A Deductive Question-Answerer for Natural Language Inference

ROBERT M. SCHWARCZ, JOHN F. BURGER, AND ROBERT F. SIMMONS
System Development Corporation, Santa Monica, California

The question-answering aspects of the Protosynthex III prototype language processing system are described and exemplified in detail. The system is written in LISP 1.5 and operates on the Q-32 time-sharing system. The system's data structures and their semantic organization, the deductive question-answering formalism of relational properties and complex-relation-forming operators, and the question-answering procedures which employ these features in their operation are all described and illustrated. Examples of the system's performance and of the limitations of its question-answering capability are presented and discussed. It is shown that the use of semantic information in deductive question answering greatly facilitates the process, and that a top-down procedure which works from question to answer enables effective use to be made of this information. It is concluded that the development of Protosynthex III into a practically useful system to work with large data bases is possible but will require changes in both the data structures and the algorithms used for question answering.

KEY WORDS AND PHRASES: question answering, natural language, Protosynthex III, LISP, semantics, artificial intelligence, computational linguistics, language processing, fact retrieval
CR CATEGORIES: 3.42, 3.61

1. Introduction and Background

The development of systems for storing and retrieving information in structured files of data that can be interacted with through subsets of natural English has received considerable attention in recent years. In developing such a system that allows not only economical representation of and rapid access to facts stored in the system but also facile representation and thorough and efficient utilization of the logical relationships between these facts, while still permitting communication with the user in a flexible subset of natural English, three major interrelated problems must be solved:

(1) The development of a formalized data representation language in which the aspects of meaning that are relevant to the system's uses are explicitly and unambiguously represented;

(2) The development of algorithms for translating inputs in the English subset into appropriate storage and retrieval commands for operating on the data base, making appropriate use of contextual information to resolve both syntactic and semantic ambiguity, and for translating structures retrieved from the data base into well-formed English answers;

(3) The development of algorithmic and/or heuristic methods for determining what subset of the data base is relevant to filling a given retrieval request and, using the logical relationships represented in the data base, for deducing answers from this subset to meet the request.

The development of an adequate data representation language for any application is, of course, a direct consequence of the particular information handling requirements of the system and the particular set of programming tools available for use. In research not directed toward any particular application this language is taken either as a matter of convenience or as a consequence of some semantic or cognitive theory which its authors are attempting to implement on the computer. The solutions to the problems of linguistic analysis and question answering depend on the data representation language chosen, although the language chosen may in turn be influenced by the linguistic analysis and question-answering algorithms desired; for example, the first-order predicate calculus may be chosen as the form of data representation because of the powerful deductive inference procedures that exist for it.

In general, methods for linguistic analysis and methods for question answering have been developed largely independently of each other. Little relation has been seen between the semantic interpretation and generation systems that constitute the man-machine interface, on the one hand, and the deductive system, which embodies the basic data retrieval capability of the system as a whole, on the other. Their use of a common formal language for data structure representation has been, in general, the only such relation. In fact, most researchers to date have concentrated their attention on only one of the two problems, addressing themselves only minimally to the other. For example, the DEACON system of Thompson et al. [28], the DISEMINER system of Klein et al. [13], and the grammar-based question-answering system of Rosenbaum [23] have sophisticated procedures for analyzing English sentences but pay insufficient attention, if any, to the matter of deductive inference. Levien and Maron [14], Elliott [7], and Green and Raphael [10] have developed data-base

This research was supported in part by Contract No. F33615-67-C-1986, Computer-Aided Instruction Using Natural Language, with the Training Research Division, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base. Reproduction in whole or in part is permitted for any purpose of the United States Government.

question-answering systems with powerful inference-making capabilities but have only recently begun to address the problem of providing their users with a flexible natural language interface. Darlington's [6] system for proving logical arguments expressed in English and Coles's [5] system for interfacing natural language with graphical information processing embody both algorithms for translating subsets of English into the first-order predicate calculus and proof procedures for the resulting predicate calculus expressions; but in both instances the two subsystems are separate—neither system represents a data-base question-answering system in the usually understood sense of the term. (Coles, however, has recently [5] integrated his natural language interface with the Green-Raphael data-base question-answerer, their common use of the predicate calculus as a data representation language providing the basis for the marriage.) Furthermore, none of these systems is capable of expressing answers to retrieval requests in a flexible subset of English.[1]

