**American Journal of Computational Linguistics** *Microfiche 36 : 26*

# HOW DOES A SYSTEM KNOW WHEN TO STOP INFERENCING?*

STAN ROSENSCHEIN**

*The Moore School of Electrical Engineering*
*University of Pennsylvania, Philadelphia 19174*

Abstract The problem of constraining the set of inferences added to a set of beliefs is considered. One method, based on finding a minimal unifying structure, is presented and discussed. The method is meant to provide internal criteria for inference cut-off.

## I. Introduction

Natural language processing systems that are sensitive to the semantic and logical content of processed sentences and to the pragmatics of their use generally draw inferences. A set of formulas representing the meaning of a sentence and the 'state of belief' of the system is augmented by other related formulas (the inferences) which are retrieved and/or constructed during the processing. The problem to be investigated here is: How can this process be controlled? Can reasonable criteria be found for restraining the addition of inferences?

Top-down inferences following from the meaning of lexical items (often expressed by decomposition into primitives) are clearly bounded, if no interactions are allowed among the generated sub-formulas. This process (which we call EXPANSION) will not be discussed here. Rather, we shall be concerned with SYNTHESIS, i.e., the addition of new formulas based on the

* This work was partially suppored by NSF Grant SOC 72-05465A01.

** Author's current address: Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012.

presence of already generated lower-level formulas, which we shall call beliefs. In particular, we are concerned with inferences added because a set of beliefs is recognized as fitting a pre-defined pattern.

The question we ask is: Given an initial set of beliefs over a set of primitives, what criterion can be used to halt the process of pattern matching and associated inference addition? The major structural feature that we use to provide such a criterion is a partial order over the set of patterns.

Before pursuing this suggestion any further, let us examine some of the alternative approaches to inference and inference cut-off.

To logicians, deductive inference involves rules by which forumlas can be added to a set (which initially contains the axioms) in certain ways provided other formulas are already in the set. In general, this sort of inference is quite open-ended in that one can keep applying the rules of inference and come up with more and more formulas all of which represent 'provable' statements. The termination criterion for a particular invocation of the mechanism might be the appearance of an 'interesting' formula or the loss of interest of the inferencer, but in general the statement of the rules of inference says nothing about when to cease deriving formulas.

This paradigm from logic has been carried over into Artificial Intelligence systems, where the issue of termination is very real. The usual solution has been to invoke the inferencer under the very strict control of a supervising program which has its own goals programmed in which makes certain that appropriate criteria are applied to halt the inferencing. This is most apparent in systems written in PLANNER-like languages which has user-programmable mechanisms for controlling the proof process.

In the work of Schank and Rieger, (Sch, 75) (Ri, 74) inference has more of the flavor of free association; inferences are conceived of as expanding spheres in'inference space.' Two termination strategies are employed: (1) the discovery of a chain of inferences leading from one of the initial beliefs to another through a shared formula, or 'contact point' in inference space, and (2) the association of numerical 'strengths' to formulas so that a line of inference can be discontinued if the strength falls below a certain threshold.

Strategy (2) is somewhat unsatisfying in view of the potential arbitrariness and attendant difficulties in evaluating the role of particular numerical constants in the total behavior of a complex system. These constants, presumably, have little to do with the theoretical structure of the formal inference scheme, and as such we would call them 'external criteria.' A strategy like (1) above, on the other hand, is more 'internal' and is to be preferred.*

A goal of the present work is to formulate a reasonable internal criterion for inference cut-off which can be stated formally as part of the inference rule. To do this, we shall impose a structure on the set of patterns to be used in inferencing, and the rule for adding inferences will be formulated in terms of this structure.

The operations to be described below are explained more fully in (R,75), where a description of a computer implementation is also presented.

* See also (C,75), (W,75).

## II. A Partial Order for the Pattern Set

The inference rule we are aiming for is to depend on the set of input beliefs and the set of patterns. The notion we are trying to formalize is "What does this set of beliefs suggest with respect to this set of patterns?" The particular class of inferences we are concerned with are those gotten by matching beliefs in the input set against a pattern and augmenting the beliefs with additional propositions as dictated by the pattern. We want to find the least instances of patterns which cover (include) the set of input beliefs. We will take as inferences' all propositions (an arbitrary number) which are entailed by that instance of the pattern.

Put another way, the inference operation is to 'jump to conclusions. However, it is only to jump to those conclusion required to make the resulting set an instance of the least possible pattern in the pattern set.

