EXPERIMENTS WITH A NATURAL LANGUAGE PROBLEM-SOLVING SYSTEM**

by
Jack P. Gelb
IBM Corporation
Systems Development Division
Poughkeepsie, New York USA

Abstract

Development work on a computer program called HAPPINESS which solves basic probability problems phrased in English is presented. Emphasis is placed on

(1) the application of heuristics to the examination of input language structure for the purpose of determining those phases richest in semantic content

(2) the piecewise construction of combinatorial formulas for problem solution

The language analysis is accomplished in several discrete stages, involving simple sentence transformation, keyword and semantic scanning, and syntactic analysis based on a simplified context-free grammar. The descriptor list result of this analysis is used as the basis for a four-stage solution procedure.

A description of the implementation, and a discussion of its limitations, extensions, and applications, is also given.

KEY WORDS AND PHRASES:
probability problem-solving, question-answering systems, formal language applications, heuristic problem-solving

** Work herein described was used as the basis for a thesis submitted in partial fulfillment of the requirements for the Ph.D. degree in Computer Science at Rensselaer Polytechnic Institute.

Introduction

During the past few years, several research projects have been devoted to the development of computerized natural-language question-answering systems. These endeavors may be roughly divided (a la Bobrow[I] and Simmons [7]) into two categories: data-based and text-based. The former classification refers to those systems which initially contain well-structured sets of data representing potential answers (or parts of answers) to questions in some given subset of knowledge. Solutions may utilize sophisticated inlormatlon retrieval methods, theorem - proving techniques, or other inference schema in performing the deduction from question to answer. The data base is usually large and essentially static. A keyword (semantic) scan of the input question often suffices to provide the extra information required to deduce the desired answer. Some systems also employ structural (syntactic) analysis of the input to determine relationships among semantic objects, and then search the data base for similar relations ships among stored objects (cf. Simmons [7]).

Text-based question-answering systems, which we shall refer to as problem-solving systems, uniformly contain small initial data bases. New data is added to the data base from the input text prior to attempting solution, but this information is normally discarded immediately after a solution is generated. Some, such as Raphael's SIR [6] and Lindsay's SAD SAM[5], continuously augment their data bases with information gleaned from the input, and utilize both kinds of data for future solutions; the new data is discarded at the end of a problem-solving session, however. Usually, some form of syntax analysis of the problem text is required so that the relationships among semantical objects may be uncovered and data base entries created. Bobrow's STUDENT [1] applied a simple transformational grammar to the input problem text, and demonstrated that such a grammar was sufficient to produce the equational representations of high-school level algebra word-story problems. Charniak [2] went on to show how a slightly more sophisticated parser, when applied to calculus rate problems by his CARPS system, could be used to determine the cross-dependency structures present in that class of problems.

In the research reported herein, we set out to design and implement a system for the solution of probability problems phrased in English, such as might be found at the end of the initial chapters of a college-level probability text. Since the input language was reasonably complex, sophisticated techniques for syntactic and semantic analyses were postulated; in reality, the judicious application of

heuristics enabled us to produce a simplified analysis which was still comprehensive enough to handle all but the most pathological cases. In addition, solutions were produced symbolically and in a piecemeal fashion, with an eye toward the possible didactic applications of that method. Only at the final stage of solution were numbers actually displayed.

The system, dubbed HAPPINESS (for Heuristic Analysis of Probability Problems In a Natural-language Environment with Symbolic Solutions) has been programmed in LISP 1.5 and is running interactively under IBM's TSS/67.

The HAPPINESS Svstem

The system is divided into two main, independent subsections: a language analyzer, and a solution generator. There is also a control section, called DRIVER, whose job it is to communicate with the user, pass pertinent information from the language analyzer to the solution generator, and generally, to drive the system (see Fig. 1). Each of the two subsections performs its own error checking and/or correction; unrecoverable errors are currently conveyed to the user via the driver, and a new problem requested. The timing information displayed by DRIVER refers to actual CPU time since the last display, and includes garbage collection and paging times. Although, for the purpose of this experiment, solution times were not considered important, the delays incurred were reasonably minimal.

