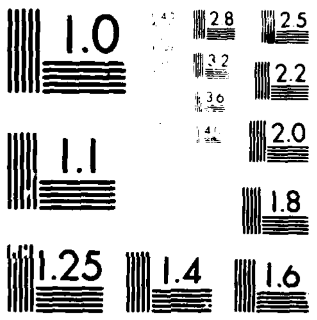

1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

12

NOSC

NOSC TD 298

LEVEL II

Technical Document 298

STAMMER2 PRODUCTION SYSTEM FOR TACTICAL SITUATION ASSESSMENT

Volume 1 — Design description
(Volume 2 consists of the code)

DC McCall (NOSC Task Leader)
PH Morris, DF Kibler, RJ Bechtel
(SDC Integrated Services)
Contract N00123-76-C-0172

October 1979

Prepared for
Naval Electronic Systems Command (NAVELEX 330)
Washington DC 20360

NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

80 5 7 060

ADMINISTRATIVE INFORMATION

Work was performed by the Tactical Command and Control Division (Code 824) as a part of the Tactical Situation Assessment (TSA) problem under Program Element 62721N, Project F21201, Task Area XF21201100 (NOSC 824-CC18). This TSA task is a part of the Command Control Block Program sponsored by NAVELEX, Code 330–the Command and Control Division of Research and Technology Directorate, NAVELEX Code 03.

This document was written by PH Morris, DF Kibler, and RJ Bechtel, of SDC Integrated Services, under Contract N00123-76-C-0172. It covers work from June through September 1979 and was approved for publication 28 November 1979.

Released by
RC Kolb, Head
Tactical Command and
Control Division

Under authority of
JH Maynard, Head
Command Control–Electronic Warfare
Systems and Technology Department

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. ~~REPORT~~ NUMBER 18 NOSC Technical Document 298 (TD 298)	2. GOVT ACCESSION NO. AD-A084 038	3. RECIPIENT'S CATALOG NUMBER 9 19 TD-298-VOL-1
4. ~~TITLE (and Subtitle)~~ 6 STAMMER2 PRODUCTION SYSTEM FOR TACTICAL SITUATION ASSESSMENT. Volume 1. Design description.		5. TYPE OF REPORT & PERIOD COVERED Technical Document Jun[redacted]Sept[redacted]79
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 DC McCall ~~(NOSC Task Leader)~~ PH Morris, DF Kibler, RJ Bechtel (SDC Integrated Services)		8. ~~CONTRACT OR GRANT~~ NUMBER(s) 15 N00123-76-C-0172
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Ocean Systems Center San Diego CA 92152		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 16 62721N, F21201 17 XF21201100 (NOSC 824-CC18)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Electronic Systems Command (NAVELEX 330) Washington DC 20360		12. ~~REPORT DATE~~ 11 Oct[redacted]79
		13. NUMBER OF PAGES 80 12 85
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. ~~(of this report)~~ Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

STAMMER2	Tactical analyses
Production systems	Correlation techniques
Rule-based inference systems	Merchant detection
Confidence factors	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

STAMMER2 is a revised version of STAMMER, a System for Tactical Assessment of Multisource Messages, Even Radar. STAMMER was created as part of an investigation of new correlation methodologies, and served as a testbed for explorations of applications of rule-based inference systems to the tactical situation assessment (TSA) problem. STAMMER concentrated on the specific task of merchant detection from radar and external messages. Experience with STAMMER revealed areas for improvement, which have led to the creation of STAMMER2. In addition to several changes in the underlying rule mechanisms used, the enhancement found in STAMMER2 arose out of a desire for greater generality and flexibility in the demonstration system, the explanation system, and the range of acceptable inputs to the system. STAMMER2 should prove to be a more useful system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

[illegible]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

20. ABSTRACT (Continued)

for testing various rule/scenario collections. During the development that led to STAMMER2, further issues in the design of rule-based inference systems for use in support of C3 activities have become apparent and they are discussed. This volume consists of the design description. Volume 2 consists of the code.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

Table of Contents--Volume 1

CHAPTER 1
INTRODUCTION

1.1 Purpose

STAMMER2 is a revised version of STAMMER[1], a System for Tactical Assessment of Multisource Messages, Even Radar. STAMMER was created as part of an investigation of new correlation methodologies, and served as a testbed for explorations of applications of rule-based inference systems to the tactical situation assessment (TSA) problem. STAMMER concentrated on the specific task of merchant detection from radar and external messages.

Experience with STAMMER revealed areas for improvement, which have lead to the creation of STAMMER2. In addition to several changes in the underlying rule mechanisms used, the enhancements found in STAMMER2 arose out of a desire for greater generality and flexibility in the demonstration system, the explanation system, and the range of acceptable inputs to the system.

STAMMER2 should prove to be a more useful system for testing various rule/scenario collections. During the development that lead to STAMMER2, further issues in the design of rule-based inference systems for use in support of C3 activities have become apparent and are discussed at the end of this report.

1.2 Overview of Changes

In converting STAMMER to STAMMER2, the following changes have been made.

---

1 NOSC TD 252, STAMMER: System for Tactical Assessment of Multisource Messages, Even Radar, by RJ Bechtel and PH Morris, May, 1979.

1.2.1 In memory

- o Assertions may now be n-tuples rather than only triples.
- o Nonatomic elements are permitted in assertions.
- o The internal representation of assertions has changed, using somewhat more space for a significant decrease in retrieval time.
- o Retrievals can now bind an arbitrary number of variables rather than the single one of the old system.

1.2.2 In the oracles

- o Certain distinguished oracles may now bind arguments.

1.2.3 In the confidence system

- o The confidence of assertions which satisfy negative conditions is now included in the confidence calculated for derived assertions.
- o Possible derivation cycle traps are detected and avoided, and the loops are treated appropriately in confidence calculation.
- o The confidence calculation process has been revised to more closely match that used by MYCIN[2]. Remaining differences arise from the differences in interpreters.

---

[2] Computer-Based Medical Consultations: MYCIN, by EH Shortliffe; Elsevier, 1976.

1.2.4 In the rules and rule interpreter

- o The format of rule conditions and actions has changed slightly.
- o Rule conditions may now contain the connectives NOT and OR.
- o Rule actions may now include disjunctive conclusions and report generation calls.
- o The rule interpreter centers on a new data type, called streams, which simulate parallel processing and permit "automatic" suspension and resumption of processes.
- o Rule interpretation is no longer an explicit part of the execution cycle.

1.2.5 In the explanation system

- o Slightly different questions and formats.
- o Use of a replacement for ASKUSER to improve response and correction time and reduce user frustration.
- o Revision of rule explanation formats.
- o Addition of graphics to explanation and justification of some rules.
- o A more flexible assertion prettyprinter.

1.2.6 In the message receipt system

- o A modification to LWES to permit its use as a scenario generator.
- o Revisions to the message monitor to permit use of LWES generated scenarios.
- o Messages are read from disk, rather than absorbing memory space.
- o Messages from new sensor types (sonar, EW) are permitted.

1.2.7 Elsewhere

- o A number of smaller test scenarios have been created, permitting the demonstration of the system and/or particular rules without the need to step through a long demonstration.

- o The modularity of the system has been improved, making incremental additions and refinements easier.

- o The display system has been modified to function properly on a Tektronix 4025.

- o New rules have been added to perform more general track identification.

1.3 Significance of STAMMER2 to Artificial Intelligence

1.3.1 Toy problems versus real problems

Workers in artificial intelligence (AI) argue over the benefits of studying researcher-defined toy problems as opposed to studying naturally occurring real-world problems. A toy problem allows the researcher to abstract an important problem and to concentrate his efforts on a general solution. He may shift his attention and viewpoint. He is not disturbed by the exigencies of producing a working system. Toy problems, however, allow the researcher to trivialize the problem, to choose a problem of little importance, and, because of the small size, to allow solutions which are in general impractical and consequently no solutions at all. Real-world problems that require novel AI approaches force the researcher to produce new ideas/techniques or to fail. He may not change the problem. The problem size limits the possible solutions. Computers have finite memory and operate at finite speeds. People also seem to have a finite memory and to think at a finite speed. This resource limitation is a fundamental constraint on the nature of solutions to problems in AI. This constraint cannot be sidestepped in solving real-world problems. Real-world problems have the disadvantages that they may be too specialized, that the pressure of producing a running system will absorb too much of the researcher's time, that the problem attacked is inappropriate, and that the solution is too ad hoc. In the end, the value to artificial intelligence of work on either a toy problem or a real-world problem will be determined by the quality of the work. Both domains allow useless and useful results.

STAMMER2 deals with information on a real-time basis. This process is further complicated by the fact that the information is suspect, and its arrival may not be in chronological order. STAMMER2 makes whatever conclusions it can on the basis of its current knowledge. Other real-time AI systems, such as HARPY[3] or HEARSAY[4] , are provided with complete information. STAMMER2 uses partial knowledge and later, if other relevant information comes in, must revise its conclusions. The combination of these problems leads to a number of issues in AI.

1.3.2 Particular AI issues

- Real-time explanation
  The explanation system is written as a miniproduction system (see 2.6 The Explanation System). Unlike MYCIN, STAMMER2 faces the problem of generating explanations for assertions it no longer has confidence in. A method for saving the derivation of an assertion is incorporated into STAMMER2 (see 2.4 Confidence).

- Semantics of Logical Connectives
  The semantics of the connectives "and, or, not" are given by the manner in which they affect the computation of the confidence. MYCIN and PROSPECTOR[5] gave different semantics to these operators, but there is no agreement on which method is better. STAMMER2 follows the MYCIN approach to confidence

---

3 The HARPY Speech Understanding System, by BT Lowerre and R Reddy; Trends in Speech Recognition, WA Lea, ed, Prentice-Hall, 1979.

4 The HEARSAY Speech Understanding System: An Example of the Recognition Process, by DR Reddy, LE Erman, RD Fennell, and RB Neely; Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford CA, 1973, p 185-193.

5 Technical Report 136, Semantic Network Representations in Rule-Based Inference Systems, by RO Duda, PE Hart, NJ Nilsson, and GL Sutherland, SRI International, 1977.

computation but extends the set of connectives to include "unless" (see 2.4 Confidence).

- Dynamic Confidence
  People permit the confidence of an assertion such as "I should move to London" to vary in time. STAMMER2 must do the same. Consequently STAMMER2 does not store a confidence with an assertion as in MYCIN; rather, it computes the confidence on demand.

- Reasoning Cycles
  Formal mathematical arguments eschew circular reasoning, but circular derivations arise naturally in reasoning with judgemental knowledge (see 2.4 Confidence). In STAMMER2, a solution to the problem of assigning a confidence to an assertion which has cycles in its reasoning chain is implemented.

- Efficiency of Production Systems
  The efficiency of production system formalisms is a well-acknowledged problem, and several researchers have proposed techniques to ameliorate the problem. By organizing the data into streams, STAMMER2 avoids all *recomputations* of rule-firing conditions. By means of a combinatorial hashing scheme, STAMMER2 can examine its data base in a time proportional to the number of retrieved items.

With the exception of the problem of efficiency, AI has paid little attention to the above issues. The construction of STAMMER2 has led to their discovery and provided some reasonable preliminary solutions.

CHAPTER 2
DESIGN AND IMPLEMENTATION ISSUES

2.1 Introduction

While the redesign of STAMMER involves many decisions on many levels, some of the most important issues concern the dynamic data base and rule interpretation. A rule-based language has a number of advantages, but with very large rule sets and large collections of data, these advantages may be overwhelmed by computational inefficiency and difficulty of use. These problems motivated most of our redesign effort.

In the existing STAMMER system, the dynamic data base was a source of both awkward usage and computational inefficiency. While the assertion format could be used to create fairly general structures, the restriction to two-place relations forced occasional circumlocutions on users, causing the creation of tangled groups of assertions to represent conceptually simple relationships. In the redesign, we retained the assertion concept while generalizing it to permit an arbitrary number of arguments for the relations involved. Thus in addition to the existing binary relations, users may define unary, ternary, and n-ary assertion forms.

The efficiency problem involved greater effort. Retrieval of assertions is a pattern-matching process, which requires search. The match was determined on the basis of the content of the assertion. What happens is essentially a simulation of an associative or content-addressable memory. To improve efficiency, we wished to improve this simulation. The goal of the improvement was a retrieval time roughly proportional to the number of assertions retrieved. This can be achieved by a sophisticated data structure representation. Our preferred idea is related to that of combinatorial hashing , discussed in Knuth[6].

Another efficiency improvement incorporated in the dynamic data base retrieval mechanism is designed to save space

6 The Art of Computer Programming, Vol 3: Sorting and Searching, by D Knuth, Addison-Wesley, 1975.

rather than time (although it should have minimal time overhead). This change replaces the traditional returned list of results with a mapping of some function over the assertions which match the retrieval specification. Instead of creating a new list of retrieval results to be used by a function, the function is applied directly to assertions as they are retrieved. This is roughly like applying a function to members of a list as they are generated rather than passing a copy of the list as an argument to a function. The implementation details of these efficiency improvements are invisible to the user.