The key concept here is 'least' in that this is what controls how many inferences are added. What would be a suitable ordering relation for patterns and propositional beliefs? One which naturally suggests itself and which is currently under investigation relies on the relations of instantiation and superset:

(1) $p \leq q$ if $q$ is a substitution instance of $p$,

and (2) $S \leq_1 S'$ if $S \subseteq S'$.

Combining these two, we say that $\{p_1,\ldots,p_n\} \leq \{q_1,\ldots,q_m\}$, where the $p_i$'s and $q_j$'s are propositional forms, if there is a substitution, $s$, for the variables of $\{p_1,\ldots,p_n\}$ such that $\{s(p_1),\ldots,s(p_n)\} \leq_1 \{q_1,\ldots,q_m\}$.

Example 1.

We adopt the notational convention of prefixing variables with '?'.

Let P = {(HAPPY ?x), (PARENT ?y ?x)}

and let Q = {(HAPPY JOHN), (GIVE MR. JONES JOHN TOY), (PARENT MR. JONES JOHN)}.

Then P $\leq$ Q under the substitution ?x←JOHN, ?y←MR. JONES.

The 'less-than-or-equal' relation is also defined for pairs of patterns:

Example 2.

Let PAT-1 = {(P ?x ?y), (Q ?y ?z)}

and let PAT-2 = {(R ?u ?v ?w), (Q ?w ?v), P ?u ?w)}.

Clearly, PAT-1 $\leq$ PAT-2 under the substitution ?x←?u, ?y←?w, ?z←?v.

This definition of $\leq$ is quite straightford and can be made to accomodate expressions with embeddings and predicate variables. (These are included in the implementation.)

Note that the relation '$\leq$' can be thought of an information-content comparison; if S $\leq$ S' then S' contains at least as much 'information' as S (and possibly more) either by virtue of variables having been replaced by particular constants or by additional formulas having been added to the set.

Given $\leq$ for relating pairs of belief sets, pairs of patterns, or belief-set/pattern pairs, we can now formulate the belief-set-extending function, SYNTHESIZE.

## III. The Inference Operation: SYNTHESIZE

Given a set of P of patterns and an input set Bel of beliefs, SYNTHESIZE returns a set I of instantiated patterns from P such that the following three conditions all hold:

(1) (Coverage of input beliefs) For each instantiated pattern p $\in$ I, Bel $\leq$ p;

(2) (Pairwise incomparability) If $p, q \in I$ then

$p \not\leq q$ and $q \not\leq p$;

(3) (Minimality) There are no other instances $r$ of patterns in $P$ which are not in $I$ and yet which are '$\leq$' to some element of $I$ and for which $Bel \leq r$.

The elements of $I$ = SYNTHESIZE(Bel) represent possible minimal extensions of Bel; $\bigcap I$ represents clear extensions of Bel, namely the superset of Bel contained in all minimal extensions.

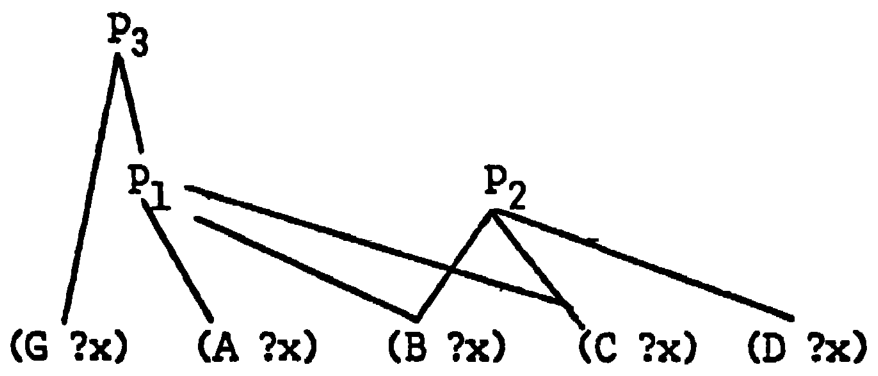
Example 3.

Let $P = \{ p_1 = \{$(A ?x), (B ?x), (C ?x)$\}$,

$p_2 = \{$(B ?x), (C ?x), (D ?x)$\}$,

$p_3 = \{$(A ?x), (B ?x), (C ?x), (G ?x)$\}$ $\}$

Represented graphically:



If input set Bel = $\{$(A JOHN), (C JOHN)$\}$

then SYNTHESIZE(Bel) = $\{$ $\{$(A JOHN), (B JOHN), (C JOHN)$\}$ $\}$.

There is only one possible minimal extension; (B JOHN) is inferred.