In Protosynthex III, the capabilities for sentence analysis and generation, lexical and syntactic paraphrase, and deductive question answering are combined for the first time in a single system. Furthermore, the deductive mechanisms used in question answering have been developed, not independently of the machinery for sentence analysis and generation, but specifically to make use of the same structures of interpretive semantic information that are used in these other operations. Our approach assumes that the subset-superset relations, semantic event forms, and conceptual paraphrase equivalences utilized in both question answering and other operations represent general information about the universe of discourse, and that this information has both functional utility for interpreting and generating natural language and logical significance in the deductive inference process (as opposed to Katz and Postal [12], for example, who treat such information as merely convenient devices for use in disambiguation, paraphrase generation, etc.). The conception of language on which these assumptions are based is that the correspondence between linguistic and conceptual structures is given by a formal nondeterministic[2] algorithm (as represented by two sets of grammar rules, one for interpretation and one for generation) phrased entirely in terms of the syntactic categories and features of the language, while the set of admissible conceptual structures (which, in conjunction with the grammar rules, determines the set of admissible linguistic structures) is determined by what can be subsumed under the system's classification of possible events in its universe of discourse.[3] Furthermore, entirely apart from the question of its linguistic validity, this conception carries the happy consequence (as we shall show below) that the conceptual classification used in semantic analysis and paraphrase may be used in guiding the search process of deductive inference so as to obtain considerably increased efficiency in question answering. We have described the linguistic aspects of Protosynthex III in [24]; it is our purpose here to describe and illustrate our method of deductive question answering and its interaction with the linguistic aspects of the system.

2. The Formal Data Structure

2.1 Concepts, Events, and Relations. The formal data structure that lies at the heart of the Protosynthex III system is a graph structure made up of event triples of the form (X R Y), where each X, R, or Y is either a primitive element of the system, a lexically defined concept (i.e. a concept corresponding to an English word sense), a "token" (instance) or "paraphrase" (refinement by inclusion in additional relational triples) of a lexically defined concept, or another event triple.[4] The X and Y of a triple of specific data must be either tokens or other data triples; the R of a data triple may be a primitive, a lexically defined concept, or a token. Triples containing other combinations of elements are used to represent the various forms of general information about the system's universe of discourse that it uses for semantic processing and deductive inference.

Each entity is represented in the data structure by an atomic symbol generated internally by the Lisp 1.5 program, and has on its property list pointers to the event triples it is used in as X, R, and Y, respectively. If it represents an event triple, it also has pointers to the X, R, and Y of that triple. For communication among many of the system's routines and for direct input by the user to the system's data structures, a form of nested bracketing is used in which a square-bracketed [X R Y] denotes the event triple itself, whereas a round-bracketed (X R Y) denotes the X term modified by being in the relation R to Y. For example, the expression ((X1 R1 Y1) R3 [X2 R2

[1] However, Teitelman [27] has implemented question-answering routines in his PILOT system that return their output in a subset of English. He limits himself, however, to simple format insertion methods and has not solved the problem of English input. In this respect he follows Raphael's SIR system [21] and Bobrow's STUDENT system [1], both of which exhibit—along with good deductive capability in narrowly circumscribed domains—a capability for input and output in a limited subset of English than is achieved through format matching and insertion rather than through linguistically motivated semantic analysis and generation procedures (though Bobrow's use of formats in a recursive manner does derive somewhat from early versions of transformational theory).

[2] By "nondeterministic" here it is simply meant that the sequential order of application is not specified in the sets of rules but is dependent on the particular parsing and generation programs that utilize the grammars.

[3] Though different in many respects, this conception has a considerable area of agreement with the more recent conceptions of Chomsky [3] and Katz [11].

[4] This representation is isomorphic to a directed graph with labeled edges in which the concepts, events, and relations are nodes and the edges, which connect an event to its X, R, and Y, are labeled, respectively, "has-X", "has-R", and "has-Y". It is *not* isomorphic to a directed graph in which the X and Y terms represent nodes and the R terms are labels on edges, since both events and their R terms may be X or Y terms of other events.