The other area which promised great gains in clarity and efficiency was the rule evaluation mechanism. For clarity, we expanded the expressive power of rules, allowing arbitrary Boolean combination of conditions along with an improved treatment of negation. We also provided a mechanism to permit what we are calling "semantic loops," caused by rules which might create cycles in deduction paths (eg If *x is a sibling of *y, then *y is a sibling of *x), but are nevertheless useful. These refinements, along with others, required some changes in the handling of confidence factors as well.

Our approach to improved efficiency may be described as "incremental deduction." It is similar to the change-driven approach which is used in a limited way in STAMMER and mentioned in the RAND rule-based language effort (called ROSIE, for rule-oriented system for implementing expertise). Incremental deduction also deals with the problem of rule selection. The central idea in incremental deduction is that a rule evaluates its conditions either until it succeeds or until a condition fails. When a condition fails, a "suspension" is created which corresponds to the remainder of the rule. As more information is added to the data base, those suspensions which can use the new data are revived and continue as before (either completing or suspending again). A suspension contains not only the remaining part of a rule, but also the bindings established by already satisfied conditions, if any. Since the evaluation of a single condition is much the same as in the present system, all possibilities are considered when the rules are first fired. Even when a condition succeeds, there may be other ways for it to be satisfied, so a suspension is left behind.

An obvious way of implementing suspensions is through coroutines. However, the shallow binding scheme of INTERLISP makes context switching expensive in the system-supported coroutining regime. For this reason, we've provided our own coroutining/context switching mechanism.

These are the issues that we have addressed during this redesign at the theoretical mechanism level. At a somewhat different level, we have implemented a method of using graphics to explain specific rule firings.

2.2 Memory

One of the major revisions that distinguish STAMMER2 is a change in the way that assertions were represented, stored, and accessed. The goals of the change were

- o to provide greater flexibility in the form of assertions by permitting n-tuples rather than just triples and by permitting nonatomic elements in assertions
- o to increase the speed of data-base retrievals, while having a minimal effect on required storage
- o to better support the revised rule interpretation mechanism.

In order to describe the new memory and its changes, we'll first review the structure of the old memory.

2.2.1 Original design and implementation

The original STAMMER had a memory made up of assertions which were triples of the form (Relation argument argument) (usually written in infix, eg aRb). Each assertion was represented in memory by a unique identifier, called the assertion node, where the elements of the assertion were stored along with confidence information. The assertion node and assertion items were associated, with forward associations linking assertion nodes to elements and back associations linking elements to assertion nodes. Every assertion node pointed to the elements of its relation, and every element (relation or argument) pointed to all the assertions in which they participated. The approximate cost in storage was acceptable, but the scheme required an unreasonable amount of time and intermediate storage to perform retrievals.

Retrievals were performed by chasing pointers and doing intersections. A primary requirement was the ability to retrieve an assertion or list of assertions given a list of elements. The assertions retrieved were to be those in which those elements participated -- in short, a total match on a partial specification. While the actual algorithm was somewhat more refined, the basic procedure involved looking at those elements (keys, in data-base terms) which were known to be in the desired assertions, finding the assertions in which those elements participated, then (if there was more than one known element) intersecting the resulting lists of assertions. Intersection is computationally quite expensive, requiring O(n2) time for calculation. Thus, while this representation was not excessive in space, the time cost of retrieval was bad enough to force reconsideration.

In addition, maintaining the pointer structures on the property lists of assertion elements forced all assertion elements to be of a particular INTERLISP data type (litatoms), since no other data type has property lists. Since litatoms specifically exclude numbers, this created particular problems in the representation of numbers, forcing an encoding and decoding sequence which required still more time. However, the cost of intersection in retrieval was the motivating factor in the memory redesign.

2.2.2 Desired features

Like many data-base retrieval problems, the difficulty in finding a fast, relatively small representation and retrieval method for memory grew out of the architecture of standard computers. In a standard computer (Von Neumann architecture), memory is referenced by location, while for data-base retrieval, one wishes to recover elements based on their content, rather than incidentals such as where the information is stored. Ideally, the memory component would be handled on a content-addressable machine[7]. However, such machines are still rare and in present implementations may still be unsuitable. Instead, it is necessary to simulate content addressability on a location-addressed machine. The property list/pointer representation of the original STAMMER provided one way of doing this at the cost of intersections. A more direct method would be to provide a function which maps content into location. If the address (location) space is smaller than the content space, this mapping is called

7
Content Addressable Parallel Processors, by CC Foster, Van Nostrand Reinhold, 1976.

hashing; this forms the basis for memory organization in STAMMER2.

2.2.3 New design

As before, memory is made up of assertions, each with a unique assertion node identifier. Assertions are statements that some relation holds over some set of arguments. An assertion's inclusion in memory, however, is not in itself a commitment to the truth or falsity of the statement but rather is simply a representation of the statement. (Truth and falsity, or belief and disbelief, are handled separately. See 2.4 Confidence.) The assertion node is associated with the assertion (and assertion elements) and an access function is provided which returns an assertion when given an assertion node. (At present, the standard LISP binding mechanism provides the association and the access function is a simple EVAL, but any suitable association/access function combination could be used.) With such an access function available, links to an assertion may be made instead to the appropriate assertion node, at the cost of one node-assertion access. This is the standard way of doing things, so that when reference is made to assertions, as in "associated with an assertion," it is in fact the assertion node that is used in the association.

The major difference is in the way assertion elements are associated with assertion nodes in which they participate. Instead of establishing an association from each element to the assertions in which it participates, associations are created between combinations of elements and the assertions in which those combinations appear. Thus for a typical assertion (R a b), the old method would have established three "back associations":

1. R -> (R a b)

2. a -> (R a b)

3. b -> (R a b)

By suitable intersection of such associations, all assertions containing (R * b), where * represents a variable, could be retrieved. For the same assertion, the new method would establish eight back associations:

1. (R a b) -> (R a b)

2. (* a b) -> (R a b)

3. (R * b) -> (R a b)

4. (* * b) -> (R a b)

5. (R a *) -> (R a b)

6. (* a *) -> (R a b)

7. (R * *) -> (R a b)

8. (* * *) -> (R a b)

This is certainly more space, but notice that now the retrieval of all assertions (R * b) is a one-step, constant time cost operation. Further, to massage the associated assertions to recover all bindings for the "missing" argument costs O(n), just as under the old scheme. However, the combined costs of the two retrievals required to bind variables are $O(n^3)$ under the old system and O(n) under the new. For an increased cost in memory of approximately $(2^k)-k$ (where k is the average number of elements in an assertion), there has been a significant gain in speed. For short assertions, the time gain can readily justify the storage expense.

2.2.4 Implementing the new design

The first step in implementation was to find an appropriate method for representing the back associations. The possible combinations of elements are referred to as retrieval specifications and are represented as lists. Since they are not atoms, the "standard" LISP association technique of property list pointers could not be used without some sort of coding and decoding. INTERLISP provides another method of establishing "content addressable" association much like property lists, called hash arrays[8]. However, the system implementation of hash arrays also assumes atoms as retrieval keys by using EQ for collision tests. This was also not acceptable. However, unlike property lists, it was possible to write a version of the hash array facility which did not assume atoms as keys. This was done, providing creation, insertion, and retrieval functions that closely match those of the system facility

8 INTERLISP Reference Manual, by W Teitelman, Xerox Palo Alto Research Center, 1978.

but which rely on EQUAL, rather than EQ. Some additional functions were also written to perform commonly desired combinations of primitive hash array functions. In use as a method of storing back associations, the hash array works as follows:

> Given a retrieval specification, an array address is generated. If the contents of the key field of the array at that address match the retrieval specification, the data field is returned. Otherwise a new address is generated, and the equality test (match) and remaining steps are repeated.

The most difficult part of the hash array implementation was devising a good hashing algorithm (address generator).

With the basic mechanism implemented, a number of decisions were made that reduced the total space required by this representation regime. A study of the particular application of memory revealed that the most general mechanism, which stores 2 back associations, would store data under retrieval paths that would never be used. By preventing storage under these paths, the space cost could be reduced in this particular application. It is important to note that these reductions are very application dependent, and are not valid in general.

In this application, it was not necessary to store any retrieval specifications in which the relation element was variable (specifications of the form (* anything anything ...)). This reduced the number of back associations from 2^k to 2^{k-1}. Thus, for the canonical three-place assertion, the following retrieval specifications were used as keys:

1. (R a b)

2. (R * b)

3. (R a *)

4. (R * *)

This is a significant gain, especially since most assertions are three-place. Under this filter, storage cost for the time gain is very favorable.

2.3 Rules and the Rule Interpreter

2.3.1 Rule formats

A large increase in flexibility has been provided in STAMMER2 by increasing the expressive power of the rules. This increased power has necessitated a change in the format of the rules. While rules still have conditions, actions, and confidences, the forms of the conditions and actions are different. Previously, conditions and actions contained explicit retrieval and construction function calls. These were necessary to insure correct variable bindings. Since the rule variable binding scheme has been changed to allow binding of multiple variables in a single condition, uniform retrieval and construction functions are now used, obviating the necessity for explicit function calls.

The new condition and action format has a further advantage. Under the new representation, conditions and actions have the same format as standard assertions, making the rules more clearly readable, and making it possible to utilize the assertion prettyprinting functions to describe the rules.

The best way to describe the changes in rule formats is simply to show a rule in the old format and the new. A fairly simple rule, in the old form, looked like the following:

```
(GETOP M MERCHANTLANE)
(GETS X (RETRIEVE2 (QUOTE % )(QUOTE PLATFORM)))
(UNLESS (RETRIEVE3 (QUOTE AIR) (QUOTE MEDIUM) X))
(GETS Y (RETRIEVE2 (QUOTE SIGHTING) X))
(RETRIEVE3 Y (QUOTE IN-LANE) M)

(CASSERT Y (QUOTE INSIDE-A-MERCHANTLANE)
           (QUOTE DUMMY))
```

This rule shows several of the problems with the old forms. Not only is it hard to read, but the special items % and DUMMY were required by the triple nature of all assertions. Contrast this with the same rule in the new format:

```
(MERCHANTLANE *MLANE)
(PLATFORM *SHIP)
(*UNLESS* (MEDIUM *SHIP AIR))
(SIGHTING *SHIP *SIGHTING1)
(IN-LANE *MLANE *SIGHTING1)

(INSIDE-A-MERCHANTLANE *SIGHTING1)
```

With the exception of connectives (*UNLESS*, *NOT*, and *OR*), all of the conditions and actions look exactly like assertions. Where connectives do appear, the assertionlike forms are merely embedded, rather than obscured.

Any element that starts with "*" is a variable. Variables are bound on first encounter, and that binding is used in later conditions and the actions. This rule does not demonstrate an additional enhancement, which permits multiple actions, or conclusions, of a single rule.

As mentioned earlier, the assertion prettyprinter is used to generate rule descriptions so that even this improved rule format is not seen by the user. However, for system builders, it is necessary to use this format when constructing new rules.

2.3.2 Connectives

The new rule interpreter and rule formats permit a broader range of connectives than found in STAMMER. Here we'll give a description of each connective, along with some information about its use.

2.3.2.1 AND

AND is the implicit connective. Whenever conditions or actions have no explicit connective, AND is assumed. AND conditions succeed if they are retrieved with positive confidence. ANDed conclusions are all constructed.

2.3.2.2 *UNLESS*

UNLESS is the other holdover from STAMMER. *UNLESS* conditions succeed if there is no retrieval with positive confidence. Due to the unusual (and not fully understood) semantics of this connective, we have placed a restriction of one retrieval per *UNLESS* on the system.

2.3.2.3 *NOT*

NOT is an addition in STAMMER2. It behaves as one might expect, succeeding for all retrievals with negative confidence. *NOT* (and *UNLESS*) are not permitted in actions.

2.3.2.4 *OR*

OR is another addition. It has the effect of creating multiple rules, one for each argument. For each argument, if the argument succeeds (in the same sense as an AND argument), the remaining conditions of the rule are used. In actions, all the arguments to *OR* are constructed, but with different confidences than would be the case with an AND.

2.3.3 Streams and the redundant inference problem

2.3.3.1 Introduction

A major innovation in STAMMER2 is the central position that streams occupy in the interpreter. This is motivated by the continuing effort to improve the efficiency of rule-based systems, while maintaining their flexibility. Success in this enterprise is vital for satisfactory application to practical situations.

In addition to being useful from a practical standpoint, the application of streams has theoretical significance. A distinguishing feature of the STAMMER environment is the ongoing nature of data entry: messages are received in a continuous unsolicited fashion. Later messages may supersede or even negate earlier ones. This is in sharp contrast to other rule-based systems wherein all the information they will use is there from the beginning (or is obtained in response to a query). The spontaneous nature of the environment suggests a data-driven system. Streams were invented, in part, to meet the requirements of data-driven languages. Thus it is no surprise that the introduction of streams has led to a conceptual simplification of the STAMMER control mechanism.

2.3.3.2 Redundant inference

To see how streams improve efficiency, we first consider a simple model for the rule interpreter. In this model the system maintains a list of rules. The rule conditions pertain to a central data base which is constantly being updated. At regular intervals the system attempts to fire each rule; that is, it checks to see if the rule conditions are currently satisfied in the data base. We may suppose that the system can retrieve a list of current items in the data base matching a given condition. The rule interpreter can then be described by the following pseudo-ALGOL program:

```
PROCEDURE interp(condition-list actions)
if condition-list is empty do actions else
 for each x in list of matches of first
  condition do
   interp( m(x, remaining conditions), actions);
```

Here m is a function that modifies the remaining conditions on the basis of bindings determined by x. (Actually, a more efficient scheme for transmitting bindings would be used in a real interpreter.)