If input set Bel = $\{$(B JOHN), (C JOHN)$\}$

then SYNTHESIZE(Bel) = $\{$ $\{$(A JOHN), (B JOHN), (C JOHN)$\}$

$\{$(B JOHN), (C JOHN), (D JOHN)$\}$ $\}$.

There are two possible minimal extensions, but the set of clear extensions contains no inferences beyond the input set, Bel. (Had $p_1$ and $p_2$ shared another clause, however, an inference would have been added.)

If the input set Bel = {(G JOHN), (B JOHN)}

then SYNTHESIZE(Bel) = { {(G JOHN), (A JOHN),
(B JOHN), (C JOHN)} }.

Pattern $p_3$ is the least pattern which when instantiated covers the inputs, and there are two inferred propositions:
(A JOHN) and (C JOHN).

The description given here has been necessarily brief and incomplete A more formal treatment of SYNTHESIZE in terms of lattice-theoretic operations is given in (R,75) and is summarized in (JR,75). One additional technical point should be made: It often happens that for a given input set there are _no_ single pattern instances which cover all the inputs, though patterns exist whose instances cover subsets of the inputs. In such a case we use an _extended_ SYNTHESIZE operation which is defined in the same spirit as SYNTHESIZE. (See (R,75).)

Even without the full formal treatment, several things should now be clear. First, the actual number of inferences drawn (propositions added) for a particular input set may be small or large (depending on the inputs and the pattern set,) but it is bounded in a principled way because of the definition of SYNTHESIZE.

Second, the usual distinction between 'antecedent' and 'consequent' clauses in the pattern is not maintained; a clause in the pattern may serve as an antecedent on one occasion and a consequent on another.

Third, if 'defined' lexical items were to be associated with the patterns, noting which variables are to be bound as arguments upon instantiation, then the SYNTHESIZE function can be used to compute summarizing expressions. Thus SYNTHESIZE represents a possible formalism for lexical insertion.

## IV. An Example of the Operation SYNTHESIZE

For the sake of illustration, let the primitives be:

(BENIGN ?x)

(THREATEN ?x ?y) -- ?x threatens ?y

(GIVE ?x ?ob ?y) -- ?x gives ?ob to ?y

(BELONG ?ob ?x) -- ?ob belongs to ?x

(INTEND ?x ?Q) -- ?x intends to do ?Q

(RETURN ?x ?ob ?y) -- ?x returns ?ob to ?y

(PAYS-INTEREST ?x ?y) -- ?x pays interest to ?y

(These primitives and the patterns below may appear somewhat artificial, but we have chosen a simple illustration due to the difficulties in following examples with more than a few clauses.)

Let the pattern set consist of the following four patterns:

PAT-1: ?x borrows ?ob from ?y:

{(BENIGN ?x), (BELONG ?ob ?y), (GIVE ?y ?ob ?x),
(INTEND ?x (RETURN ?x ?ob ?y))}

PAT-2: ?x takes-loan-from ?y:

{(BENIGN ?x), (BELONG ?ob ?y), (GIVE ?y ?ob ?x),
(INTEND ?x (RETURN ?x ?ob ?y)), (PAYS-INTEREST ?x ?y)}

PAT-3: ?x robs ?y:

{(NOT (BENIGN ?x)), (BELONG ?ob ?y), (THREATEN ?x ?y),
(GIVE ?y ?ob ?x), (NOT (INTEND ?x (RETURN ?x ?ob ?y)))}

PAT-4: ?x plays-practical joke on ?y:

{(BENIGN ?x), (BELONG ?ob ?y), (THREATEN ?x ?y),
(GIVE ?y ?ob ?x), (INTEND ?x (RETURN ?x ?ob ?y))}

A rough graphic representation of the set of patterns is shown in Figure 1.