Y2]) would denote X1 in the relation R1 to Y1 and also in the relation R3 to the event [X2 R2 Y2], as in the phrase "young boys who watch men sail ships", which could be represented as ((BOYS MOD YOUNG) WATCH [MEN SAIL SHIPS]).[5] A third form of bracketing, angled brackets, is used to input semantic event forms (SEFs), such as ⟨ANIMAL EAT FOOD⟩, which represent classes of events that are semantically interpretable by the system. The form of bracketing of each event triple is represented in the data structure by the value of a "dependency" property on the property list of the associated symbol.

2.2 Semantic Organization: SUPchains, SEFs, and Paraphrase Equivalences. The lexically defined concepts of the system are related semantically to each other in three distinct ways. First, every concept is in a SUP (superset, or "is-a-kind-of") relation to at least one other concept and may also be in an EQUIV (equivalence) relation to one or more concepts, concept paraphrases, or event triples. From every concept, then, one or more chains of SUP and EQUIV relations lead up to the primitive *TOP, which is by definition the most general concept in the system. The union of all such chains leading up from a given concept is called the SUPchain of the concept. Similarly, for every concept there is a SUBchain consisting of all concepts that have that concept on their SUPchain plus all tokens, concept paraphrases, and event triples subordinate[6] or equivalent to these concepts. The configuration of SUP and EQUIV relations represents a taxonomy of the lexically defined concepts in the system's universe of discourse at any time.

The semantic event forms, or SEFs, represent the second type of semantic relation between concepts. Each SEF ⟨X R Y⟩ defines as semantically interpretable events (or what might be thought of as possible events) in the system's universe of discourse, all ordered triples consisting of an element of the SUBchain of X, an element of the SUBchain of R, and an element of the SUBchain of Y. In sentence analysis, the SEFs resolve both syntactic and lexical ambiguity by eliminating interpretations that contain one or more semantically uninterpretable event triples; in this, the SEFs function in much the same way as Katz and Postal's selectional restrictions.

The third type of semantic relation is that of equivalence among concepts and concept paraphrases. Equivalence information is used in sentence generation to determine the various alternative phrasings of a concept, or modificational (round-bracketed) triple of concepts, that can be input to the syntactic transformational rules that produce the surface form of the sentence.

2.3 Deductive Logic: Relational Properties and Operators. The deductive properties of the Protosynthex III formalism are expressed in terms of five properties on relations and five operators for combining relations into more complex relations. The converse operator on relations (called INVERSE in the program because of a misnaming early in the history of the program that was simply too much trouble to correct) is also employed as part of the formalism.

The five relational properties are reflexivity, symmetry, transitivity, left-collapsibility, and right-collapsibility. They are defined as follows:

Reflexivity (REFL): If R is reflexive and (A R A) is a semantically interpretable event, then (A R A) is true.

Symmetry (SYMM): If R is symmetric and (A R B) is true, then (B R A) is true.

Transitivity (TRANS): If R is transitive and both (A R B) and (B R C) are true for some B, then (A R C) is true.

Left-collapsibility (LCOL): If R is left-collapsible and (A R B) is true, then for any question triple containing B the event triple derived by substituting A for B constitutes a valid answer. Any product of left-collapsible relations is also left-collapsible.

Right-collapsibility (RCOL): If R is right-collapsible and (A R B) is true, then for any question triple containing A the event triple derived by substituting B for A constitutes a valid answer. Any product of right-collapsible relations is also right-collapsible.

The converse of any reflexive, symmetric, or transitive relation is also reflexive, symmetric, or transitive, respectively. The converse of any left-collapsible relation is right-collapsible, and vice versa. Furthermore, any left-collapsible or right-collapsible relation is also transitive. Reflexivity, symmetry, and transitivity are properties that should be familiar to anyone who has studied logic or abstract algebra. Left-collapsibility is the property used to characterize the SUP relation and other relations (including TKOF and PAROF) that behave like it logically; right-collapsibility is the property characterizing the SUB relation (the converse of SUP) and other relations that behave like it logically. EQUIV, of course, is both left-collapsible and right-collapsible. In addition, SUP and SUB are defined as reflexive, and EQUIV as both reflexive and symmetric. The properties of left-collapsibility and right-collapsibility are intimately related to the notions of SUPchain and SUBchain; for the SUPchain of a concept is simply the set of all elements to which the concept is in any left-collapsible relation, and the SUBchain of a concept is the set of all elements to which the concept is in any right-collapsible relation (this is, in fact, the way that SUPchains and SUBchains are computed in the program).