The Language Analyzer

The purpose of the language analyzer is to create a descriptor-list (or, desclist) representation of the input problem. This list is a compact tree-like representation of those semantic objects in the problem statement essential to effect a solution. The nodes of the tree are categorically prespeeified; the tips of the tree refer to the semantic objects. For example, the analysis of the problem:

(WHAT IS THE PROBABILITY OF
GETTING TWO OR) MORE HEADS
OR EXACTLY 3 TAILS WHEN FOUR
COINS ARE TOSSED *ONCE?)*

would ultimately result in a desclist represented by the tree in Fig. *2.*

Examination of elementary textbook probability problems indicated that they overwhelmingly appear in one of two forms. The first is a one-sentence question containing the description of the probability outcome space, combined with an indication of which of those outcomes are to be considered favorable; this is the case with our sample problem. The second form consists of one or more sentences describing the outcome space, followed by a question sentence specifying the desirable outcomes. The goal of the language analyzer is to reduce both forms to a common form via transformations, and then apply semantic and syntactic analysis to provide a desclist equivalent of the problem. This goal is accomplished in three or four steps, as outlined below.

In the first pass over the input text, "number" words are replaced by their equivalent digits, phrases and synonyms are replaced by more meaningful forms, and phrase-modified numbers arc so indicated. At the conclusion of this first transformational pass, the above problem becomes

(WHAT IS THE PROBABILITY OF
GETTING (GE 2) HEADS OR 3
TAILS WHEN 4 COINS ARE
TOSSED 1 TIMES ?)

A secondary transformational pass is then performed. The purpose of this analysis is to break the problem down further into a series of simple sentences, followed by a question sentence. It will subsequently be assumed that the simple sentences contain a complete outcome space specification, while the question sentence contains a description of the favorable events. Examples of the kind of rearranging transformations performed arc

A WHEN B ? B . A ?
A FROM B WHICH verb C . →BverbC . A .
A THAT , IF B , C . → B. A THAT *C* .

where A, B , and C are clauses or phrases. The simple sentences thus produced are each then further rearranged, via the application of a simple context-free grammar, into a form consisting of a verb and its voice, a subject (possibly compount), and a predicate. The verb may not be the same as the one specified in the input, but one which is synonymous and representative oi a class of like verbs. Aside from singling it out, no further action is taken with respect to the question sentence. In our example, the simple sentences would be represented as

((TOSS / PASSIVE) (4 COINS) (1 TIMES))

and the question sentence reduced to

(WHAT IS THE PROBABILITY OF
GETTING (GE 2) HEADS OR 3
TAILS ?)

At this point, a third pass over the question sentence alone may occur, depending on its form. In our example, no turther action would be taken.

If, however, the question sentence had read

(WHAT IS THE PROBABILITY THAT 2 OR MORE HEADS, OR 3 TAILS APPEAR ?)

it would be reduced exactly to the previous (gerundive) form. This global reduction simplifies the grammar for the parser in the next pass by restricting the allowable sentential forms.

The final step in the language analysis consists of a keyword scan over the simple sentences combined with a syntax analysis of the question sentence. The keyword scan attempts to classify the problem by looking Ior those words with highest semantic content; in the type of probability problems under consideration, these words are those referring to dice, coins, or cards. If such a classification is possible, a search for modifiers of those words and related words is instituted; for the types of problems examined, this was sufficient to determine the outcome space exactly. If a classification of this nature is not possible, the verbs of the simple sentences are examined for semantic content. For example, verbs such as "distribute" and "place" would indicate arrangement-type problems, whereas "contain" and "draw" would indicate sampling. The voice of the verb determines which of the subject or predicate are to be examined for the objects being acted upon. In these cases, more information is extracted in the form of numbers and the modifiers and/or nouns they specify. This type of heuristic analysis was again adequate in determining the outcome space.

The syntax analysis of the question sentence is performed utilizing an LL (1) grammar for the English language subset on a simulated one-state pushdown machine (cf. Lewis et al [4]). The language subset is restricted to include just those grammatical constructs which normally appear in basic probability problems; excessive wordage was regarded as typical of more advanced texts. The actual words appearing can be arbitrary: the top-down, predictive analysis assumes word classifications if no semantics are present. Context sensitivities, such as antecedent references, are processed by associating "properties" with particular sentential forms and passing them through the analysis by combining them with affected productions (cf. Stearns and Lewis [8]). This permits the analyzer to make assumptions about relations between input words despite the lack of semantic information about them. The result of the syntax analysis is a symbolic description of the simple probability events described in the problem, together with a specification of the combination of these events required to solve the problem. This information, combined with the results of the semantic scan, constitute the desclist for a given problem. For our example, the desclist is pictured in Fig. 2. A more complete description of the analysis may be found in Gelb [3].