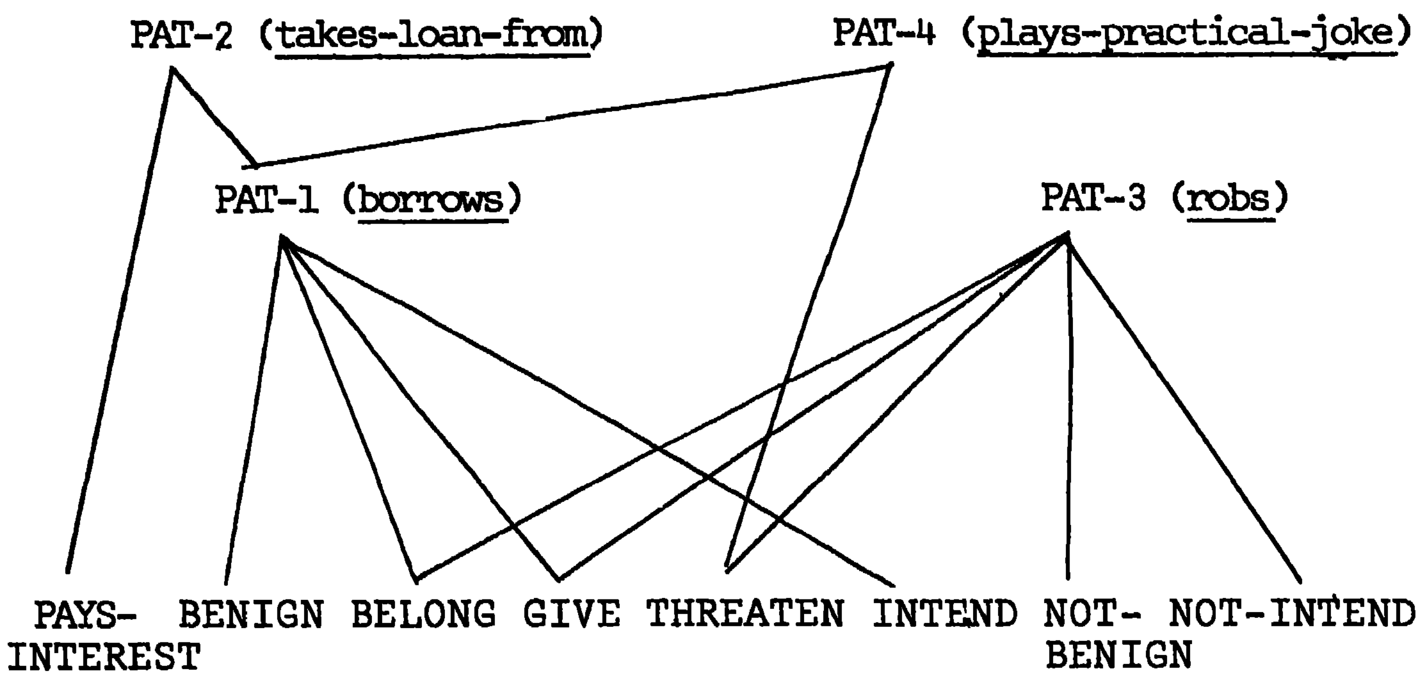


Figure 1

Now consider the following situations:

Situation 1. Input beliefs

Bel = {(BELONG WALLET HARRY), (GIVE HARRY WALLET MOE)}

SYNTHESIZE(Bel) =

{{(BELONG WALLET HARRY), (BENIGN MOE), (GIVE HARRY WALLET MOE),

(INTEND MOE (RETURN MOE WALLET HARRY))}

{(NOT BENIGN MOE)), (BELONG WALLET HARRY), (THREATEN MOE HARRY),

(GIVE HARRY WALLET MOE),

(NOT (INTEND MOE (RETURN MOE WALLET HARRY)}}

The minimal matched patterns are rob and borrow, adding the (conjectural) information that either Harry was threatened, or Moe intends to return the wallet.

Situation 2. Input beliefs:

```
Bel = {(GIVE BANK 1000-DOLLARS JOHNDOE),(PAYS-INTEREST JOHNDOE BANK)}
SYNTHESIZE(Bel) =
   {{(BENIGN JOHNDOE), (BELONG 1000-DOLLARS BANK),
           (GIVE BANK 1000-DOLLARS JOHNDOE),
           (INTEND JOHNDOE (RETURN JOHNDOE 1000-DOLLARS BANK)),
           (PAYS-INTEREST JOHNDOE BANK)}}
```

As a result of matching the loan pattern, we have added three clauses.

Situation 3. Input beliefs

```
Bel ={(INTEND JOHNDOE (RETURN JOHNDOE 1000-DOLLARS BANK)),
        (PAYS-INTEREST JOHNDOE BANK)}
```

Here SYNTHESIZE(Bel) returns exactly the same set as was returned in Situation 2. Note, however, that the roles of

(1) (GIVE BANK 1000-DOLLARS JOHNDOE)

and (2) (INTEND JOHNDOE (RETURN JOHNDOE 1000-DOLLARS BANK))

have been reversed. In Situation 2, (1) was an input and (2) was inferred, whereas in Situation 3, (2) was input and (1) inferred. The corresponding clauses of the loan pattern were serving as antecedents on one occasion and consequents on the other. This follows naturally from the way SYNTHESIZE was defined.