[5] In the actual internal representation, the English words would be replaced by generated concept or token symbols; only primitives like MOD would remain the same. For the sake of clarity in our explanations, however, we shall stay with English words in the text examples. Examples of internally generated nodes appear in the sample computer printouts in the Appendix.

[6] Each token is connected by the relation TKOF to the concept of which it is an instance, and each concept paraphrase is connected by the relation PAROF to the concept whose lexical expression is the head of the phrase that expresses the paraphrase. The relation SUP is used to subordinate event triples to concepts.

Relational operators are the means employed in the formalism, in conjunction with the SUP and EQUIV relations, to indicate the logical relations between relations. Such a relation between relations, called an "inference rule," is formed by setting a complex relation formed with one or more relational operators in a SUP or EQUIV relation to an atomic (primitive or lexically defined) relation, as in the examples given below. A complex relation is represented in the form of a square-bracketed event triple [R1 OP R2], where R1 and R2 are relations and OP is a relational operator. Since the relations that a relational operator combines can themselves be complex relations, complex relations may be nested to arbitrary depth. The five relational operators that the system employs are complex (or relative) product, intersection, modification, restricted domain, and restricted range. They are defined as follows:

Complex product (C/P): If (A R1 C) and (C R2 B) are both true for some C, then (A [R1 C/P R2] B) is true.
Intersection (S*AND): If (A R1 B) and (A R2 B) are both true, then (A [R1 S*AND R2] B) is true.
Modification (S*MOD): If (A (R1 R2 B) C) is true for some C, then (A [R1 S*MOD R2] B) is true.
Restricted domain (RSTL): If (A R B) is true and A is on the SUBchain of D, then (A [R RSTL D] B) is true.
Restricted range (RSTR): If (A R B) is true and B is on the SUBchain of D, then (A [R RSTR D] B) is true.

Since inference rules are represented in the system as nested event triples just like any other piece of data or semantic information, they may, if the system has been given an adequate grammar, be entered into the system not only directly but by means of English sentences. In the following examples of inference rules, the grammar required to get from the English sentences to the nested event triples would be simple indeed to implement:

(a) To be the brother of the father of is to be the uncle of. [[BROTHER C/P FATHER] SUP UNCLE]
(b) To be above and touching is to be on-top-of.[7] [[ABOVE S*AND TOUCHING] SUP ON-TOP-OF]
(c) To travel to is to visit. [[TRAVEL S*MOD TO] SUP VISIT]
(d) To be part of the group to cause is to aid. [[PART C/P [CAUSE RSTL GROUP]] SUP AID]
(e) To be leader of the group to lose a contest is to lose. [[LEADER C/P [[LOSE RSTR CONTEST] RSTL GROUP]] SUP LOSE]
(f) To give a good thing to is to reward. [[[GIVE RSTR (THING MOD GOOD)] S*MOD TO] SUP REWARD]

Examples of inference rules actually input to the computer in English are given in the third example in Section 4.1 and the Appendix.

[7] The program does not as yet possess the capability of treating multiword idioms as single lexical items.

There is one other way in which complex relations can be used in the system. A complex relation defined with RSTL and RSTR as its only relational operators can be given the property LCOL or RCOL with appropriate results; for example, to define "to be a kind of" as left-collapsible, one would input directly to the system the nested triple [[OF RSTL KIND] PROPERTY LCOL].

One may perhaps question the inferential power of a deductive formalism that is obviously lacking in two essential features of a logistic system—negation and quantification—without which the formalism could not conceivably be regarded as logically complete. The omission of negation, in particular, limits substantially the range of facts that can be stored, questions that can be asked, and inferences that can be handled. Negation could, however, be handled without excessive difficulty by a complement operator for relations implemented just as the converse operator has been implemented. The one additional complication this would introduce is the possibility of logical inconsistencies in the data structure; in the present system either a question is answered positively or it is not answered at all. To automatically reject input statements that would render the data base inconsistent, as Elliott [7] has done for his system, would require a good bit of additional processing. Perhaps a better method, and one closer to what most people evidently use, is one which, like the "exception principle" that Raphael [21] employs, gives preference to more specific statements over more general statements, and to shorter inference paths over longer inference paths, in choosing among an inconsistent set of answers to a question. It would be left to the user, in this case, not to introduce more inconsistencies into the data base than the system could reasonably deal with.