The Solution Generator

Solutions are generated in four steps using the information contained in the desclist. The first three stages produce symbolic results in a tutorial progression; the final stage produces a numerical result.

The first stage examines the compound event structure (i.e., the combination of simple events) in the desclist and performs a symbolic expansion according to the addition rule of probability:

$$P(\bigcup_i Ai) = \sum_i P(Ai) - \sum_{i<j} P(Ai \cap Aj) +$$

$$\sum_{i<j<k} P(Ai \cap Aj \cap Ak) - \ldots \pm P(\bigcap_i Ai)$$

This is a straightforward combinatorial problem, and, in our example, would result in a LISP equivalent of

P(E1) + P(E2) - P(EI∩E2)

Using this expansion, the second solution stage examines each multiple event term to determine if the component events are mutually exclusive or independent. In the latter case, the joint probability is replaced by the (symbolic) product of the component probabilities; in the former case, the term is just discarded. The criteria for mutually exclusive or independent events are a collection of heuristics established for each problem type; if there is doubt, the terms are not replaced. The sample problem would be recognized as consisting of two mutually exclusive events, and would be reduced to a form representing

P(E1) + P(E2)

In the third stage of solution, each symbolic probability term is examined and replaced by a combinatorial form representing its value. These new forms constitute the most explicit symbolic representation of the probabilities, containing combinations, permutations, fractions, and arithmetic operations. Producing these terms involves dissecting the events and combining indicated formulas and/or probability distributions; in certain cases, it is necessary to generate and examine the outcome space explicitly (this is particularly true of joint events). Much of this procedure is algorithmic; heuristics are applied when the desclist information is insufficient or incomplete. In our example, the result of the third stage would be the LISP equivalent of

$$\underbrace{\binom{4}{3}\left(\frac{1}{2}\right)^3\left(\frac{1}{2}\right)^1}_{P(E2)} + \underbrace{\binom{4}{2}\left(\frac{1}{2}\right)^2\left(\frac{1}{2}\right)^2 + \binom{4}{3}\left(\frac{1}{2}\right)^3\left(\frac{1}{2}\right)^1}_{P(E1)} + \binom{4}{4}\left(\frac{1}{2}\right)^4\left(\frac{1}{2}\right)^0$$

The fourth stage applies the indicated operations to achieve a numerical result; both a fraction and its decimal equivalent are displayed. The fractional result is presented in lowest terms as a consequence of expressing numerator and denominator as the products of their prime factors, and removing common factors. For the sample problem, this would result in the final solution

15/16 (or .9375)

The Driver

The driver requests a problem from the user, and passes it on to the language analyzer without embellishment. Part of the driver is resident in the language analyzer, controlling the flow from pass to pass, and reflecting errors back to the user. The actual construction of the desclist is performed by the driver, which then passes control to the solution generator. Each stage of the solution generation is returned to the driver for display, the driver then returns the result to the solution generator for further analysis. At the conclusion of the fourth solution stage, the driver re-initializes itself and requests a new problem. STOP terminates the problem-solving session.

Limitations and Extensions

HAPPINESS is currently limited to the solution of very basic probability problems. However, the system is extendable along several dimensions:

(1) The language complexity may be fairly easily increased by augmenting the current LL(1) grammar and specifying the new productions

(2) The problem types may be expanded by specifying the new keywords and adding the solution routines

(3) Interactive error recovery is readily implementable on a language level thanks to the power of predictive parsing techniques

(4) The words and phrases in the data base may be modified dynamically, thus permitting the system to learn (an artificial belief system would be helpful here)

(5) The addition of a theorem-prover to HAPPNESS would permit the solution of many basic probability problems of a theoretical nature

As HAPPINESS was constructed, areas (1) and (2) above were modified in a straightforward manner to provide increased solving capability. The power of LISP greatly enhances the ability to provide system extensions.

Discussion

The difficulties inherent in the computer solution of natural language probability problems were myriad and required the development of more sophisticated and general techniques to overcome than were utilized by the earlier STUDENT and CARPS systems. Aside from the challenging problems of linguistic analysis, the determination of the pertinent symbolic result formulas leading ultimately to a numerical solution was a formidable task.