If one of many conditions is not satisfied, a rule may fail to fire. In this case, any partial work done in satisfying the rule is lost. The simple model also dictates that a rule is tried without regard to the nature of intervening changes to the data base. If such changes are irrelevant to a particular rule's conditions, attempting the rule leads to exactly the same work and the same result as the previous trial.

We might incorporate special mechanisms to deal with such extreme cases of wasted effort. The following example shows a more subtle instance, indicating that redundant computation pervades the simple model. Consider the rule (in coded form):

```
(sighting *x *y)(storm *z)(inside *x *z) -> (not (merchant *y))
```

This may be paraphrased as "sighting a ship in a storm suggests that it is not a merchant." Let us suppose that for a fixed ship we have m sightings and n storms. To try to fire the rule we must then do m*n computations of INSIDE. If we subsequently receive a new sighting, the rule is again applicable, and we now do (m+l)*n computations of INSIDE. However, only n of the combinations are new; the rest are redone and represent wasted effort. Clearly, only the new sighting should be used in combination with each of the recorded storms. Conversely, if a report of a new storm is received, only that storm should be tested in conjunction with the previous sightings. The problem, then, is to find a mechanism that will achieve this. In STAMMER a context restriction mechanism realized this economy in the example mentioned. However, it is still possible to construct examples wherein wasted effort occurs. Seeking a general solution to this problem led us to the use of streams.

It is important to recognize that the possibility of redundant computation arises only because the same rule can be tried more than once. In a situation wherein all the information the rules will need is present from the beginning, such repeated application is unnecessary and this kind of inefficiency will not occur. This point is the key to our solution.

2.3.3.3 Streams

Streams have been introduced in the development of advanced programming languages. Such experimental languages are radically different from the conventional languages in use today. LUCID[9], a nonprocedural language, is motivated by mathematical considerations--the desire for clean semantics and the facilitation of correctness proofs. Dataflow languages[10] have been constructed to take advantage of the coming prospect of multiprocessor machines. Both make use of streams in an explicit way.

A stream may be defined as a distinguished sequence of values, existing over time in a computation. If we execute a program in a conventional language, then the history of successive values of a variable forms a stream. Thus, in contrast to static data structures such as lists and arrays (where all the component elements must exist at one time), streams are DYNAMIC data structures. Although streams exist in all computations, they are not recognized as addressable objects by conventional languages and so cannot be manipulated to advantage. By identifying certain streams and providing stream operators, it is possible to achieve a natural form of parallelism in an otherwise sequential language. This allows simpler programming of what are, conceptually, interacting processes.

---

9 LUCID, A Nonprocedural Language with Iteration, by EA Ashcroft and WW Wadge; CACM, vol 20, 1977, p 519.

10 Technical Report 114A, An Asynchronous Programming Language and Computing Machine, by Arvind, KP Gostelow, and W Plouffe, Information and Computer Science Department, University of California, Irvine, 1978.

We have implemented streams in INTERLISP by providing three operators:

- o NEWSTREAM is a LISP function of no arguments that creates a stream. It returns a structure that serves as a means of addressing the stream. Thus rather than allowing access to existing implicit streams (which would require a way of intercepting the LISP interpreter), we provide for the construction of explicit streams which can then be used as desired. Rieger, in his use of "channels" (which have much in common with streams) takes the opposite approach[11].
- o MAPSTREAM (stream info fn) is a LISP function that sets up demons on streams. It is analogous to MAPC, except that it operates on streams instead of lists. When called, it immediately maps fn across whatever has been put in the stream so far. In addition, it sets up a "suspension" which is revived whenever anything new is put into the stream, causing fn to be called on the new stream element. Actually, fn must be a function of two arguments. The second argument is bound to info at each call, allowing a local environment to be saved and used with the stream.

- o PUTSTREAM (stream elt) is a LISP function used to put elements into a stream. It causes any suspensions attached to the stream to go into action with the new element immediately (unless a FREEZE is in effect--see below).

These are the primary operators for use on streams. For added control, we provide some additional functions. ENDSTREAM (stream) may be used to indicate that a stream can expect no more elements. This allows the suspensions associated with the stream to be thrown away, releasing space to the system. FREEZE is used to prevent (temporarily) suspensions from awakening while a number of additions are made to different streams in the system. UNFREEZE reverses this condition and activates a "catch up" procedure.

11

Spontaneous Computation and its Role in AI Modeling, by C Rieger; Pattern-Directed Inference Systems, DA Waterman and F Hayes-Roth, ed, Academic Press, 1978.

2.3.3.4 Streams and the rule interpreter

Streams are incorporated into the rule interpreter mechanism by representing the set of assertions that match a given rule condition as a stream. A simplified definition of the interpreter is given as follows:

```
PROCEDURE interp(condition-list actions)
if condition-list is empty do actions else
  for each x in stream matching first condition do
    interp( m(x, remaining conditions), actions);
```

The "for each" is implemented by a MAPSTREAM. Here INTERP is directly called once on each rule. This sets up the condition-matchers as demons which cause INTERP to be reactivated whenever a relevant item is placed in the data base. The effect is as though the item had been present when INTERP was originally called.

Comparing this interpreter to the earlier one, we see that the only difference is the replacement of a list by a stream. The significance of this is that a single application of INTERP to each rule is sufficient. The situation is formally identical to the one wherein all the information is present from the beginning; thus redundant computation of the type described will not occur.

The use of the stream mechanism has some additional implications for the interpreter. Suspensions are saved at the level of each condition of a rule. A new item of information may arouse a suspension and cause the interpreter to move through a few rule conditions before halting again. The effect is to create new suspensions farther down the rule. This exercise has done some of the work towards a potential future firing of the rule. Thus the stream regime causes deduction to proceed in an INCREMENTAL fashion, with as much use as possible made of new information, even if it doesn't result in a full firing of a rule. This is highly appropriate in the present task, where the system may have a considerable amount of idle time between significant occurrences. (It would also be useful in a chess-playing program, allowing information to be saved between moves.) A second implication is that rule selection in the conventional sense disappears, since each rule is activated exactly once. However, a suspension selection problem arises in its place. Fortunately, it is not difficult to determine exactly the suspensions which are relevant to a new data-base entry and to activate them through a suitable hashing scheme.

2.3.3.5 Pulsars

Since STAMMER deals with probabilistic rather than absolute deductions, an additional problem can arise. For purposes of economy, a potential deduction is prevented if the confidence in a particular condition falls below a threshold value , even though a matching item is in the data base. If the confidence should subsequently change, it may be necessary to revive the deduction at that point. Conceptually, each assertion in the stream of assertions has, in turn, a stream of confidences associated with it. For certain reasons, having to do with the way derivation trees are constructed, we do not want to use the full stream mechanism here. Instead a new streamlike entity called a "pulsar" is attached to each assertion node. Rather than having distinct elements placed on it, a pulsar is merely "pulsed." The effect of this is to revive suspensions saved on the pulsar. These may succeed or fail. If they succeed, they are removed from the pulsar. We use this mechanism as follows. If an assertion matches a rule condition but the deduction is blocked by a low confidence in the assertion, we save the state of the deduction in the pulsar associated with the assertion. If the confidence changes, the pulsar is pulsed and the deduction tries to proceed once more. When and if it succeeds, the suspension is discharged from the pulsar.

The following functions are provided to manipulate pulsars:

- o PULSAR creates a new pulsar.
- o SOMEPULSE (pulsar info pulsefn) saves a new suspension on the pulsar. pulsefn is a function of one argument which is bound to info when a pulse occurs. If it returns a non-nil value, it is discharged from the pulsar.
- o PULSE (pulsar) obvious.

2.4 Confidence

The increased flexibility in rule formats and interpretation has made it necessary to increase the power of the confidence calculating functions as well. Negative and unless conditions are now included in the calculation of confidence, where they were ignored before. More importantly, we have devised a method for dealing with cycles in the derivation process, which we are calling "semantic loops."

In STAMMER, confidence was calculated dynamically on request rather than being calculated and stored at the time of a derivation. This approach had the advantage of allowing the confidence to be updated automatically by later derivations without the need for recalculation unless it was specifically requested. The information required to enable the dynamic recalculation did not require additional storage, since it was needed for explanation.

The dynamic calculation of confidence was retained in STAMMER2, since the benefits of flexibility far outweigh the disadvantages of extra computation time. However, the structure of explanation and confidence information was changed slightly, to save space and increase the versatility of the system. Since back associations are handled differently in STAMMER2 and since no use was made of them in explanation, it was possible to reduce the complexity of the derivation tree by eliminating back pointers. It is now possible to go from any assertion to those rules and assertions which offer evidence for it, but it is not possible to find (directly) those assertions for which a particular rule or assertion is evidence. This simplification is matched by a slight complication: instead of being lumped together and ignored under NEGEV, assertions satisfying NOT and UNLESS conditions are stored in the tree with special markers which direct the computation mechanism.

These markers cause special operations to be performed. First the confidence of the affected assertion is calculated, then the special operations map that confidence into an appropriate changed confidence. Thus (NOT A0042) has confidence 0.7 when A0042 has confidence -0.7. The NOT operation is simple inversion. UNLESS is more complex, mapping the interval from -1.0 to 0.0 inclusive onto the single value 1.0 and mapping the interval from 1.0 to 0.0 onto 0.0 to -1.0 by the function x -> -x.

Including NOT and UNLESS in the calculation of confidence forces consideration of "semantic loops." What has until now been considered a derivation tree may in fact be a graph, complete with cycles. We wish neither to ignore such cycles completely nor to get caught in them forever. The solution is a marking algorithm that marks elements of the tree (graph), and considers contributions only from previously unmarked elements. Thus, each element of the graph contributes at most once in any branch. After confidence calculation, the marks are removed. The marks are also removed in case of an error exit, to avoid any possible complication. This algorithm permits the use of rules that might ordinarily be considered "circular," such as (Sibling *x *y)->(Sibling *y *x).

2.5 Oracles

Oracles are distinguished relations which do not rely solely on the data base and standard retrieval. They provide an escape to LISP code for certain relations that are either difficult or tedious to represent as rules and for relations that should only be calculated on demand. Since oracles are a way of incorporating LISP function calls, they lose the advantage of derived assertions in explanation.

In the original STAMMER, oracles examined their arguments and returned T or NIL, depending on whether or not the given relation held among the arguments. The oracle call was then treated as an assertion and added to memory, with a confidence based on the value returned.

All of this still happens in STAMMER2. However, other things happen as well. Due to changes in the rule interpreter, the arguments to relations are no longer evaluated. Thus it is no longer possible to use calls to LISP functions (other than oracles) to generate arguments to relations in conditions. Typical uses of this ability previously were calculation of course and speed. With the evaluation gone, some other way of generating the arguments was required.

An enhancement in STAMMER2 permits certain distinguished oracles to return answers other than T or NIL and thus to bind arguments to relations. If all the arguments to one of these distinguished oracles are bound when the oracle is called, the effect is the same as a normal oracle.

These distinguished oracles (called lastarg oracles) are LISP functions just as are standard oracles. However, the functions that implement them take one fewer arguments than would appear from the call, then calculate and return an answer which is the correct (confidence 1.0) binding for the last argument in the call. If the last argument was unbound at the time of the call, the function call with the returned value replacing the unbound argument is asserted in memory with absolute positive confidence. If the last argument was bound at the time of call, the function call is asserted with confidence 1.0 if the value returned is equal to the binding, and -1.0 if they differ.

Oracles are distinguished by a marker on their property lists. Lastarg oracles are marked both as oracles and as lastarg type.

2.6 The Explanation System

2.6.1 A production-driven user interaction facility

QH is a subsystem to assist in programming a comfortable interaction with a user. The services provided the user are modelled after those of ASKUSER, with its shortcomings corrected and a few improvements made. To the programmer, QH offers a clean and convenient way of specifying the interaction: by means of productions. These productions also direct the processing of the user input.

In the past there have been several other methods for implementing question-and-answer subsystems. For completely arbitrary input one would need some technique from natural language processing, such as the ATN as used by Woods in the LUNAR program[12,13], but this is computationally expensive and often leads to questions which the system is unable to decipher. The standard way of implementing user interaction is to give him a fixed menu of responses. This is computationally efficient, but very rigid. The production system approach guides the user into forming questions which the system can understand. The options that the user has can be dynamically computed. This gives the user flexibility while at the same time providing the programmer with some capability for context-sensitive parsing.

---

12 Transition Network Grammars for Natural Language Analysis, by WA Woods; CACM, vol 13, 1970, p 591-606.

13 BBN Report 2378, The LUNAR Sciences Natural Language Information System: Final Report, by WA Woods, RM Kaplan, and BL Nash-Webber; Bolt, Beranek and Newman, Inc, 1972.

2.7 Position Functions

The functions in the file PLATPOS determine the position of a sighting at an arbitrary time. If the position of two sightings of the same platform is known with certainty, the position at other times is determined by a simple extrapolation or interpolation. More likely, at least one of the positions is given only as a polygon. How may one interpolate or extrapolate polygons?