With regard to quantification, the SUPchain logic provides this to a sizable—if still limited—extent, for "all X" can be represented in the structure by the concept X, and "some X" can be represented by a token of X (i.e. in the relation TKOF to X). The limitations are that the order of quantification has no means of representation (for example, "Every ocean is sailed by some ships." could not be represented in its true meaning but would have the same representation as "Some ships sail every ocean.", i.e. as [(T101 TKOF SHIPS) SAIL OCEAN]),[8] and that the count-mass distinction is lost.

The deductive formalism also lacks the capability of transforming one complex relation into another (inference rules only relate complex relations to atomic relations); this capability is sometimes to be desired. Despite these deficiencies, however, the deductive formalism and the question-answering logic based on it possess a good bit of inferential power, as is demonstrated in the examples that follow.

[8] This only represents the way it is stored internally in the structure; what is passed from the semantic analyzer to the storage mechanism has the form [(SHIPS TMOD SOME) SAIL (OCEAN TMOD EVERY)].

3. Process of Question Answering

3.1 Input and Output. The input to the question-answering algorithm is a nested triple of concepts produced from the question by the semantic analyzer. The nested triple may contain TMODs (determiners) including "what" or "which"; these are stripped off the head concept on entry of the question, simply because the question-answerer treats every concept in every question as carrying an implicit "what" determiner. Accordingly, "who", "what", "when", "where", and other interrogative pronouns are treated by setting their concepts equivalent in the structure to the concepts of the corresponding declarative nouns: "person", "thing", "time", "place", etc.

The output delivered by the question-answering algorithm to the sentence generator is a (possibly empty) list of answers, each of which is, in general, in the exact form of the question but with the terms of the question being replaced by elements of their SUBchains, and with square-bracketed question triples (other than the outermost triple of the entire question) being replaced by the symbolic names of the event triples found to correspond to them.[9] The sentence generator may then, at the user's discretion, express (either completely or in abbreviated form) or paraphrase each of the answers.

3.2 Basic Procedure. The method of deductive inference used in the question-answerer is basically a modified "top-down" approach inspired by the heuristic search methods developed by Newell and Simon in their LT [16], and GPS [17] theorem-proving and problem-solving programs. The method is similar in design to (although conceived independently of) the MULTIPLE program of Slagle and Bursky [26] in that, rather than employing either a depth-first or breadth-first search procedure, it follows at every choice point the path that it evaluates as having the greatest likelihood of success. The evaluation is made in terms of a heuristic "path length to solution" measurement that is described in Section 3.3.

For each question presented to it, the question-answerer develops a hierarchical structure of goals and subgoals of a form quite similar to the structure of data in memory but with some additional features of functional importance to the question-answering process; the original question is the topmost goal on this structure. The first step in the question-answering process is to break up the original question and each conceptual paraphrase of it into their constituent triples (with sets of "agreement conditions" being generated to indicate the sets of elements of different triples that must correspond in any combination of answers). The set of triples resulting from each such decomposition is then entered into the structure as a "subgoal expansion" of the original question. Each constituent triple, or subgoal, is then subjected to a "generalized direct lookup" procedure that finds all answers that can be inferred from it by direct match and the use of properties on relations. If every subgoal in any expansion of the question is answered by generalized direct lookup and the agreement conditions are met for at least one combination of these answers, the question has been answered, and the resulting answers are passed on to the sentence generator.[10]

If not every subgoal is answered by generalized direct lookup, the heuristic search procedure is begun in an attempt to derive answers by means of inference rule expansions. Each step in this search procedure consists in choosing a subgoal, in the expansion with the minimum "path length to solution", that has not yet been expanded by inference rules and in applying the inference rules that can be applied (as determined by whether the relation in the right half of the inference rule is on the SUBchain of the subgoal's R term or its converse) to generate expansions of this subgoal (according to the definitions of the relational operators in Section 2.3). The expansions generated are then entered, along with their conceptual paraphrases, into the goal hierarchy and tested by generalized direct lookup in the same manner as the expansions of the original question. If all the subgoals of any expansion are answered and the generated agreement conditions are met for at least one combination of answers, the original subgoal has been answered and its answers are passed upward in the goal hierarchy.