Probability questions, including those of a basic nature intended for solution by HAPPINESS, encompass an extremely diverse class of problems which do not readily lend themselves to a generalized treatment. They each tend to require the creation of a single, specially tailored solution function as opposed to mere substitution into a universal form and, because of limited applicability, the developed formula Is normally discarded after use. To maintain a certain degree of flexibility, then, it became necessary to develop a collection of very powerful, primitive functions with multiple applications. This approach may be contrasted with the limited formulations available to STUDENT (simultaneous equations) and CARPS (differentiated system of equations).

The production of the solution formulation itself involved a great deal more than the identification of admissible problem variables and constants. Analytic treatment of the semantics of these elements was indicated at various stages in the formula generation process so as to eliminate the subsequent creation of unnecessary and incorrect solution components. Moreover, since the objects in the system were probabilistic events with potentially multiple semantic constraints (as opposed to variables with unique specifications), considerable circumspection of the Interrelationships among events constituting joint probabilities was required.

Because of the design intention to permit "unrestricted" language input (or, perhaps, limiting to empirically determined common forms), HAPPINESS was able to circumvent somewhat a common complaint in earlier systems. The utilization of heuristics in this regard, coupled with the judicious application of powerful syntactic analysis techniques adapted from recent compiler advances, allowed for greater scope and versatility in communicating problems to the system.

The order of magnitude increase in complexity of the problems attacked by HAPPINESS, combined with the required greater depth of solution capability within the system, has resulted in a successful and effective problem solving tool.

Conclusions

Many of the techniques utilized by this system appear to wear well; they can be applied equally well to problem-solving systems unconnected to probability. The predictive parsing algorithm utilizing LL(k) property grammars with heuristic decision-making procedures, permits a flexible analysis of the input language while maintaining necessary semantic constraints. Keyword analysis using heuristics also seems to allow reasonable ability to recognize problem elements containing semantic information. Symbolic solutions provide a tutorial capability, as well as guiding Interactive error recovery.

Combining the above techniques with more powerful deduction schema may provide a method with which to construct general problem-solving systems. Such systems, operating with small, extendable data bases, could be readily tailored to meet specific processing needs.