If p1 and p2 are two polygons known at times t1 and t2 respectively, then one way to interpolate them would be to union the interpolations formed by taking one point from p1 and one point from p2 for all such points. In the case that p1 and p2 are degenerate polygons, ie points, this yields the correct result. In the general case, this computation is very expensive. In PLATPOS a computationally efficient method is used which gives an approximation to the technique just described. In the method adopted, if p1 and p2 are the two polygons, a "span" is constructed. If the length of p1 is greater than or equal to p2, then for each vertex of p1, find the nearest vertex in p2. This set of pairs of vertices is called the span. Each pair of vertices defines a line segment on which a standard interpolation can be performed. The collection of such interpolated points defines a polygon, which is, by fiat, the polygon that PLATPOS creates from p1 and p2.

2.8 The Message Monitor and the LWES Connection

One of the greatest inconveniences of the original STAMMER was the difficulty in creating scenarios. Since scenarios were represented by messages containing location and (occassionally) identification information, these messages had to be created to establish a new scenario. Unfortunately, there were no facilities to aid in the creation of these reports, which made scenario generation an extremely time-consuming cut-and-try hand operation, with the further problem that there was no assurance that the resulting scenarios were reasonable.

Since the completion of STAMMER, a LISP version of the Warfare Environment Simulator (LWES), written by Frank Zydbel, has become available. LWES permits the user to give commands to various platforms, causing them to move about, sense each other, and attack. Since LWES simulates sensor operation to provide reports to the user, it was easily modifiable to produce a file of sensor reports for STAMMER2.

In addition to the radar reports dealt with in STAMMER, LWES generates sonar and EW reports, and this is an added advantage. The format of the sensor report file written by LWES is detailed in appendix III.

To utilize the sensor report file written by LWES, a new message monitor (receipt function) was written. This new monitor "knows" about the format of sensor report file entries. It reads the reports and places appropriate assertions in memory. The message monitor can also deal with reports of own-ship location change and weather reports (in the restricted sense of storm warnings).

CHAPTER 3
USER'S HANDBOOK

3.1 Sample Run

```
_(STAMMER)
 Welcome to version 2 of the STAMMER TSA system.
What file would you like to take messages from?
(Default is SCENE.ICE): TEST2.DAT


Are you running on a Tektronix?No
Do you have a Tektronix available for display? No

     RADAR contact at (63.67 -24.17)  Time: 115
     Associated with track CONTACT2

Report: CONTACT2 was sighted in the merchant lane LANE2
A0223: CONTACT2 is somewhat unlikely to be (-.19) a MERCHANT
Question?  WHY is A0223
STAMMER applied the rule(s)
CLOSE-POPUP
Question?  HOW does rule CLOSE-POPUP apply to  A0223
The rule was applied with the assertions

A0215: CONTACT2 is a contact

A0214: SIGHTING3 is definitely (.99) the first sighting of
             CONTACT2

A0220: 11.73514 is the range of SIGHTING3

A0222: 11.73514 is less than 12

Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.74 -24.25)  Time: 125
     Associated with track CONTACT2

Report: CONTACT2 was sighted in the merchant lane LANE2
A0223: CONTACT2 is somewhat unlikely to be (-.39) a MERCHANT
Question?  WHY is A0223
STAMMER applied the rule(s)
FASTER-THAN-A-MERCHANT CLOSE-POPUP
Question?  HOW does rule FASTER-THAN-A-MERCHANT apply to
              A0223
```

```
The rule was applied with the assertions

A0215: CONTACT2 is a contact

A0239: SIGHTING4 is a sighting of CONTACT2

A0244: SIGHTING4 is other than a first sighting of its
              platform.

A0247: 28.26315 is the speed of SIGHTING4

A0257: 28.26315 is greater than 25

Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.78 -24.29)  Time: 130
     Associated with track CONTACT2

A0223: CONTACT2 is probably not (-.68) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.81 -24.32)  Time: 135
     Associated with track CONTACT2

A0223: CONTACT2 is very probably not (-.79) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.95 -24.47)  Time: 155
     Associated with track CONTACT2

A0223: CONTACT2 is almost certainly not (-.89) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.99 -24.46)  Time: 160
     Associated with track CONTACT2

A0223: CONTACT2 is almost certainly not (-.95) a MERCHANT
Question?  WHY is A0223
STAMMER applied the rule(s)
SPEED-CHANGED FASTER-THAN-A-MERCHANT OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT SPEED-CHANGED OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT SPEED-CHANGED OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT FASTER-THAN-A-MERCHANT
CLOSE-POPUP
Question?  HOW does rule SPEED-CHANGED apply to  A0223
```

```
Which occurrence? 2
The rule was applied with the assertions

A0215: CONTACT2 is a contact

A0300: SIGHTING6 is a sighting of CONTACT2

A0309: SIGHTING6 is other than a first sighting of its
               platform.

A0315: SIGHTING6 is other than a last sighting of its
                platform.

A0373: SIGHTING7 is the successor (in time) of SIGHTING6

A0313: 24.15066 is the speed of SIGHTING6

A0354: 27.86787 is the speed of SIGHTING7

A0375: 27.86787 is not roughly the same speed as 24.15066

Question?  HOW does rule OUTSIDE-ALL-LANES apply to  A0223
Which occurrence? 2
The rule was applied with the assertions

A0226: CONTACT2 is a platform

A0231: CONTACT2 is not known to be AIR

A0338: SIGHTING7 is a sighting of CONTACT2

A0369: SIGHTING7 is not known to be inside a merchantlane

Question?  WHY is A0369
Assertion based on the absence of information
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (64.06 -24.45)  Time: 170
     Associated with track CONTACT2

A0223: CONTACT2 is definitely not (-.97) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (64.13 -24.44)  Time: 180
     Associated with track CONTACT2

A0223: CONTACT2 is definitely not (-.98) a MERCHANT
Question?  WHY is A0223
```

```
STAMMER applied the rule(s)
SPEED-CHANGED FASTER-THAN-A-MERCHANT OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT SPEED-CHANGED OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT SPEED-CHANGED FASTER-THAN-A-MERCHANT
OUTSIDE-ALL-LANES FASTER-THAN-A-MERCHANT SPEED-CHANGED
OUTSIDE-ALL-LANES FASTER-THAN-A-MERCHANT SPEED-CHANGED
OUTSIDE-ALL-LANES FASTER-THAN-A-MERCHANT OUTSIDE-ALL-LANES
FASTER-THAN-A-MERCHANT FASTER-THAN-A-MERCHANT CLOSE-POPUP
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (64.16 -24.43)  Time: 185
     Associated with track CONTACT2

A0223: CONTACT2 is definitely not (-.99) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (64.24 -24.41)  Time: 200
     Associated with track CONTACT2

A0223: CONTACT2 is definitely not (-.99) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (64.35 -24.39)  Time: 220
     Associated with track CONTACT2

A0223: CONTACT2 is definitely not (-.99) a MERCHANT
Question?  Quit
Leaving EXPLAIN
 Thank you for your interest in the STAMMER system.
```

```
_(STAMMER)
 Welcome to version 2 of the STAMMER TSA system.
What file would you like to take messages from?
(Default is SCENE.ICE): TEST7.DAT


Are you running on a Tektronix?No
Do you have a Tektronix available for display? No

     SONAR contact at (63.75 -23.95)  Time: 0
     Associated with track CONTACT1

Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -23.95)  Time: 0
     Associated with track CONTACT1

Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  WHERE IS CONTACT1
We have a sighting of the platform.
((63.75 -23.95))
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.09)  Time: 15
     Associated with track CONTACT1

Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -24.09)  Time: 15
     Associated with track CONTACT1

A0300: CONTACT1 is somewhat likely to be (.15) a PATROL
A0299: CONTACT1 is somewhat likely to be (.15) a FISHING
Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  WHY is A0300
STAMMER applied the rule(s)
SMALL-CRAFT6
Question?  HOW does rule SMALL-CRAFT6 apply to  A0300
The rule was applied with the assertions

A0237: CONTACT1 is a contact

A0283: SIGHTING6 is a sighting of CONTACT1
```

```
A0289: SIGHTING6 is other than a first sighting of its
               platform.

A0297: 11.27595 is the range of SIGHTING6

A0275: 11.27595 is less than 16

A0276: 11.27595 is greater than 9

A0282: Signal at SIGHTING6 is WEAK

A0291: 14.82488 is the speed of SIGHTING6

A0298: 14.82488 is not greater than 20

Question?  WHERE IS CONTACT1
We have a sighting of the platform.
((63.75 -24.09))
Question?  WHERE WAS CONTACT1 at time 10

Estimated from the sightings SIGHTING4 and SIGHTING5
((63.75 -24.04333))
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.14)  Time: 20
     Associated with track CONTACT1

A0300: CONTACT1 is somewhat likely to be (.28) a PATROL
A0299: CONTACT1 is somewhat likely to be (.28) a FISHING
Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -24.14)  Time: 20
     Associated with track CONTACT1

A0300: CONTACT1 is somewhat likely to be (.39) a PATROL
A0299: CONTACT1 is somewhat likely to be (.39) a FISHING
Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.19)  Time: 25
     Associated with track CONTACT1

A0301: (MODE *X SURFACE) is probably (.48)  a (TYPE *X SUB).
A0300: CONTACT1 is probably (.48) a PATROL
A0299: CONTACT1 is probably (.48) a FISHING
Report: CONTACT1 was sighted in the merchant lane LANE2
```

```
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -24.19)  Time: 25
     Associated with track CONTACT1

Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.24)  Time: 30
     Associated with track CONTACT1

Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -24.24)  Time: 30
     Associated with track CONTACT1

Report: CONTACT1 was sighted in the merchant lane LANE2
Question?  WHAT IS CONTACT1
probably PATROL
probably FISHING
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.33)  Time: 40
     Associated with track CONTACT1

A0570: CONTACT1 is probably not (-.54) a MERCHANT
Question?  WHY is A0570
STAMMER applied the rule(s)
SPEED-CHANGED SPEED-CHANGED OUTSIDE-ALL-LANES
Question?  HOW does rule SPEED-CHANGED apply to  A0570
Which occurrence? 1
The rule was applied with the assertions

A0237: CONTACT1 is a contact

A0504: SIGHTING12 is a sighting of CONTACT1

A0516: SIGHTING12 is other than a first sighting of its
                platform.

A0519: SIGHTING12 is other than a last sighting of its
                 platform.

A0588: SIGHTING13 is the successor (in time) of SIGHTING12
```

```
A0518: 15.10729 is the speed of SIGHTING12

A0561: 14.24329 is the speed of SIGHTING13

A0586: 14.24329 is not roughly the same speed as 15.10729

Question?  HOW does rule SPEED-CHANGED apply to  A0570
Which occurrence? 2
The rule was applied with the assertions

A0237: CONTACT1 is a contact

A0455: SIGHTING11 is a sighting of CONTACT1

A0470: SIGHTING11 is other than a first sighting of its
                 platform.

A0477: SIGHTING11 is other than a last sighting of its
                platform.

A0585: SIGHTING13 is the successor (in time) of SIGHTING11

A0475: 15.10729 is the speed of SIGHTING11

A0561: 14.24329 is the speed of SIGHTING13

A0586: 14.24329 is not roughly the same speed as 15.10729

Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -24.33)  Time: 40
     Associated with track CONTACT1

A0570: CONTACT1 is probably not (-.58) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.42)  Time: 50
     Associated with track CONTACT1

A0570: CONTACT1 is very probably not (-.8) a MERCHANT
Question?  Quit
Leaving EXPLAIN

     RADAR contact at (63.75 -24.42)  Time: 50
     Associated with track CONTACT1

A0570: CONTACT1 is very probably not (-.82) a MERCHANT
Question?  WHAT IS CONTACT1
```

```
very probably not MERCHANT
probably PATROL
probably FISHING
Question?  Quit
Leaving EXPLAIN

     SONAR contact at (63.75 -24.52)  Time: 60
     Associated with track CONTACT1

A0570: CONTACT1 is almost certainly not (-.96) a MERCHANT
Question?  Quit
Leaving EXPLAIN
```

3.2 User's Guide to STAMMER2

To run the STAMMER2 system, all you need do is type STAMMER2 at the monitor level. This will automatically load all the programs and data for the STAMMER2 system. As the programs are loaded, instructions on how to continue will be printed. Prompted by the system, the user is asked to type (STAMMER) to start the processing of messages.

Once you have started execution of the STAMMER2 system (as opposed to merely loading it), you will be asked several questions about files and the characteristics of the terminal you're at. Questions about files can be answered with either a file name or a carriage return. All questions about terminals and related information are to be answered Y (for yes) or N (for no).

After messages and reports are received, you will be prompted for input by the explanation system. Instructions on its use are contained in the next section.

After completing a STAMMER run, or at any BREAK (see next section), you may leave STAMMER2 by typing (LOGOUT). After a run is over, you may type (EXPLAIN) to reenter the explanation system. You may leave STAMMER2 at any point by typing two control-Cs (^C).

3.3 User's Guide to the Explanation System

STAMMER's explanation system, written within the QH production system, requires about two dozen productions. EXPLAIN allows you to ask about the contents of the data base (memory) and how items were placed in the data base. You may trace the derivation of any conclusion. When a display appears, you may use any of the DSPLA function key commands[14] to manipulate the picture.

