
(LAMBDA (HORNS)
   (PROG (HORNPILE)
         (MAPC HORNS
               (LAMBDA (HSET)
                  (IF (EQUISET (GET (CAR HSET) 'ARG) (MAPCAR (CDR HSET) CAR))
                      (SETQ HORNPILE (CONS HSET HORNPILE)))))
         (RETURN HORNPILE)))

PLIST
NIL

 **************************************************************
 * [TOPMOST-TOPINF] MODIFIED BY HYY ON 12/21/81
 *   IN ORDER TO ADD AN EXTRA REGISTOR <D-ANS:> THAT WILL RECEIVE AN ANSWER
 *   FROM A DILEMMA INFERENCE ROUTINE.  SHOULD BE FURTHER IMPROVED IN THE FUTURE.
 *                                          HYY -- 12/21/81


     TOPMOST-TOPINF
VALUE
(LAMBDA (NAME:    CLINK:   CQ:      DATA:    MSG:     N-ANS:   P-ANS:   TOT:     N-POS
         N-NEG:   /#SUSPS# BOSSES:  MTR:     D-ANS:)
   (IF D-ANS:
       (PRIN3 <> " HERE, WE INFER A DISJUNCTIVE ANSWER" <>)
       (APPLY DESCRIBE D-ANS:)
       (SETQ D-ANS: NIL))
   (IF (SETQ MSG: (MEMBER-S (MAPCONC MSG: (LAMBDA (X) (CDR X))) (SBINDS DATA:)))
       (SEND MSG: BOSSES:)
       (SETQ DATA: (APPEND DATA: MSG:))
       (MAPC
        (MAPCAR MSG:
                (LAMBDA (X)
                   (CONS (FIRST-ATOM (NBUILD (COND ((SAME-SIGN (CAR X) CQ:) CQ:)
                                                   (T (NEGATE CQ:)))
                                             (CADR X)
                                             FORBTOP))
                         (CDR X))))
        (LAMBDA (ANS)
           (IF (NOT (MEMBER (CAR ANS) RESULTS:))
               (SETQ RESULTS: (SNOC RESULTS: (CAR ANS))))
           (COND ((NEGATED (CAR ANS)) (SETQ N-ANS: (ADD1 N-ANS:)))
                 (T (SETQ P-ANS: (ADD1 P-ANS:))))))
       (IF (NOT (CONT? N-ANS: P-ANS: TOT: N-POS: N-NEG:))
           (SETQ
            /#SUSPS# (APPEND
                      /#SUSPS#
                      (MAPCONC
                       (APPEND SUSPS: EVNTS)
                       (LAMBDA (E)
                          (IF (OR (BELOWP E CURNT:)
                                  (MEMBER (REGFETCH E 'NAME:) '(I-MTR I-MTR-R)))
                              (LIST E)))))
            EVNTS   (MAPCONC EVNTS (LAMBDA (E) (IF (NOT (MEMBER E /#SUSPS#)) (LIST E))))
            SUSPS:  (MAPCONC SUSPS: (LAMBDA (E) (IF (NOT (MEMBER E /#SUSPS#)) (LIST E)))))
           (IF (EQ TP CURNT:)
               (MAPC EVNTS (LAMBDA (X) (SUSPENDEM X)))
               (MAPC SUSPS: (LAMBDA (X) (SUSPENDEM X)))
               (SETQ EVNTS NIL SUSPS: NIL)))
       (SETQ MSG: NIL))
   (SET CURNT:
```

```
        (LIST    NAME:    CLINK:   CQ:      DATA:    MSG:     N-ANS:   P-ANS:   TOT
         N-POS:   N-NEG:   /#SUSPS# BOSSES:  MTR:     D-ANS:)))

PLIST
(LREGS: (NAME:    CLINK:   CQ:      DATA:    MSG:     N-ANS:   P-ANS:   TOT:     N-POS
         N-NEG:   /#SUSPS# BOSSES:  MTR:     D-ANS:))

 ***********************************************************************
 * [UNTORN-CQ] HELPS [HORN-TORN-CQ] TO GENERATE A NEW CQ TO BE PROVED.
 *   [UNTORN-CQ] RETURNS A LIST OF NODES DISTINCT FROM HORN NODE.
 *                                     HYY -- 12/21/81


     UNTORN-CQ
VALUE
(LAMBDA (HCQ CQ)
   (COND ((EQ (CAR CQ) HCQ) (CDR CQ))
         (T (CONS (CAR CQ) (UNTORN-CQ HCQ (CDR CQ))))))

PLIST
NIL

 ***********************************************************************
 * [WORTH-DIL?] TESTS IF OR NOT A DILEMMA INFERENCING IS WORTH TO BE EVER
 *   ATTEMPTED FOR THE GIVEN <CQ>.  NO D-INFERENCE IS ATTEMPTED FOR AN ALREADY
 *   DISJUNCTIVE QUERY.
 *                                    HYY -- 12/21/81


     WORTH-DIL?
VALUE
(LAMBDA (CQ)
   (PROG (MAXI)
         (RETURN
          (OR (AND (SETQ MAXI (CAR (GET CQ 'MAX)))
                   (EQ MAXI (CAR (GET CQ 'MIN)))
                   (NON-NILS (MAPCAR (GET CQ 'ARG) ATOMOLECULE?)))
              (ATOMOLECULE? CQ)))))

PLIST
NIL




CROSS REFERENCE OF DILMA


ALL-HORNS?     D-ANSCAT

ATOMOLECULE?   D-HORN?  WORTH-DIL?

ATTACK.HORN    D-INFER

D-ANSCAT       DILEMMA-INFER

D-HORN?        SELECT.HORNS

D-INFER        DILEMMA-INFER

D-SEND         D-ANSCAT

DEDUCE*

DILEM-RPT      DILEMMA-INFER
```

```
DILEMMA         DEDUCE*

DILEMMA-INFER   INFER

DRAFT.D-ANS     D-ANSCAT

EQUISET         SORT.HORNS

GATHER.HORNS    INFER

HORN-ANSCAT     D-INFER

HORN-TORN-CQ    D-INFER

INFER           DEDUCE*

NO-FREEVAR?     D-HORN?

NON-NILS        ALL-HORNS?         ATOMOLECULE?       D-HORN?  EQUISET
                NO-FREEVAR?        NON-NILS WORTH-DIL?

PACK-IN         PACK-IN  PUTIN-BASKET

PUTIN-BASKET    GATHER.HORNS

RECORD.DANS     D-ANSCAT RECORD.DANS

SELECT.HORNS    GATHER.HORNS       SELECT.HORNS

SET.PREG

SORT.HORNS      GATHER.HORNS

TOPMOST-TOPINF D-ANSCAT DEDUCE*  DILEM-RPT          INFER

UNTORN-CQ       HORN-TORN-CQ       UNTORN-CQ

WORTH-DIL?      INFER
NIL
 ?(EXIT)
REVERT.
/
```