Figures

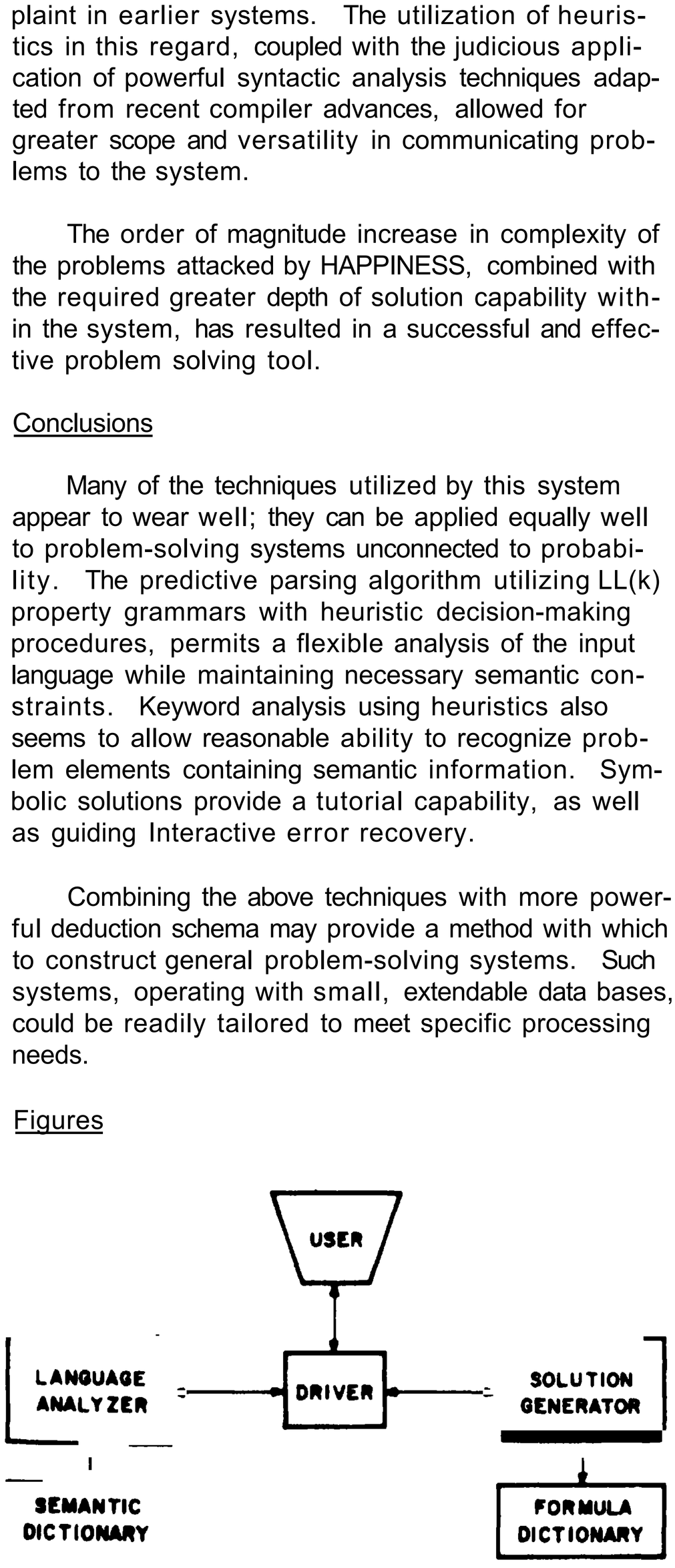
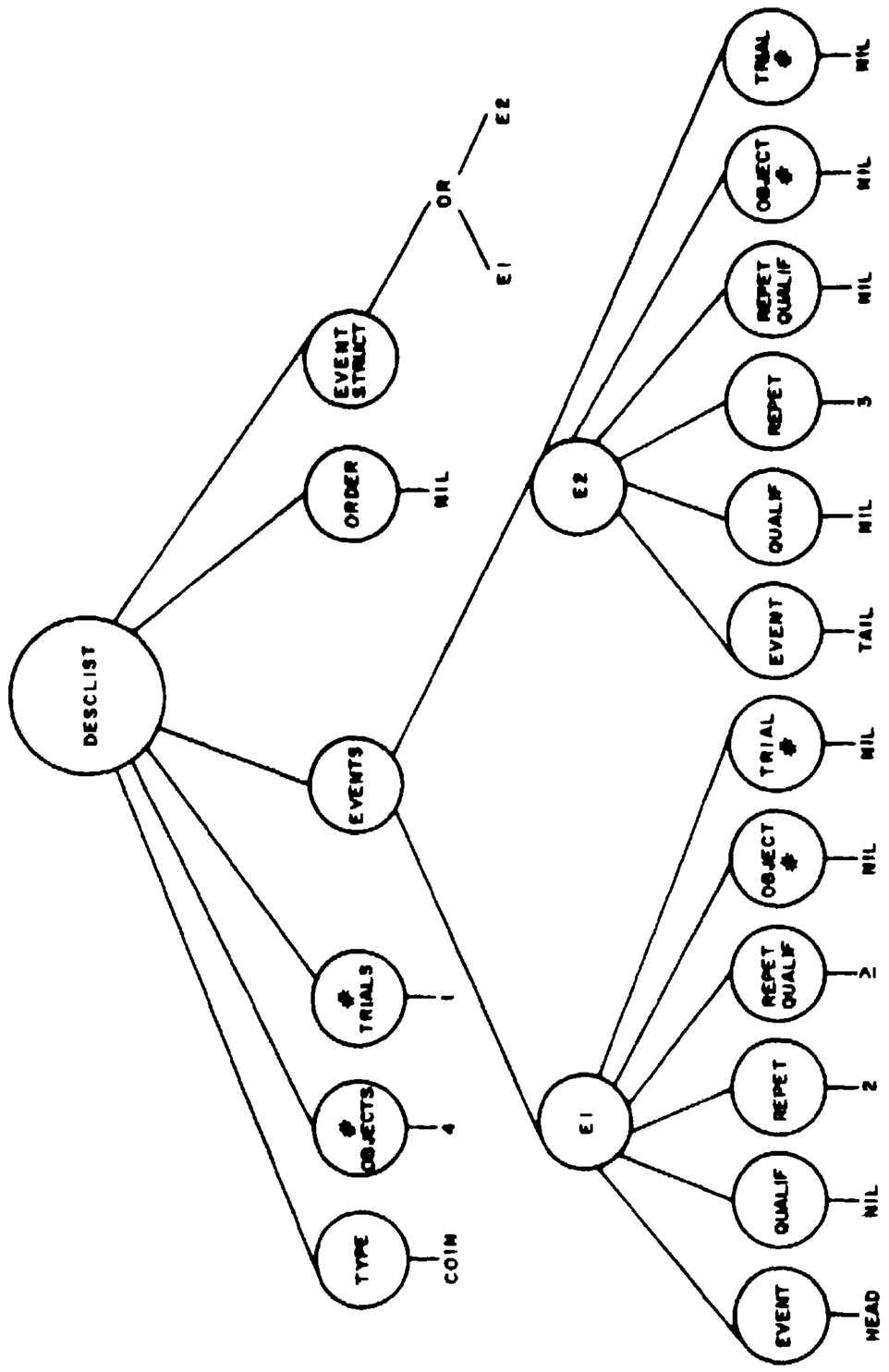