Interactions with STAMMER2 fall into two categories: commands and queries. Commands allow the user to change STAMMER's data base. Queries allow the user to inquire about the data base and about how STAMMER2 reasoned.

A summary of the commands is as follows:

1. SAVE memory <filename>
2. NEW rule <ruledefinition>
3. CHANGE confidence in the rule <rulename>
4. Quit
5. RECAP
6. DISPLAY
7. HELP
8. BREAK

A more complete description of the commands follows:

SAVE

Format: SAVE <filename>
example: SAVE scene.mem
This command allows the user to save the current state of the data base on a file of his own choosing.

NEW

Format: NEW rule
This command begins an interactive session in which the user is led through the definition of a new rule. During this interaction the user is asked about the rule name, its firing conditions, its actions, and its confidence.

CHANGE

Format: CHANGE confidence in the rule

---

14 NOSC TN 530, DSPLA--A Graphics Package for Tactical Situation Assessment (Version II), by GG McIntyre, 15 September 1978. NOSC TNs are informal documents intended chiefly for internal use.

<rulename>
The CHANGE command allows the user to modify the confidence of a selected rule. The conditions and actions may not be changed.

Q (The system completes this command to form QUIT.)
This command allows the user to exit the explain system.

RECAP
Format: RECAP
This command gives a summary of all the conclusions that STAMMER has reached.

DISPLAY
Format: DISPLAY
This command puts the user into graphic mode. All the commands of DSPLA are now available to him. To leave this mode the user types a Q.

HELP
Format: HELP
This gives the user a brief description of the commands and queries that are available to him.

BREAK
Format: BREAK
This command puts the experienced user into a BREAK and allows him to execute any LISP function or examine system variables.

A summary of the queries is as follows:

1. What ...
2. Is ...
3. TELL me about ...
4. WHERE ...
5. WHY is ...
6. HOW does <RULE> apply to <ASSERTION>
7. WHOSE ...
8. WHO ...

A more complete description of the queries follows:

WHAT
Format: WHAT is (THE AN A) <RELATION> (OF) <ITEM>
Example: WHAT is the COURSE of SIGHTING3

Example: WHAT is CONTACT1
This query provides one of several question formats for asking about entries in the data base.

IS

Formats: IS (THE AN A) <RELATION> (OF) <ITEM> <ITEM>
IS (THE) <ITEM> (A AN THE) <RELATION> (OF) <ITEM>
Example: Is RADAR the SOURCE of SIGHTING32
This question form allows the user to ask about specific entries in the data base.

TELL me about

Format: TELL me about <anything>
Example: TELL me about SIGHTING5
This is the most flexible query, and allows the user to ask rules, categories, relations, or specific facts.

WHERE

Formats: WHERE is <OBJECT>
WHERE was <OBJECT> at <TIME>
Example: WHERE is CONTACT7
With this command the user can ask about the position of a platform, merchant lane, or storm. The position may be asked for any time, in the future or the past. The answer is usually a polygon unless the position is known with certainty. If no time is given, STAMMER2 assumes that the current time is the time involved.

WHY

Format: WHY is <ASSERTION>
Example: WHY is A00345
With this command the user can find out the primary or immediate reasons that STAMMER2 used to conclude any assertion. He will be given all the rules involved in the decision.

HOW

Format: HOW does rule <RULE> apply to <ASSERTION>
Example: HOW does rule ID-LANE apply to A0034
This query allows the user to find out what assertions or facts were involved in

permitting the rule to help conclude the given assertion. If the rule is applied in more than one way, the user will be asked to specify which occurrence he would like to examine.

WHOSE

Format: WHOSE <RELATION> is <ITEM>
Example: WHOSE TYPE is MERCHANT
This query acts as a partial inverse to the WHAT query. It may return more than one object.

WHO

Format: WHO is (THE AN A) <RELATION> (OF) <ITEM>
Example: WHO is INSIDE STORM3
Example: WHO is HOSTILE
This provides another form for inquiring the data base that is similar to the WHOSE query.

<u>Special user inputs</u>

In addition to the standard user inputs, there are a number of special user inputs. These include the following:

- o rubout: deletes the current field entry.
- o ? : gives all options for the field.
- o ^E: deletes the entire line input.
- o <escape>: completes current input if only one completion is possible.
- o &VAR: gives all possible values for VAR.

3.4 Programmer's Guide to QH

The interaction is broken up into segments. The segments are described by productions with the segment name on the left-hand side and a sequence of fields on the right. The productions resemble those of BNF and are used in a similar way to guide the parsing of the user input. However, user input is solicited in an ongoing fashion, as the parsing proceeds. If a string is encountered in the production, it is treated as a prompting message to be printed to the user.

At the end of the production there is generally a LISP expression, indicating what is to be returned as the value of the interaction segment. This corresponds to the semantic routine of a compiler.

The fields in a production may be filled in a number of ways:

- o ATOM -- with the exceptions below this is treated as a single key to be matched by user input.
- o STRING -- a message to printed to the user. An initial - in the string will cause a backspace.
- o <ATOM> -- a new interaction segment to be called recursively.
- o (= X) -- binds X to the value of the last field encountered.
- o (f1 f2 ...) -- A sort of OR expression. The system tries to match any one of the fields f1, f2, ... The value of the first one matched is returned as the value of the whole field.
- o (~ f1 f2 ...) -- an OR of optional fields, one of which MAY be matched before continuing.
- o !ATOM -- indicates that the value of ATOM is to be used instead as the current field. To the user, this option is described (in response to a question mark) as: a ATOM.
- o : -- indicates that the next field is the LISP expression to be evaluated and is the last field. If this is not present then the value of the last field is returned as the value of the segment.
- o <esc> -- indicates that the next field will simply be read by the function (READ). There should be no other alternative field.

The QH user interaction system generator also permits a number of single control character inputs from the user. If the user types an <esc> and there is only one way of completing his current input, the completion will take place automatically. If the user types "?", the available options will be given. If a !ATOM is an option, the user can find out the value of ATOM by typing &ATOM. A <rubout> by the user will delete the current field input, while a "control

E" will abort the entire line input.

A list of the right-hand sides of the productions describing a particular interaction segment is stored on the property list of the segment name, under the property QHPRODS. The function QHTAKE is an NLAMBDA nospread which can be given an initial field sequence.

```
    Define <MEXP> by:

    PUTPROP(<MEXP> QHPRODS (

    (MARY)

    (THE (NAME BOOK SISTER) (= X) "of" <MEXP> (= Y) :
       (LIST  X Y]

    Then (QHTAKE "What?" <MEXP> will accept:

    MARY ,returning MARY;  or:

    THE BOOK of MARY ,returning (BOOK MARY);  or:

    THE NAME of THE SISTER  of  MARY  ,returning  (NAME
      (SISTER MARY)) ,etc.
```

Notes. Left recursion, direct or indirect, is not allowed. If several productions are applicable in which the same key occurs, the later occurrence supersedes the earlier in determining which production is followed subsequently. EXAMPLE: If <MEXP> is defined by PUTPROP(<MEXP> QHPRODS (

```
(MARY :  (PRINT 'HI))

(MARY :  (PRINT 'BYE]
```

then only the second production is called when the user types MARY.

3.5 Programmer's Guide to the Assertion Prettyprinter

ıhe assertion prettyprinting mechanism (PRETTYASSR) is, to a certain extent, under the control of sophisticated users. PRETTYASSR prints assertions in a format determined by a canned instruction set stored on the relation name of the assertion. Such canned instructions are called PRINFORMS.

Each relation has a list of prinforms stored on its property list under the name PRINFORMS. The length of this list is unrestricted, and the list may be empty. If the list is empty, a default format is used, which depends on the length of the assertion.

Each prinform is in the following format:

```
<prinform> := (<prinspec>+)

<prinspec> := <string> | <number> | <fn call> | T
```

and is interpreted as follows:

- o strings are printed directly
- o function calls are evaluated (and, it is assumed, do some printing)
- o numbers cause the (CAR (NTH assertion number)) to be printed (eg 1 would cause the relation of an assertion to be printed)
- o T causes a carriage return line feed pair

So to print the assertion (CONTACT RED1) as "RED1 is a contact", the prinform would be (2 " is a contact" T). (This prinform would not actually cause the double quotes to be printed, but would force a new line at the end.)

PRETTYASSR takes three arguments: an assertion node or a list in assertion form, a prinform selector, and a confidence override. Only the first argument is required. The second argument selects which of the (possibly multiple) prinforms stored on the relation is to be used. If this argument is missing, or exceeds the number of prinforms available, the first prinform is selected as a default. The third argument permits an override on the confidence printed for the assertion. If the third argument is missing, the confidence that is dynamically calculated for the assertion is used. If an argument is given, the assertion is printed as if the override confidence were the confidence in the assertion.

3.6 LWES and Modifications

LWES is an INTERLISP implementation of the Warfare Effectiveness Simulator (WES), written by Frank Zydbel. This section is intended to serve as a user's guide to LWES, including the modifications made since the original implementation to utilize the simulator capabilities as a scenario and data generator.

3.6.1 Accessing LWES

LWES is written in and runs under INTERLISP. To use LWES, it is necessary first to invoke LISP, then to load the programs that make up LWES. To invoke LISP, type (at the monitor level)

```
LISP
```

The LISP prompt is an underscore (left arrow on some terminals). To load the LWES programs once in LISP, type

```
LOAD(LWES.LOAD)
```

No carriage return is necessary. As the files are loaded, messages will be printed at the terminal. A list of the necessary files appears later in this report.

3.6.2 Using LWES

LWES does not have the BUILD capability of the original WES. Instead, a file containing information about selected units is loaded as the default force for use in simulation. These default units are grouped into two sides, Blue and Orange. You may suppress reports from either Blue platforms or Orange platforms by typing

```
(SETQ BLUEFLG NIL) or

(SETQ ORANGEFLG NIL)
```

respectively, before invoking LWES. Typically, one wishes to suppress reports from Orange platforms, to more nearly simulate true conditions.

To invoke LWES, type

```
(STARTWES)
```

The system will respond by telling you about available units and then allowing you to select some or all of these units for use in your simulation. You will be queried as to the inclusion of units and the initial values for attributes of units you select. "Yes/no" questions can be answered with a Y or N as appropriate. You should give latitude, longitude, and course in degrees, speed in knots, and altitude in feet.

The selection of a home ship is required for the construction of a sensor report file. Only one platform may be selected as home ship. Once a home ship is selected, you will not be asked again.

After selecting participating units, you will reach the command level of LWES, indicated by a two-asterisk (**) prompt. At this point, you may give orders to the units you have selected, or commands to the simulation mechanism. The following description of the options at this point is drawn primarily from documentation of the original LWES.

At LWES command level, you may type any input described in the following BNF:

```
<input> := <orderpackage> | <simcmd>
```

3.6.2.1 Simulator commands

```
<simcmd> := <stopcmd> | <timecmd> | <plotcmd> |
            <gocmd>
```

```
<stopcmd> := <stoptoken> <cr>
<stoptoken> := STOP | END | Q | QUIT | EXIT | OK
```

STOP halts the simulation and leaves LWES.

```
<timecmd> := <timetoken> (<sp> <posint>) <cr>
<timetoken> := TIME | CLOCK | T
```

TIME is used to find out the present simulated time and to change the present simulated time. With no argument, TIME will return the current time. If a numeric argument is given, the simulation clock will be advanced to that time. Time does not move backward.

```
<plotcmd> := PLOT <cr> | DRAW <cr> | PL <cr>
```

PLOT draws a rough approximation of the current situation on the terminal.

```
<gocmd> := <gotoken> (<sp> <posint>) <cr>
<gotoken> := GO | RUN | MOVE | SIM
```

GO causes the simulation to be advanced until the simulator clock equals the numeric argument given to GO. If the argument is left out, the simulation advances one clock tick.

3.6.2.2 Orders to platforms

```
<orderpackage> := <addrline> <order>+ <timespec>
<addrline> := TO <addrlist> | FOR <addrlist> |
              : <addrlist>
<addrlist> := <platform/name> (<sp>
                   <platform/name>)* <cr>
<timespec> := <cr> | <posint> (<sp> <posint>) <cr>
<posint> := 1 | 2 | 3 | 4 |...
```

Orders are addressed to one or more platforms. After starting an order package, you must complete it. Order packages must include at least one order. The timespec specifies when the order is to be executed. An empty timespec calls for immediate execution. A single number as a timespec specifies when execution of the order is to begin. If two numbers are given in the timespec, they are taken to be the starting and ending time, respectively, of the order package.

```
<order> := <killord> | <reportord> | <EWord> |
          <courseord> | <speedord> | <altord> |
          <attackord>
```

```
<killord> := <killtoken> (<sp> <posint>)
                 (<sp> <posint>) <cr>
<killtoken> := KILL | CANCEL | FORGET
```

The KILL order tells the addressed platforms to forget previously issued orders that become effective at certain times. The numbers given to KILL function just like the posints in timespecs.

```
<reportord> := <reportoken> <sp> <reporttype> <cr>
<reportoken> := REPORT | RPT | R | SEARCH
<reporttype> := SELF | SUB | SHIP | PLANE | BASE |
                 ALL
```

The REPORT order causes the addressed platforms to turn on their appropriate sensors (eg SONAR for SUB) at the specified time to detect non-own-force platforms of the specified type. When such detections are made, a report will be printed at the terminal (assuming that such reports have not been suppressed). REPORT SELF causes a status (damage) report to be issued.

```
<EWord> := EW <cr> | PASSIVE <cr> | PASS <cr> |
             E <cr>
```

The EW order causes the addressed platform to make reports about passive sensor (intercept, acoustic, passive sonar) detections.

```
<courseord> := <coursetoken> <sp> <bearing> <cr>
<coursetoken> := COURSE | HEADING | COMPASS | HDG |
                 CRS | BRNG | H | C | B
<bearing> := 0 | 1 | 2 | ... | 360
```

The COURSE order causes the addressed platform to change its course to be bearing. 0 and 360 are true north.

```
<speedord> := <speedtoken> <sp> <posint> <cr>
<speedtoken> := SPEED | SPD | S | KNOTS
```

SPEED causes a change in the platform's speed until it matches posint.

```
<altord> := <altoken> <sp> <posint> <cr>
<altoken> := ALTITUDE | ANGELS | ALT
```

ALTITUDE causes a change in the altitude of an aircraft to match the value given. Aircraft that go below 12 feet crash.