This process of subgoal expansion and inference is iterated until an answer is found to the original question by solution of one of its expansions, or until no more subgoal expansions can be made (i.e., no inference rules can be applied to expand any as-yet-unexpanded subgoal), or until every expansion of the original question grows to a "path length to solution" that exceeds a certain preset maximum. In the last two cases, although no answer is found, termination is still achieved fairly rapidly, largely because of the reduction in the extent of search that is achieved by the employment of semantic information in directing the search process (as described in Section 3.3).

In termination on any of the three criteria, the subgoal structure is erased from memory and the (possibly empty) set of answers obtained is passed on to the sentence generation program.

3.3 Details of the Procedure: An Illustrative Example. To illustrate and explain in greater detail just how the system answers questions, we consider an example that is fairly trivial in terms of what the system is capable of handling but whose solution illustrates all the major operations involved in the question-answering process. Suppose that the system had as part of its data-

[9] The one exception to this rule is where a term or triple of the question has been paraphrased on entry to the question-answering goal structure (as will be indicated later) and this paraphrase has been successfully answered, in which case the answer is in the form of the paraphrase.

[10] If every subgoal of an expansion is answered but the agreement conditions are not met, the subgoals for which the agreement conditions failed are "reactivated," or treated as though they had not yet been answered, so that more answers to them might be found through inference.

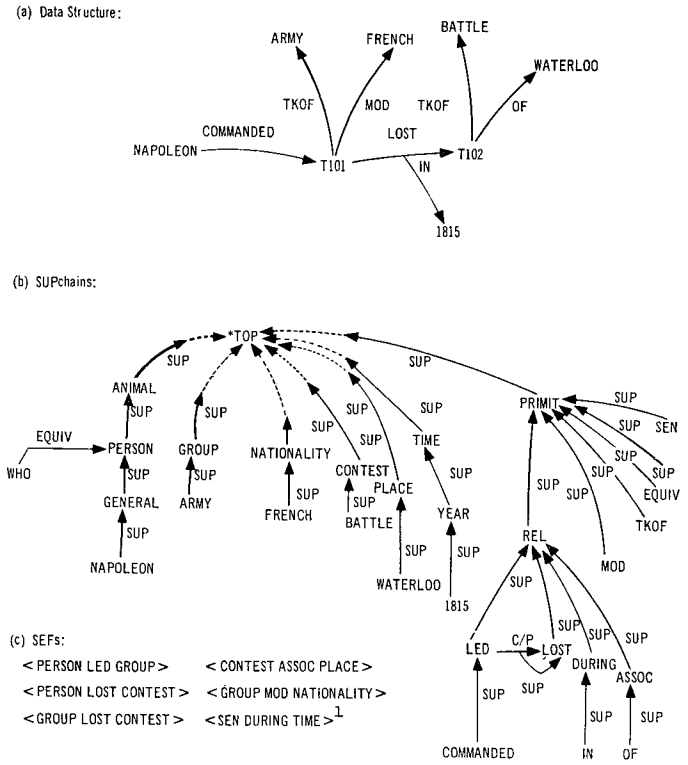

FIG. 1. Information structures for illustrative example. (The primitive SEN is used as a "universal" SUP for all event triples.)

base the structure in Figure 1(a), corresponding to the sentence "Napoleon commanded the French Army which lost the battle of Waterloo in 1815.", and as part of its semantic information the SUPchain structure in Figure 1(b) (including the inference rule [[LED C/P LOST] SUP LOST]) and the SEFs in Figure 1(c). Let us now look at the operations that would be performed in answering the question "Who lost the battle of Waterloo?", which would, after semantic analysis and deletion of the definite article, be transformed into a nested triple of concepts representing the structure (WHO LOST (BATTLE OF WATERLOO)).[11]

The goal hierarchy is begun by entering the question as the top goal SG0, with the property list structure shown in Figure 2(a). The question is stored under the property IS of SG0, and the properties MPATHL (the "measured path length") and DPATHL (the "dynamic path length," which changes as subgoal expansions are generated) are both set to MAXR, the maximum allowable path length of a subgoal expansion, which we take here to be seven (it may be set by the user to any value he desires).