Figure 1



Examples

```
(PLEASE INPUT PROBLEM)
(A pair of die* are tossed twice '. What it the probability of getting 7 points on the first toss and 11 points on the second ?)

(A PAIR OF DICE ARE TOSSED TWICE . WHAT IS THE PROBABILITY OF GETTING 7 POINTS ON THE FIRST TOSS AND 11 POINTS ON THE SECOND ?)

(ELAPSED TIME NOW 0.08099997 SECONDS)

(AFTER PRIMARY (IDIOMATIC) TRANSFORMATION , PROBLEM BECOMES)
(2 DICE ARE TOSSED 2 TIMES . WHAT IS THE PROBABILITY OF GETTING 7 POINTS ON THE (1) TOSS AND 11 POINTS ON THE (2) ?)

(ELAPSED TIME NOW 0.329 SECONDS)

(AFTER SECONDARY (REARRANGING) TRANSFORMATION i)
(THE SIMPLE SENTENCES ARE)
((TOSS / PASSIVE) (2 DICE) (2 TIMES))
(THE QUESTION SENTENCE IS)
(WHAT IS THE PROBABILITY OF GETTING 7 POINTS ON THE (1) TOSS AND 11 POINTS ON THE (2) ?)

(ELAPSED TIME NOW 0.132 SECONDS)

(AFTER SYNTAX ANALYSIS OF QUESTION SENTENCE :)

(DESCLIST FOR THIS PROBLEM CONTAINS i)
   PROBLEMTYPE= DICE
   POPULATION= 2 OBJECTS
   SAMPLESIZEARIALS= 2
   SIMPLE EVENTS= ( G12480 G12479)
    G12480= ((11 POINT) NIL (1) NIL (2) NIL)
    G12479= ((7 POINT) NIL (1) NIL (1) NIL)
   COMPOUND EVENT STRUCTURE= (AND (OR G12479) (OR G12480))
   REPLACEMENT_INVOLVED? NO

(ELAPSED TIME NOW 0.656 SECONDS)

(FIRST LEVEL SOLUTION TO PROBLEM IS)
(PLUS F (PROB (QUOTE ( G12479 G12480))))
(TIME FOR EVALUATION WAS 0.032 SECONDS)
(SECOND LEVEL SOLUTION TO PROBLEM IS)
(PLUSFN (TIMESFN (PR (QUOTE G12479)) (PR (QUOTE G12480))))
(TIME FOR EVALUATION WAS 0.05 SECONDS)
(STATUS OF EVENTS WAS)
((( G12479 G12480) (INDEPENDENT MOTNUTUALLYEXCLUSIVE)))
(THIRD LEVEL SOLUTION TO PROBLEM IS)
(PLUSFRAC (TIMESFRAC (TIMESFRAC (FRACTION (NOOFOMENS 1 1) 1)
                                (POWERFRAC (1 6) 1) (POWERFRAC (5 6) 0))
          (TIMESFRAC (FRACTION (NOOFOMENS 1 1) 1) (POWERFRAC (1 18) 1)
                                (POWERFRAC (17 18) 0))))
(TIME FOR EVALUATION WAS 0.143 SECONDS)

(FOURTH LEVEL SOLUTION TO PROBLEM IS)
1/108 (OR 0.009259257)
(ELAPSED TIME NOW 0.065 SECONDS)

(TOTAL TIME FOR PROBLEM SOLUTION WAS 1.504 SECONDS)


(PLEASE INPUT PROBLEM)
(A box contains sixteen good and four defective fuses '. If ? are drawn ', what is the probability that they are defective ?)

(A BOX CONTAINS SIXTEEN GOOD AND FOUR DEFECTIVE FUSES . IF 2 ARE DRAWN , WHAT IS THE PROBABILITY THAT THEY ARE DEFECTIVE ?)

(ELAPSED TIME NOW 0.08200002 SECONDS)

(AFTER PRIMARY (IDIOMATIC) TRANSFORMATION , PROBLEM BECOMES)
(1 BOX CONTAINS 16 GOOD AND 4 DEFECTIVE FUSES . IF 2 ARE DRAWN , WHAT IS THE PROBABILITY THAT THEY ARE DEFECTIVE ?)

(ELAPSED TIME NOW 0.3 SECONDS)

(AFTER SECONDARY (REARRANGING) TRANSFORMATION i)
(THE SIMPLE SENTENCES ARE)
((CONTAIN / ACTIVE) (1 BOX) (16 GOOD AND 4 DEFECTIVE FUSES))
((DRAW / PASSIVE) (2) NIL)
(THE QUESTION SENTENCE IS)
(WHAT IS THE PROBABILITY THAT THEY ARE DEFECTIVE ?)
(ELAPSED TIME NOW 0.199 SECONDS)

(AFTER TERTIARY (CASE-REDUCING) TRANSFORMATION , THE QUESTION SENTENCE BECOMES :)
(WHAT IS THE PROBABILITY OF GETTING ALL DEFECTIVE ?)
(ELAPSED TIME NOW 0.06800002 SECONDS)
(NO RECOGNIZABLE KEYWORDS FOUND IN PROBLEM
 ASSUMED SAMPLING PROBLEM)
(AFTER SYNTAX ANALYSIS OF QUESTION SENTENCE :)
(ASSUMING THAT " 16 GOOD " MEANS " 16 GOOD FUSES ")
(ASSUMING THAT " 2 " MEANS " 2 FUSES ")

(DESCLIST FOR THIS PROBLEM CONTAINS i)
```

```
  PROBLEMTYPE= SAMPLING