In this regard the reader may notice that some input belief sets might yield 'unwarranted' or 'spurious' inferences--jumping to too many conclusions. However, the incremental addition of new patterns corrects this anomaly in a natural way: Patterns which formerly were 'least covers' may cease to be so in the extended pattern set.

## V. Using Definitions to Set Up the Pattern Space

We have been particularly interested in using definitions of words to set up pattern spaces in which SYNTHESIZE could work as an inferencer and a lexical insertion technique. Special attention was payed to the 'speech act' verbs, and a brief sample list is presented below. (The symbol '?Pr' denotes a predicate variable. Also, primitive predicates are capitalized, while defined predicates are underlined.) Again, the definitions are greatly oversimplified for illustrative purposes.

```
(define tell (?x ?y ?p ?t)
     (and (BEFORE ?t0 ?t)
          (NOT (KNOW ?y ?p ?t0))
          (SAY ?x ?y ?p ?t)
          (KNOW ?y ?p ?t)
          (CAUSE (SAY ?x ?y ?p ?t)(KNOW ?y ?p ?t))))
(define request (?x ?y ?p ?t)
      (tell ?x ?y (WANT ?x ?p ?t) ?t))
(define promise (?x ?y ?Pr ?t)
      (and (FEELS-OBLIGATED ?x (?Pr ?x) ?t)
           (tell ?x ?y (INTEND ?x (?Pr ?x) ?t) ?t)))
(define command (?x ?y ?Pr ?t)
        (and (AUTHORITY-OVER ?x ?y)
             (request ?x ?y (?Pr ?y) ?t)))
(define implore (?x ?y ?Pr ?t)
        (and (WANTS-FAVOR-FROM ?x ?y)
             (request ?x ?y (?Pr ?y) ?t)))
```

The expansion of these items to patterns over the primitives yields a set in which, for example, KNOW ≤ <u>tell</u> ≤ <u>request</u> ≤<u>command</u>. The input set Bel = {(BEFORE t1 t2), (SAY JAMES MASTER (INTEND JAMES (OPEN JAMES DOOR) t2) t2), (FEELS-OBLIGATED JAMES (OPEN JAMES DOOR) t2)}

would be synthesized to (<u>promise</u> JAMES MASTER (OPEN + DOOR) t2), with added inferences (KNOW MASTER (INTEND JAMES (OPEN JAMES DOOR) t2) t2), etc., as dictated by the pattern instance of <u>promise</u>.

## VI. <u>Conclusion</u>

A method has been proposed for 'free' inferencing by pattern matching in which inference cut-off can be structurally constrained: A pattern is matched if it is one of the minimal patterns whose instantiation covers the input information--even if this necessitates adding an arbitrary amount of additional information. Similarly, on the question of how many inferences to draw: Enough extra inferences are drawn to enable a coherent pattern to be matched.

The method we have proposed is general in that it makes no assumptions about the particular predicates to be used in the patterns and beliefs. (Of course, it does make assumptions about what counts as a pattern or a belief.) The inferencing could be done by a general purpose component which accepts a set of patterns as a parameter. Thus, a programmer designing a system for inference by pattern match need not devise external criteria, and certainly not criteria to be associated with every pattern. Rather the criteria are implicit in the system as a whole; any patterns which can be described in a very general pattern description language will generate its own set of internal criteria for inference cut-off.

We are continuing to investigate formalisms for structuring pattern sets in the hope of gaining further insights into this class of inferences.

References

(C,75) Clark, H.H. "Bridging," in Proceedings of the Workshop on Theoretical Issues in Natural Language Processing. Cambridge, Massachusetts., June 1975.

(JR,75) Joshi, A.K. and Rosenschein, S. "A Formalism for Relating Lexical and Pragmatic Information," in Proceedings of the Workshop on Theoretical Issues in Natural Language Processing. Cambridge, Massachusetts. June 1975.

(Ri,74) Rieger, C. Conceptual Memory. Ph.D. Thesis Stanford University. Stanford, California. 1974.

(R,75) Rosenschein, S. Structuring a Pattern Space, with Applications to Lexical Information and Event Interpretation. Doctoral Dissertation. University of Pennsylvania. Philadelphia, Pennsylvania. 1975.

(Sch,75) Schank, R., Goldman, N., Rieger, C., and Riesback, C. "Inference and Paraphrase by Computer " Journal of the A.C.M. Volume 22, No. 3. July, 1975.

(W,75) Wilks, Y. "A Preferential, Pattern-seeking Semantics for Natural Language Inference." Artificial Intelligence. Volume 6. 1975.