All the value-changing commands (COURSE, SPEED, and ALTITUDE) use models of the maneuverability of the platform to cause the change over time. For example, a ship may be able to change course by only one degree per second.

```
<attackord> := <attacktoken> <sp> <attacktype>
               <sp> <condition> <sp> <weapon>
               <sp> <posint> <cr>
<attacktoken> := ATTACK | ATT | A | KIT | ZAP
<attacktype> := SUB | SHIP | PLANE | BASE | ALL
<condition> := NOW | ON CONTACT | IF ATTACKED
<weapon> := the name of a weapon carried by
                the addressed unit
```

ATTACK causes the addressed platforms to attack the nearest non-own-force platform of the specified type with posint rounds of the named weapon when the condition is met. So that the platform will be able to tell where a suitable victim is, sensors are turned on if they are not already active. (This has the effect of issuing a report command for the attacktype.)

Several notes are in order.

1. If you give a single time in a timespec, that time is taken as both the starting and ending time for the order. For example:

```
** TO CONNOLE
** REPORT ALL
** 5
```

   would turn on all sensors (and report) only at time 5. No sensing or reports would take place after time 5.

2. Sensing reports are generated (at present) only on

nonfriendly units. Sensors are "blind" to own force units.

3. EW detects prop noise and con detect emissions from only those units with sensors turned on. Sensors can only be turned on by the REPORT and ATTACK orders.

4. Some consistency checks are not made until the time of order execution. For example, you could order an attack on planes using a torpedo as weapon, and the impossibility would not be discovered until the attack was attempted.

5. Stopping the simulation stops everything -- it is not possible to resume.

6. The TIME command, if used to reset the clock, DOES NOT advance the simulation. Nothing happens except that the value of the clock is changed.

3.6.3 Modifications to Zydbel's original implementation

There have been several modifications and enhancements to the original LWES. Most of these have been with the purpose of using LWES as a scenario and data generator for other systems.

Originally, LWES simply accepted the forces described in data files or laboriously entered by hand at the terminal and used them in its simulation. This required that either the file be edited or time-consuming hand coding be done to change even the initial position of any unit. An initialization routine has been added which permits the user to construct a simulation using only those units and initial values which he desires (within certain limits). Unit selection is still limited to those units defined in the data file (there is still no BUILD capability), but changing the canned scenario is much easier.

In an attempt to reduce computational expense, LWES did not update the location of all units on every simulator clock tick. This resulted in reports that magnified the discrete nature of the simulation. A modification has insured that unit locations are updated every clock tick.

While units were always distinguished by side, no use was made of the information during report generation. All reports were dumped to the terminal without regard to order. This has been modified so that reports are sorted by side before printing. Additionally, reports from either side may be suppressed. To use LWES as a scenario and data generator for other systems, enhancements were added that create files based on the location of units and reports generated. Two files are created. The first contains the type and location of every active unit at every clock tick and is named by the variable GTHFILE. The second, containing sensor reports (both active and passive) is named by SENSORFILE. While these files have default names, the name of the files created may be changed by resetting the appropriate variable. Formats for these files are described later.

CHAPTER 4
DIRECTIONS FOR FUTURE WORK

The STAMMER2 project represents work on a novel AI problem -- that of making incremental deductions in realtime based on information which may be incomplete or out of chronological order. The demand for real-time performance is not for user convenience but a fact of the environment in which it is hoped that STAMMER2 is to be used. Work on STAMMER2 has highlighted a number of issues in AI which have not yet been carefully addressed. These include the following:

- Memory
  1. In order to achieve fast access to the data base, STAMMER2 employs a hashing scheme which entails redundant storage of information. New results published in a recent JACM article[15] show how to remove some of this redundancy. This would yield a space savings at a slight cost in speed.
  2. Another problem that several AI workers have considered without much success is that of aging and forgetting information. Currently STAMMER2 keeps all the history information in order to generate appropriate explanation. Some space could be saved by judicious forgetting.
- Control
  STAMMER2 has been designed as a forward-chaining production system. To increase the time efficiency of the system, a backward-chaining system sometimes should be used. Typically AI production systems are either backward or forward chaining. It is possible to have a system which both forward and backward chains. One way of implementing such a hybrid control would be to allow "goals" to occur in the conditions and actions of rules. This permits the backward chaining to be simulated by the forward-chaining component.

---
15 Optimality of Multiple-Key Hashing Functions, by A Bolour; JACM, vol 26, 1979.

- o Simulation
  The creation of scenarios for STAMMER2 is a time-consuming, laborious task. Another approach, which would be be an entire project in itself, is to have a problem-solving simulation generator that behaves similarly to TALESPIN, a program written by Jim Meehan of UCI[16]. In such a simulator, one would give the initial positions and goals of the craft and the program would generate the appropriate messages and actions.
- o Learning
  The main difficulty with building STAMMER2 has been the lack of expert advice or expert rules. One reason for this lack is that there are apparently no experts! If we were given instances in which human operators failed to draw the proper conclusions, however, then a system could be built, perhaps modeling DENDRAL, which would induce the necessary rules. Another area where learning could be applied would be in allowing the system to learn what it should forget. Forgetting and learning are closely linked problems.

- o Confidence
  1. Neither MYCIN nor PROSPECTOR deal with the problem of assigning confidences to constituents of an "or" conclusion. Since backward-chaining systems can avoid this problem by setting up subgoals, one solution for a forward-chaining system is to allow some back chaining.
  2. STAMMER2 introduced an "unless" logical operator and gave it semantics (confidence manipulating functions). The semantics need to be extended to allow multiple arguments.

---
16 The Metanovel: Telling Stories by Computer, by JR Meehan, PhD Thesis, Yale University, December 1976.

- Time
  Questions of representing and using temporal information strike STAMMER2 in several ways.
  1. In order that appropriate explanations are generated, temporal information must be stored or computed. In either case there is an expense to the system which must be minimized.
  2. Some of the data in the memory are timeless, some have time but is always true, some information degrades with time, and other information changes with time (a reappearance of the frame problem in the context of temporal knowledge). Perhaps a taxonomy of relations based on the time property of the relation would be useful.
  3. A distinction should be drawn between system time and world time. This might entail grouping data into events.

- Space
  An efficient way of representing global spatial situations such as land masses is needed in STAMMER2.

- Natural Language
  For the most part STAMMER2 avoids the serious problem of natural language processing. Nevertheless, an improvement in the style of the system's output would be a convenience to the casual user.

Appendix I. RULES IN THE DEMONSTRATION SYSTEM

```
NAME: INHERIT

CONDITIONS:
*UNKNOWN is really *PLAT
*PLAT is a *TYP
*PLAT is *ID1
*PLAT is *IDMP
*PLAT is a *CLS
*PLAT is *MED

ACTION:
*UNKNOWN is a *TYP
*UNKNOWN is *ID1
*UNKNOWN is *IDMP
*UNKNOWN is a *CLS
*UNKNOWN is *MED

CONFIDENCE: 1.0

NAME: NOT-LAST-SIGHTING

CONDITIONS:
*S1 is a sighting of *PLAT
*S2 is a sighting of *PLAT
*NOT*
     *S1 is the same as *S2
*S1 occurred at *T1
*S2 occurred at *T2
*T1 is less than *T2
*UNLESS*
     *S1 is other than a last sighting of its
        platform.

ACTION:
*S1 is other than a last sighting of its platform.

CONFIDENCE: 1.0

NAME: NOT-FIRST-SIGHTING

CONDITIONS:
*S1 is a sighting of *PLAT
*S2 is a sighting of *PLAT
*NOT*
     *S1 is the same as *S2
*S1 occurred at *T1
*S2 occurred at *T2
```