  POPULATION= ((4 DEFECTIVE FUSES ) (16 GOOD FUSES ) )
                                        AS OBJECTS
  SAMPLESIZE/TRIALS= 2
  SIMPLE EVENTS: ( G12485)
   G1285= ((FUSE) (DEFECTIVE) (2) NIL NIL NIL)
COMPOUND EVENT STRUCTURE= (OR G12485)
  REPLACEMENT_INVOLVED? NO
(ELAPSED TIME NOW 0.469 SECONDS)

(FIRST LEVEL SOLUTION TO PROBLEM IS)
(PLUSF (PROB (QUOTE ( G12485))))
(TIME FOR EVALUATION WAS 0.028 SECONDS)
(SECOND LEVEL SOLUTION TO PROBLEM IS)
(PLUSFN (FR (QUOTE G12485)))
(TIME FOR EVALUATION WAS 0.026 SECONDS)
(THIRD LEVEL SOLUTION TO PROBLEM IS)
(PLUSFRAC (SIMPLIFYFRAC (LIST (COMBNL 4 2) (COMBNL 16 0))
                        (COMBNL 20 2)))
(TIME FOR EVALUATION WAS 0.07499999 SECONDS)

(FOURTH LEVEL SOLUTION TO PROBLEM IS)
3/95 (OR 0.03157895)
(ELAPSED TIME NOW 0.108 SECONDS)

(TOTAL TIME FOR PROBLEM SOLUTION WAS 1.37 SECONDS)


(PLEASE INPUT PROBLEM)
(From a torch containing 4 ferd and 3 brakky and 5 chartreuae
 warfela ', 3 art drawn '. What is the probability that 2 are
 chartrauaa and the other brakky 7)

(FROM A ZORCH CONTAINING 4 FERD AND 3 BRAKKY AND 5 CHARTREUSE
 WERFELS , 3 ARE DRAWN . WHAT IS THE PROBABILITY THAT 2 ARE
 CHARTREUSE AND THE OTHER BRAKKY 7)

(ELAPSED TIME NOW 0.09299999 SECONDS)

(AFTER PRIMARY (IDIOMATIC) TRANSFORMATION , PROBLEM BECOMES)
(FROM 1 ZORCH CONTAINING 4 FERD AND 3 BRAKKY AND 5 CHARTREUSE
 WERFELS , 3 ARE DRAWN . WHAT IS THE PROBABILITY THAT 2 ARE
 CHARTREUSE AND 1 BRAKKY 7)

(ELAPSED TIME NOW 0.384 SECONDS)

(AFTER SECONDARY (REARRANGING) TRANSFORMATION :)
(THE SIMPLE SENTENCES ARE)
((CONTAIN / ACTIVE) (1 ZORCH) (4 FERD AND 3 BRAKKY AND
                                5 CHARTREUSE WERFELS))
((DRAW / PASSIVE) (3) NIL)
(THE QUESTION SENTENCE IS)
(WHAT IS THE PROBABILITY THAT 2 ARE CHARTREUSE AND 1 BRAKKY 7)
(ELAPSED TIME NOW 0.227 SECONDS)

(AFTER TERTIARY (CASEREDUCING) TRANSFORMATION , THE QUESTION
                              SENTENCE BECOMES :)
(WHAT IS THE PROBABILITY OF GETTING 2 CHARTREUSE AND 1 BRAKKY 7)
(ELAPSED TIME NOW 0.108 SECONDS)
(NO RECOGNIZABLE KEYWORDS FOUND IN PROBLEM
 ASSUMED SAMPLING PROBLEM)
(AFTER SYNTAX ANALYSIS OF QUESTION SENTENCE :)
(ASSUMING THAT " 3 BRAKKY " MEANS " 3 BRAKKY WERFELS ")
(ASSUMING THAT " 4 FERD " MEANS " 4 FERD WERFELF ")
(ASSUMING THAT " 3 " MEANS " 3 WERFELS ")
(DESCLIST FOR THIS PROBLEM CONTAINS :)
  PROBLEMTYPE= SAMPLING
  POPULATION= ((5 CHARTREUSE WERFELS ) (3 BRAKKY WERFELS )
                (4 FERD WERFELS ) ) AS OBJECTS
  SAMPLESIZE/TRIALS= 3
  SIMPLE EVENTS: ( G12503 G12502)
   G12503= ((WERFELS) (BRAKKY) (1) NIL NIL NIL)
   G12502= ((WERFELS) (CHARTREUSE) (2) NIL NIL NIL)
COMPOUND EVENT STRUCTURE= (AND (OR G12502) (OR G12503)
  REPLACEMENT_INVOLVED? NO
(ELAPSED TIME NOW 0.715 SECONDS)

(FIRST LEVEL SOLUTION TO PROBLEM IS)
(PLUSF (PROB (QUOTE ( G12502 G12503))))
(TIME FOR EVALUATION WAS 0.032 SECONDS)
(SECOND LEVEL SOLUTION TO PROBLEM IS)
(PLUSFN (FR (QUOTE ( G12502 G12503))))
(TIME FOR EVALUATION WAS 0.034 SECONDS)
(THIRD LEVEL SOLUTION TO PROBLEM IS)
(PLUSFRAC (SIMPLIFYFRAC (LIST (COMBNL 5 2) (COMBNL 3 1)
                              (COMBNL 4 0))
                        (COMBNL 12 3)))
(TIME FOR EVALUATION WAS 0.14 SECONDS)

(FOURTH LEVEL SOLUTION TO PROBLEM IS)
3/22 (OR 0.1363636)
(ELAPSED TIME NOW 0.134 SECONDS)

(TOTAL TIME FOR PROBLEM SOLUTION WAS 1.882 SECONDS)
```