The question is then broken up into its constituent triples (WHO LOST BATTLE) and (BATTLE OF WATERLOO), which are entered as subgoals SG1 and SG2, respectively. The agreement condition ((SG1 3)·(SG2 1)), which specifies that the values found for BATTLE in the answers to the two subgoals must be either the same or linked via a SUPchain (in which case the lower or more specific value is used), and the answer form ((SG1 1) (SG1 2) ((SG2 1) (SG2 2) (SG2 3))), which prescribes the construction of an answer to the question from answers to its subgoals, is generated. Generalized direct lookup is then done on both subgoals to find whatever answers are reachable directly in the data structure.

In generalized direct lookup on a question triple, the SUBchains of the X, R, Y, and R-converse of the triple

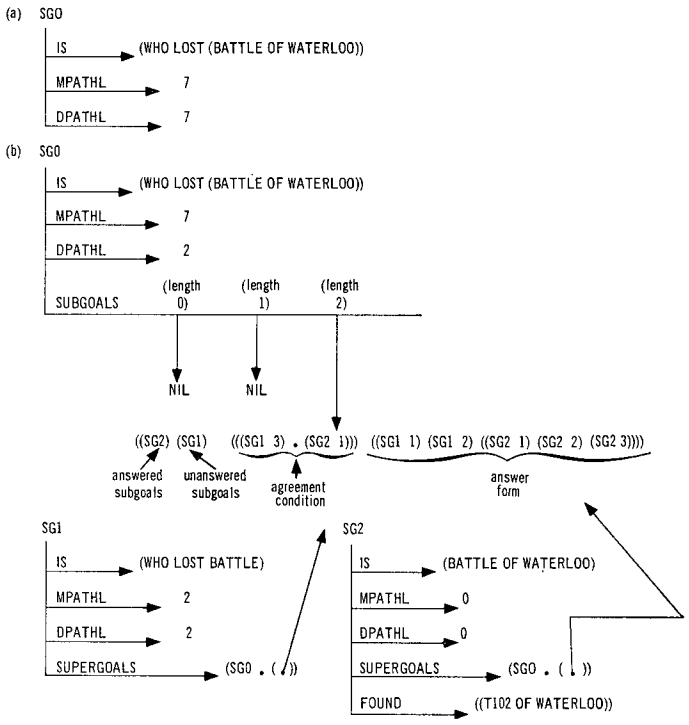

FIG. 2. Goal structures in stages of example

and the SUPchains of the X and Y are computed. Several direct lookups are then done on the concept and data structures and result in the following being returned as answers:

(1) All stored concept or data triples (X′ R′ Y′), where X′ and Y′ are on the SUBchains of X and Y, respectively, and either R′ is on the SUBchain of R or its converse is on the SUBchain of R-converse.[12]

(2) All triples (X R′ Y′) where R′ and Y′ are on the SUBchains of R and Y, respectively, and (X″ R′ Y′) is a

[11] Note here that "Who" means specifically "What person".

[12] In the case of a square-bracketed question triple, this is the only lookup done, since square-bracketing means that the name of an existing triple must be returned.

stored concept or paraphrase triple for some X″ on the SUPchain of X.

(3) All triples (X′ R″-converse Y), where X′ and R″ are on the SUBchains of X and R-converse, respectively, and (Y″ R″ X′) is a stored concept or paraphrase triple for some Y″ on the SUPchain of Y.[13]

(4) All triples (Z′ R′ Z′) where Z′ is on the SUBchain of both X and Y and R′ is a reflexive relation on the SUBchain of R.[14]

(5) All triples (X′ R′ Y′), where X′ and Y′ are on the SUBchains of X and Y respectively, R′ is a transitive relation on the SUBchain of R or the converse of a transitive relation on the SUBchain of R-converse, and there exists a sequence of data or concept triples through which X′ is ultimately connected to Y′ and each of whose relational terms is either R′ or an element of its SUBchain.

Thus in our example, for the subgoal (WHO LOST BATTLE) the SUBchains WHO-PERSON-GENERAL-NAPOLEON, LOST-[LED C/P LOST], BATTLE-T102, and LOST-converse are generated for the X, R, Y, and R-converse terms respectively, and the SUPchains WHO-PERSON-ANIMAL-...-*TOP and BATTLE-CONTEST-...-*TOP are generated for the X and Y terms. No triples are found in the structure to satisfy any of the five criteria, so this subgoal does not answer by generalized direct lookup. For the subgoal (BATTLE OF WATERLOO), the X, R, Y, and R-converse SUBchains are computed as BATTLE-T102, OF, WATERLOO, and OF-converse respectively, and the X and Y SUPchains are BATTLE-CONTEST-...-*TOP and WATERLOO-PLACE-...-*TOP respectively. Here the data triple (T102 OF WATERLOO) is found and returned as an answer.