PRECEDING PAGE BLANK - NOT FI

```
*T2 is less than *T1
*UNLESS*
     *S1 is other than a first sighting of its
        platform.

ACTION:
*S1 is other than a first sighting of its platform

CONFIDENCE: 1.0

NAME: FIRST-VIEW

CONDITIONS:
*S1 is a sighting of *PLAT
*UNLESS*
     *S1 is other than a first sighting of its
        platform.

ACTION:
*S1 is the first sighting of *PLAT

CONFIDENCE: .99

NAME: NOT-KNOWN-COMBATANT

CONDITIONS:
*CONT is a contact
*S1 is a sighting of *CONT
*UNLESS*
     *S1 is reachable from *S2
even considering possible patrol overflights
*S2 is a sighting of *PLAT
*PLAT is MIL-BATTLE
*UNLESS*
     *PLAT is  a OWNSHIP

ACTION:
*CONT is a MERCHANT

CONFIDENCE: .45

NAME: REACHABLE

CONDITIONS:
*CONT is a contact
*S1 is a sighting of *CONT
*S2 is a sighting of *PLAT
*NOT*
     *PLAT is the same as *CONT
*UNLESS*
```

```
     *PLAT is  a OWNSHIP
*UNLESS*
     *PLAT is  a OWNSHIP
*S1 is reachable from *S2
ignoring contrary evidence.
*UNLESS*
     The ship in *S1 is the ship in *S2
because of an intersecting patrol overflight

ACTION:
*S1 is reachable from *S2
even considering possible patrol overflights

CONFIDENCE: .97

NAME: SIMPLY-REACHABLE

CONDITIONS:
*CONT is a contact
*S1 is a sighting of *CONT
*S2 is a sighting of *PLAT
*PLAT is MIL-BATTLE
*NOT*
     *CONT is the same as *PLAT
*UNLESS*
     *PLAT is  a OWNSHIP
*P1 is the position of *S1
*P2 is the position of *S2
*S1 occurred at *T1
*S2 occurred at *T2
A ship at *P1 at time *T1 could
reach *P2 at time *T2
 by travelling at top speed (or less).

ACTION:
*S1 is reachable from *S2
ignoring contrary evidence.

CONFIDENCE: .98

NAME: POSS-RPT

CONDITIONS:
*PTL is a patrol
*CONT is a contact
*S1 is a sighting of *CONT
*S2 is a sighting of *PLAT
*PLAT is MIL-BATTLE
*UNLESS*
     *PLAT is  a OWNSHIP
```

```
*PTL is the source of *S2
*NOT*
     *S1 is the same as *S2
*UNLESS*
     *CONT is dissimilar to *PLAT

ACTION:
One of the reports from *PTL concerns *CONT

CONFIDENCE: .95

NAME: BLOCKER

CONDITIONS:
*CONT is a contact
*S1 is a sighting of *CONT
*S2 is a sighting of *PLAT
*PLAT is MIL-BATTLE
*NOT*
     *CONT is the same as *PLAT
*UNLESS*
     *PLAT is  a OWNSHIP
*PTL is a patrol
*UNLESS*
     One of the reports from *PTL concerns *CONT
*S3 is a sighting of *PTL
*S3 is other than a last sighting of its platform.
*S4 is the successor (in time) of *S3
*P1 is the position of *S1
*P2 is the position of *S2
*P3 is the position of *S3
*P4 is the position of *S4
*S1 occurred at *T1
*S2 occurred at *T2
*S3 occurred at *T3
*S4 occurred at *T4
*OR*
     The path from *P1 to *P2
 does cross the path from
*P3 to *P4
     The path from *P1 to *P2
 does graze the path from
*P3 to *P4
*NOT*
     A ship moving from *P2 to *P1
 between the times *T2 and *T1
could have avoided sighting by a patrol travelling
from
*P3 to *P4 between *T3 and *T4
```

```
by traversing the patrol viewing area before the
        flight
*NOT*
     A ship moving from *P2 to *P1
 between the times *T2 and *T1
could have avoided sighting by a patrol travelling
from
*P3 to *P4 between *T3 and *T4
by traversing the patrol viewing area after the
        flight

ACTION:
The ship in *S1 is the ship in *S2
because of an intersecting patrol overflight

CONFIDENCE: -.9

NAME: CREATEDETECT

CONDITIONS:
*SGT is a sighting of *PLAT
EW is the source of *SGT
*UNLESS*
     *PLAT is a detection

ACTION:
*PLAT is a detection

CONFIDENCE: 1.0

NAME: CREATECONTACT

CONDITIONS:
*SGT is a sighting of *PLAT
RADAR is the source of *SGT
*UNLESS*
     *PLAT is a contact

ACTION:
*PLAT is a contact

CONFIDENCE: 1.0

NAME: CREATEPLAT

CONDITIONS:
*SGT is a sighting of *PLAT
*UNLESS*
     *PLAT is  a OWNSHIP
*UNLESS*
```

```
     *PLAT is a platform

ACTION:
*PLAT is a platform

CONFIDENCE: 1.0

NAME: SMALL-CRAFT9

CONDITIONS:
*WHO is a contact
*S1 is the first sighting of *WHO
RADAR is the source of *S1
*R1 is the range of *S1
*R1 is less than 8
Signal at *S1 is STRONG

ACTION:
*WHO is a SUB
*WHO is SURFACE

CONFIDENCE: .5

NAME: SMALL-CRAFT6

CONDITIONS:
*X is a contact
*SIGHT is a sighting of *X
*SIGHT is other than a first sighting of its
        platform.
*R is the range of *SIGHT
*R is less than 16
*R is greater than 9
Signal at *SIGHT is WEAK
*SPD is the speed of *SIGHT
*UNLESS*
     *SPD is greater than 20

ACTION:
*OR*
     *X is a FISHING
     *X is a PATROL
     (MODE *X SURFACE) is  a (TYPE *X SUB)

CONFIDENCE: .15

NAME: SMALL-CRAFT5

CONDITIONS:
*WHO is a contact
```

```
*S1 is a sighting of *WHO
*S1 is other than a first sighting of its platform
RADAR is the source of *S1
*RANGE is the range of *S1
*RANGE is less than 16
*RANGE is greater than 9
Signal at *S1 is WEAK
*SPEED is the speed of *S1
*SPEED is greater than 20

ACTION:
*OR*
     *WHO is a SUB
     *WHO is a PATROL

CONFIDENCE: .3

NAME: SMALL-CRAFT4

CONDITIONS:
*UNKNOWN is a contact
*SIGHTING1 is a sighting of *UNKNOWN
*SIGHTING1 is *DIST miles from land
RADAR is the source of *SIGHTING1
*RANGE is the range of *SIGHTING1
*RANGE is less than 9
*RANGE is greater than 3
Signal at *SIGHTING1 is WEAK
*DIST is less than 50

ACTION:
*OR*
     *UNKNOWN is a SUB
     *UNKNOWN is a SHORE-PATROL
     *UNKNOWN is a PLEASURE
     *UNKNOWN is a COMMERCIAL
     *UNKNOWN is a LANDING

CONFIDENCE: .1

NAME: SMALL-CRAFT3

CONDITIONS:
*UNKNOWN is a contact
*SIGHTING is a sighting of *UNKNOWN
*SIGHTING is *DIST miles from land
RADAR is the source of *SIGHTING
*RANGE is the range of *SIGHTING
*RANGE is less than 9
*RANGE is greater than 3
```

```
Signal at *SIGHTING is WEAK
*DIST is greater than 50

ACTION:
*UNKNOWN is a SUB

CONFIDENCE: .35

NAME: SMALL-CRAFT2

CONDITIONS:
*UNKNOWN is a contact
*SIGHTING is a sighting of *UNKNOWN
*SIGHTING is other than a first sighting of its
        platform.
RADAR is the source of *SIGHTING
Signal at *SIGHTING is WEAK
*SPEED is the speed of *SIGHTING
*UNLESS*
     *SPEED is greater than 3

ACTION:
*OR*
     *UNKNOWN is a DEBRIS
     *UNKNOWN is a SUB
     *UNKNOWN is a BUOY

CONFIDENCE: .12

NAME: SMALL-CRAFT1

CONDITIONS:
*UNKNOWN is a contact
*SIGHTING is a sighting of *UNKNOWN
*SIGHTING is other than a first sighting of its
        platform.
RADAR is the source of *SIGHTING
*RANGE is the range of *SIGHTING
*RANGE is less than 3
Signal at *SIGHTING is WEAK
*SPEED is the speed of *SIGHTING
*SPEED is greater than 3

ACTION:
*UNKNOWN is a SUB
*OR*
     *UNKNOWN is PERISCOPE
     *UNKNOWN is SNORKEL

CONFIDENCE: .6
```

```
NAME: ID-LANE

CONDITIONS:
*LANE is a merchant lane
*SHIP is a platform
*SIGHTING is a sighting of *SHIP
The location of *LANE is *LANELOC
*POS is the position of *SIGHTING
*POS is in the merchantlane *LANELOC

ACTION:
*SIGHTING is inside a merchantlane

CONFIDENCE: 1.0

NAME: INSIDE-A-STORM

CONDITIONS:
*SHIP is a platform
*SIGHTING is a sighting of *SHIP
*STORM is  a STORM
*POS is the position of *SIGHTING
The location of *STORM is *STMLOC
*POS is inside *STMLOC

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.25

NAME: CLOSE-POPUP

CONDITIONS:
*SHIP is a contact
*SIGHTING is the first sighting of *SHIP
*RANGE is the range of *SIGHTING
*RANGE is less than 12

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.2

NAME: DISTANT-POPUP

CONDITIONS:
*SHIP is a contact
*SIGHTING is the first sighting of *SHIP
*RANGE is the range of *SIGHTING
```

```
*RANGE is greater than 30

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.2

NAME: COURSE-CHANGED

CONDITIONS:
*SHIP is a contact
*SIGHTING1 is a sighting of *SHIP
*SIGHTING1 is other than a first sighting of its
         platform.
*SIGHTING1 is other than a last sighting of its
        platform.
*SIGHTING2 is the successor (in time) of
        *SIGHTING1
*COURSE1 is the course of *SIGHTING1
*COURSE2 is the course of *SIGHTING2
*UNLESS*
     *COURSE2 is roughly the same course as
        *COURSE1

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.3

NAME: SPEED-CHANGED

CONDITIONS:
*SHIP is a contact
*SIGHTING is a sighting of *SHIP
*SIGHTING is other than a first sighting of its
        platform.
*SIGHTING is other than a last sighting of its
        platform.
*SIGHTING2 is the successor (in time) of *SIGHTING
*SPEED1 is the speed of *SIGHTING
*SPEED2 is the speed of *SIGHTING2
*UNLESS*
     *SPEED2 is roughly the same speed as *SPEED1

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.3

NAME: FASTER-THAN-A-MERCHANT
```

```
CONDITIONS:
*SHIP is a contact
*SIGHTING is a sighting of *SHIP
*SIGHTING is other than a first sighting of its
       platform.
*SPEED is the speed of *SIGHTING
*SPEED is greater than 25

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.25

NAME: SLOWER-THAN-A-MERCHANT

CONDITIONS:
*SHIP is a contact
*SIGHTING is a sighting of *SHIP
*SIGHTING is other than a first sighting of its
       platform.
*SPEED is the speed of *SIGHTING
*SPEED is less than 9

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.15

NAME: MATCH-PLAT

CONDITIONS:
*SGT1 is a sighting of *PLAT1
*SGT1 is other than a first sighting of its
       platform.
*SGT2 is a sighting of *PLAT2
*UNLESS*
     *PLAT1 is the same as *PLAT2
*UNLESS*
     *SGT2 is other than a last sighting of its
       platform.
*CRS1 is the course of *SGT1
*SPD1 is the speed of *SGT1
*POS1 is the position of *SGT1
*SGT1 occurred at *T1
*POS2 is the position of *SGT2
*SGT2 occurred at *T2
*T2 is less than *T1
The course from *POS2 to *POS1 is *CRS2
To move from *POS2 to *POS1
```

```
between *T2 and *T1 implies a speed of *SPD2
*CRS2 is roughly the same course as *CRS1
*SPD2 is roughly the same speed as *SPD1

ACTION:
*PLAT1 is really *PLAT2

CONFIDENCE: .5

NAME: OUTSIDE-ALL-LANES

CONDITIONS:
*SHIP is a platform
*UNLESS*
     *SHIP is AIR
*SIGHTING is a sighting of *SHIP
*UNLESS*
     *SIGHTING is inside a merchantlane

ACTION:
*SHIP is a MERCHANT

CONFIDENCE: -.08
```

Appendix II. TECHNICAL DATA BASE CONTENT

```
A0175: VIKING is a S-3A
A0174: VIKING is AIR
A0173: VIKING is a RECONNISANCE
A0172: VIKING is MIL-AUXIL
A0171: VIKING is FRIEND
A0170: VIKING is a platform
A0169: SEASPRITE is a SH-2F
A0168: SEASPRITE is AIR
A0167: SEASPRITE is a HELICOPTER
A0166: SEASPRITE is MIL-BATTLE
A0165: SEASPRITE is FRIEND
A0164: SEASPRITE is a platform
A0163: ORION is a P-3C
A0162: ORION is AIR
A0161: ORION is a RECONNISANCE
A0160: ORION is MIL-AUXIL
A0159: ORION is FRIEND
A0158: ORION is a platform
A0157: HORMONE is a KA-25
A0156: HORMONE is AIR
A0155: HORMONE is a HELICOPTER
A0154: HORMONE is MIL-BATTLE
A0153: HORMONE is HOSTILE
A0152: HORMONE is a platform
A0151: HAWKEYE is a E-2B
A0150: HAWKEYE is AIR
A0149: HAWKEYE is a RECONNISANCE
A0148: HAWKEYE is MIL-AUXIL
A0147: HAWKEYE is FRIEND
A0146: HAWKEYE is a platform
A0145: HARRIER is a AV-8A
A0144: HARRIER is AIR
A0143: HARRIER is a FIGHTER
A0142: HARRIER is MIL-BATTLE
A0141: HARRIER is FRIEND
A0140: HARRIER is a platform
A0139: FOXBAT is a MIG25
A0138: FOXBAT is AIR
A0137: FOXBAT is a FIGHTER
A0136: FOXBAT is MIL-BATTLE
A0135: FOXBAT is HOSTILE
A0134: FOXBAT is a platform
A0133: CORSAIR is a A-7
A0132: CORSAIR is AIR
A0131: CORSAIR is a FIGHTER
A0130: CORSAIR is MIL-BATTLE
A0129: CORSAIR is FRIEND
```

```
A0128: CORSAIR is a platform
A0127: BACKFIRE is a RV-G
A0126: BACKFIRE is AIR
A0125: BACKFIRE is a BOMBER
A0124: BACKFIRE is MIL-BATTLE
A0123: BACKFIRE is HOSTILE
A0122: BACKFIRE is a platform
A0121: RATHBURNE is a KNOX
A0120: RATHBURNE is SURFACE
A0119: RATHBURNE is a FRIGATE
A0118: RATHBURNE is MIL-BATTLE
A0117: RATHBURNE is FRIEND
A0116: RATHBURNE is a platform
A0115: YANK-1 is a YANKEE
A0114: YANK-1 is SUB
A0113: YANK-1 is a SUB
A0112: YANK-1 is MIL-BATTLE
A0111: YANK-1 is HOSTILE
A0110: YANK-1 is a platform
A0109: WAINWRIGHT is a BELKNAP
A0108: WAINWRIGHT is SURFACE
A0107: WAINWRIGHT is a CRUISER
A0106: WAINWRIGHT is MIL-BATTLE
A0105: WAINWRIGHT is FRIEND
A0104: WAINWRIGHT is a platform
A0103: SUNFISH is a STURGEON
A0102: SUNFISH is SUB
A0101: SUNFISH is a SUB
A0100: SUNFISH is MIL-BATTLE
A0099: SUNFISH is FRIEND
A0098: SUNFISH is a platform
A0097: PROVORNY is a KASHIN
A0096: PROVORNY is SURFACE
A0095: PROVORNY is a DESTROYER
A0094: PROVORNY is MIL-BATTLE
A0093: PROVORNY is HOSTILE
A0092: PROVORNY is a platform
A0091: MINSK is a KIEV
A0090: MINSK is SURFACE
A0089: MINSK is a CARRIER
A0088: MINSK is MIL-BATTLE
A0087: MINSK is HOSTILE
A0086: MINSK is a platform
A0085: MEYERCORD is a KNOX
A0084: MEYERCORD is SURFACE
A0083: MEYERCORD is a FRIGATE
A0082: MEYERCORD is MIL-BATTLE
A0081: MEYERCORD is FRIEND
A0080: MEYERCORD is a platform
A0079: LAWRENCE is a CHAS.ADAMS
```

```
A0078: LAWRENCE is SURFACE
A0077: LAWRENCE is a DESTROYER
A0076: LAWRENCE is MIL-BATTLE
A0075: LAWRENCE is FRIEND
A0074: LAWRENCE is a platform
A0073: HASSAYAMPA is a NEOSHO
A0072: HASSAYAMPA is SURFACE
A0071: HASSAYAMPA is a OILER
A0070: HASSAYAMPA is MIL-AUXIL
A0069: HASSAYAMPA is FRIEND
A0068: HASSAYAMPA is a platform
A0067: HALSEY is a LEAHY
A0066: HALSEY is SURFACE
A0065: HALSEY is a CRUISER
A0064: HALSEY is MIL-BATTLE
A0063: HALSEY is FRIEND
A0062: HALSEY is a platform
A0061: ECHO-1 is a ECHOII
A0060: ECHO-1 is SUB
A0059: ECHO-1 is a SUB
A0058: ECHO-1 is MIL-BATTLE
A0057: ECHO-1 is HOSTILE
A0056: ECHO-1 is a platform
A0055: DESNA is a KAZBEK
A0054: DESNA is SURFACE
A0053: DESNA is a OILER
A0052: DESNA is MIL-AUXIL
A0051: DESNA is HOSTILE
A0050: DESNA is a platform
A0049: CONSTELLATION is a KITTYHAWK
A0048: CONSTELLATION is SURFACE
A0047: CONSTELLATION is a CARRIER
A0046: CONSTELLATION is MIL-BATTLE
A0045: CONSTELLATION is FRIEND
A0044: CONSTELLATION is a platform
A0043: ADMIRAL MAKAROV is a KRESTAII
A0042: ADMIRAL MAKAROV is SURFACE
A0041: ADMIRAL MAKAROV is a CRUISER
A0040: ADMIRAL MAKAROV is MIL-BATTLE
A0039: ADMIRAL MAKAROV is HOSTILE
A0038: ADMIRAL MAKAROV is a platform
A0037: ADMIRAL GOLOVKO is a KYNDA
A0036: ADMIRAL GOLOVKO is SURFACE
A0035: ADMIRAL GOLOVKO is a CRUISER
A0034: ADMIRAL GOLOVKO is MIL-BATTLE
A0033: ADMIRAL GOLOVKO is HOSTILE
A0032: ADMIRAL GOLOVKO is a platform
A0031: CONNOLE is a KNOX
A0030: CONNOLE is a FRIGATE
A0029: CONNOLE is MIL-BATTLE
```

```
A0028: CONNOLE is FRIEND
A0027: CONNOLE is  a OWNSHIP
A0026: The location of LANE3 is((55.66 -39.84)
(57.23 -36.36) (58.56 -32.89) (59.77 -29.01)
(61.17 -23.79) (62.08 -19.37) (62.99 -13.96)
(63.79 -6.72))
A0025: The location of LANE2 is ((56.04 -42.25)
(58.45 -37.9) (60.37 -33.75) (61.85 -29.94)
(63.19 -26.0) (64.01 -22.99))
A0024: The location of LANE1 is((68.93 -13.82)
(68.39 -16.57) (66.79 -23.11) (66.11 -25.32)
(65.02 -28.53) (64.19 -30.47) (63.34 -32.47)
(62.11 -35.08) (60.64 -37.76) (59.21 -40.16)
(58.14 -41.7))
A0023: ST.JOHNS is the starting port of LANE3
A0022: MURMANSK is the destination port of LANE3
A0021: ST.JOHNS is the starting port of LANE2
A0020: REYKJAVIK is the destination port of LANE2
A0019: MURMANSK is the starting port of LANE1
A0018: REYKJAVIK is the destination port of LANE1
A0017: LANE3 is a merchant lane
A0016: LANE2 is a merchant lane
A0015: LANE1 is a merchant lane
```