Bibliography

[1] Bobrow, D.G. "Natural Language Input for a Computer Problem Solving System," Ph.D. Thesis, Department of Mathematics, MIT, Cambridge, Mass.; 1964.

[2] Charniak, E. "Computer Solution of Calculus Word Problems, " Proc. Intl. Joint Confc, Artif. Intell., Washington, D.C.; May, 1969.

[3] Gelb, J. P. "The Computer Solution of English Probability Problems", Ph.D. Thesis, Computer Science Program, RPI, Troy, New York; 1971.

[4] Lewis, P.M. et al. Theory of Compiler Design, in preparation.

[5] Lindsay, R.K. "Inferential Memory as the Basis of Machines Which Understand Natural Language." In Feigenbaum, E.A. and Feldman, J. (eds.), Computers and Thought, McCraw-Hill, New York; 1963.

[6] Raphael, B. "SIR: A computer Program for Semantic Information Retrieval." In Minsky, M. (ed.), Semantic Information Retrieval, The MIT Press, Cambridge, Mass.; 1968.

[7] Simmons, R.F. "Answering English Questions by Computer: A survey," Comm. ACM, vol. 8, no. 1; January, 1965.

[8] Stearns, R.E., and Lewis, P.M. "Property Grammars and Table Machines," Information and Control, vol. 14, no. 6; June, 1969.