Appendix III. FORMAT OF LWES-PRODUCED FILES

LWES produces two files during a run. These are the sensor report file and the ground truth location file. The actual file names are determined by the values of the variables SENSORFILE and GTHFILE, which may be set before beginning a run in LWES.

The ground truth file is intended for use with the Baseline Correlation Program. At present, the file format (subject to change) is a series of platform entries, each of which consists of a type indicator (eg FF, CLG) followed by latitude, longitude, altitude, and time. These platform entries are sorted by time (ascending) and, within each time class, by platform name (despite the fact that platform name is not included in the file).

The sensor report file is used by STAMMER2. Entries in this file are in one of four formats. During the LWES run, one ship is selected as the home ship. This should be the same one used in STAMMER2. Sensor detections by the home ship are labelled with the detecting sensor type, while detections by other ships are merely labelled as "external." Sensor detections can also be divided into "precise" and "imprecise" position classes. Radar and sonar give precise locations for the detected platforms, while EW intercepts provide only line-of-bearing information. In the case of external platforms, this problem is even greater.

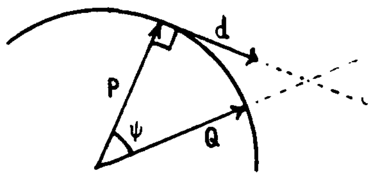
Report classifications and formats are as follows:

- o own ship/active--(detectedname sensortype lat lon time)
- o own ship/passive--(detectedname EW bearing detectedsensor time)
- o external/active--(detectedname EXTERNAL lat lon time)
- o external/passive--(detectedname EXTERNAL bearing detectingname detectinglat detectinglon time)

Appendix IV. MATHEMATICAL TECHNIQUES

The methods were those used in STAMMER [1] with the addition of a formula for computing latitude and longitude given range and bearing from a reference point. The derivation is as follows.

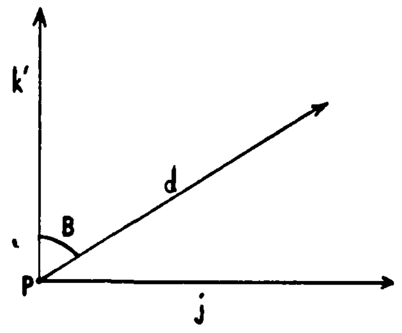
Assume, without loss of generality, that units are chosen so that the earth has radius 1. First suppose the reference point has longitude zero. Let P be the reference point and Q the target point. We will also use P and Q as the position vectors of these points. Thus P and Q are unit vectors. Let B be the bearing from P to Q. Let ψ be the angle subtended at the Earth's center by P and Q. (This is easily computed from the range.) Let d be a unit vector tangent to the Earth at P, aimed toward Q as shown:



We see that $Q/\cos\psi = P + d \tan\psi$, by vector algebra, or

$$Q = P \cos\psi + d \sin\psi. \qquad (1)$$

Let k' be a unit vector tangent to the Earth at P, in a northerly direction. The unit vector j points due east from P.



We may write

$$d = k' \cos B + j \sin B. \qquad (2)$$

If we let θ be the latitude of P and k be a unit vector parallel to the Earth's axis, in the northward direction, then

PRECEDING PAGE BLANK-NO

$k' = -i \sin\theta + k\cos\theta.$ (3)

We may now combine 1, 2, and 3 to get

$$Q = \cos\psi\,(i\cos\theta + k\sin\theta) + \sin\psi\,[\cos B(-i\sin\theta + k\cos\theta) + j\sin B].$$

Suppose now that θ_2 and ϕ_2 are the derived latitude and longitude, respectively, of Q. We have

$$Q = i\cos\theta_2\cos\phi_2 + j\cos\theta_2\sin\phi_2 + k\sin\theta_2 .$$

By equating corresponding coefficients of the two vector equations for Q, we may derive the following formulae:

$\theta_2 = \sin^{-1}[\sin\theta\cos\psi + \cos B\cos\theta\sin\psi]$ (4)

$\phi_2 = \sin^{-1}[\sin B\sin\psi / \cos\theta_2]$ (5)

$\phi_2 = \cos^{-1}[(\cos\theta\cos\psi - \cos B\sin\theta\sin\psi) / \cos\theta_2]$ (6)

Equations 5 and 6, while seemingly redundant, are both necessary because arcsin and arccos are ambiguous. We can combine them into one unambiguous formula by writing

$\phi_2 = S\cos^{-1}[(\cos\theta\cos\psi - \cos B\sin\theta\sin\psi) / \cos\theta_2]$ (7)

where

$S = 1$ if $(\sin B\sin\psi / \cos\theta_2)$ geq 0
-1 otherwise.

Finally, we consider the situation in which the longitude of the reference point is non-zero. By a suitable rotation of coordinates about the Earth's axis, we can reduce this case to the previous one. Let ϕ_2' be the longitude thereby obtained, using 7. To derive the correct longitude in the original coordinate system, we must now rotate back, giving us

$$\phi_2 = \phi_2' + \phi$$

where ϕ is the longitude of the reference point. Thus, in general,

$$\phi_2 = \phi + \mathrm{SIGN}(\sin B\sin\psi / \cos\theta_2)\,\cos^{-1}[(\cos\theta\cos\psi - \cos B\sin\theta\sin\psi) / \cos\theta_2].$$

REFERENCES

1. NOSC TD 252, STAMMER: System for Tactical Assessment of Multisource Messages, Even Radar, by RJ Bechtel and PH Morris, May, 1979.

2. Computer-Based Medical Consultations: MYCIN, by EH Shortliffe; Elsevier, 1976.

3. The HARPY Speech Understanding System, by BT Lowerre and R Reddy; Trends in Speech Recognition, WA Lea, ed, Prentice-Hall, 1979.

4. The HEARSAY Speech Understanding System: An Example of the Recognition Process, by DR Reddy, LE Erman, RD Fennell, and RB Neely; Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford CA, 1973, p 185-193.

5. Technical Report 136, Semantic Network Representation in Rule-Based Inference Systems, by RO Duda, PE Hart, NJ Nilsson, and GL Sutherland, SRI International, 1977.

6. The Art of Computer Programming, Vol 3: Sorting and Searching, by D Knuth, Addison-Wesley, 1975.

7. Content Addressable Parallel Processors, by CC Foster, Van Nostrand Reinhold, 1976.

8. INTERLISP Reference Manual, by W Teitelman, Xerox Palo Alto Research Center, 1978.

9. LUCID, A Nonprocedural Language with Iteration, by EA Ashcroft and WW Wadge; Communications of the Association for Computing Machinery, vol 20, 1977, p 519.

10. Technical Report 114A, An Asynchronous Programming Language and Computing Machine, by Arvind, KP Gostelow, and W Plouffe, Information and Computer science Department, University of California, Irvine, 1978.

11. Spontaneous Computation and its Role in AI Modeling, by C Rieger; Patten-Directed Inference Systems, DA Waterman and F Hayes-Roth, ed, Academic Press, 1978.

12. Transition Network Grammars for Natural Language Analysis, by WA Woods; Communications of the Association for Computing Machinery, vol 13, 1970, p 591.

13. BBN Report 2378, The LUNAR Sciences Natural Language Information System: Final Report, by WA Woods, RM Kaplan, and BL Nash-Webber; Bolt, Beranek, and Newman, Inc, 1972.

14. NOSC TN 530, DSPLA--A Graphics Package for ıactical Situation Assessment (Version II), by GG McIntyre, 15 September 1978.

15. Optimality of Multiple-Key Hashing Functions, by A Bolour; Journal of the Association for Computing Machinery, vol 26, 1979.

16. The Metanovel: Telling Stories by Computer, by JR Meehan, PhD Thesis, Yale University, December 1976.

INDEX

INITIAL DISTRIBUTION

NAVAL ELECTRONIC SYSTEMS COMMAND
CODE 330 (CC STOUT) (2)
CODE PME-108 (D SCHUTZER)

NAVAL RESEARCH LABORATORY
CODE 7509 (JH KULBACK)
CODE 7932 (HL WIENER)

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
IPTO (DR R ENGELMORE) (2)
IPTO (LCDR AJ DIETZLER) (2)
IPTO (DR R KAHN)

SYSTEMS DEVELOPMENT CORP
3065 ROSECRANS PLACE
SAN DIEGO, CA 92110
RJ BECHTEL (4)
PH MORRIS (4)
DF KIBLER (4)

MITRE CORP
WESTGATE RESEARCH PARK
MC LEAN, VA 22101
JW BENOIT

RAND CORP
1700 MAIN STREET
SANTA MONICA, CA 90406
G MARTINS
J GILLOGLY

BOLT, BERANEK AND NEWMAN
50 MOULTON STREET
CAMBRIDGE, MA 02138
RJ BOBROW
NR GREENFIELD
O SELFRIDGE
J VITTAL

COMPUTER CORPORATION OF AMERICA
575 TECHNOLOGY SQ
CAMBRIDGE, MA 02139
GA WILSON
JB ROTHNIE

VERAC INC
4901 MORENA BLVD
SAN DIEGO, CA 92117
C MOREFIELD
J NASH
J TIERNAN

ROME AIR DEVELOPMENT CENTER
GRIFFIS AFB, NY 13441
ISIS (N FOWLER III)

NAVAL AIR DEVELOPMENT CENTER
CODE 6021 (LT S HARRIS) (2)

MITRE CORP
PO BOX 208
BEDFORD, MA 01730
CARL ENGLEMAN

ESL INC
495 JAVA DR
SUNNYVALE, CA 94086
GK KIRENUDJIAN

SYSTEMS CONTROL INC
1801 PAGE MILL RD
PALO ALTO, CA 94304
JR PAYNE
RP WISHNER

SYSTEMS EXPLORATION INC
1340 MUNRAS AVE
MONTEREY, CA 93940
R BUTTERWORTH
G GIBBONS

SRI INTERNATIONAL
333 RAVENSWOOD AVE
MENLOW PARK, CA 94025
E SACERDOTI
J OLMSTEAD
D SAGALOWICZ

CTEC INC
7777 LEESBURG PIKE
FALLS CHURCH, VA 22043
KD SHERE

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93940
CODE 55 PK (G POOCK)
CODE 52 RL (RJ ROLAND)

DEFENSE TECHNICAL INFORMATION CENTER (12)