Since subgoal SG1 did not answer by generalized direct lookup, the next step is to measure its path length (since SG2 was answered, its path length is automatically 0). The path length is computed as being 1 if there is a triple containing elements of the SUBchains of both the X term and the Y term, 2 if a triple containing an element of the SUBchain of the X term has an element in common with a triple containing an element of the SUBchain of the Y term and the common element is not an R term in both triples, and 3 otherwise. In either of the first two cases, the R terms of all the relevant triples must be reachable from the R term of the subgoal through some combination of SUBchains and inference rules. In our example, since NAPOLEON is on the SUBchain of WHO and T102 is on the SUBchain of BATTLE, T101 is included in both (NAPOLEON COMMANDED T101) and (T101 LOST T102), and COMMANDED and LOST are both reachable from LOST, the measured path length of subgoal SG1 (WHO LOST BATTLE) will be 2.

The expansion consisting of SG1 and SG2 is now stored under the value of the property SUBGOALS on SG0, and pointed back to from the property SUPERGOALS on both SG1 and SG2. Since this expansion has a total path length of 2, it is stored in the appropriate "bucket" underneath the property SUBGOALS on SG0 (the "bucket" structure orders expansions of a goal in terms of increasing path length), and the value of DPATHL for SG0 is changed appropriately to 2. The goal structure developed to this point in the process is shown in Figure (2b).

The next step in the process is the expansion by inference rules of an appropriately chosen unanswered subgoal of the top goal, SG0. The first (and only) subgoal expansion under the property SUBGOALS of SG0 has as its first (and only) unanswered subgoal SG1. Since SG1 has not yet been expanded, it is chosen as the goal to expand; otherwise, the selection process would have been repeated on the first unanswered subgoal of its first (minimum-path-length) subgoal expansion. The set of inference rules that apply to expand SG1 is found by taking the SUBchains of the relational term LOST and its converse and using all complex relations on either of these SUBchains to generate possible expansions of SG1. In this instance the complex relation [LED C/P LOST] is found on the SUBchain of LOST and nothing is found on the SUBchain of LOST-converse; therefore, [LED C/P LOST] generates the only expansion of SG1.

Following the definition of the C/P operator in Section 2.3, the expansion of SG1, or (WHO LOST BATTLE), generates a pair of subgoals of the form (WHO LED C) and (C LOST BATTLE) for some C. It is here that the semantic information incorporated in the set of SEFs is brought into play, for it is desired, in question answering by inference, to search only those areas of the data that correspond to questions that the system can interpret as meaningful. (The value of using meaningfulness in this way as a search-limiting heuristic is indeed considerable, as will be illustrated in example 2 of Section 4.) In the case of the C/P operator, as in this example, one or more concepts that will make both triples of the expansion meaningful must be found to substitute for the middle term C.[15] To find this set of appropriate concepts for the middle term, the set of Y terms from SEFs that "cover" the first subgoal of the expansion in their X and R positions (i.e. whose X and R terms are on the SUPchains of the respective X and R terms of the new subgoal) is computed, and likewise the set of X terms from SEFs that "cover" the second subgoal in their R and Y positions. The set of common terms is then taken as the intersection of these two sets, or, if this intersection is empty, it is

[13] The lookups in (2) and (3) are to determine answers from paraphrase equivalences, such as deriving "Fred is unmarried." from the datum "Fred is a bachelor." and the equivalence of "bachelor" to "unmarried man".

[14] Such triples are included only if no answers are obtained from (1), (2), or (3), in order that trivial answers be disregarded where nontrivial answers exist.

[15] In expansions employing the other relational operators, the SEFs serve either as simply a meaningfulness check or (in the case of the S*MOD operator) to fill in a term in one of the generated subgoals.

taken to be the union of the set of elements of the first set that are also on the SUBchain of some element of the second set and the set of elements of the second set that are also on the SUBchain of some element of the first set. In our example the SEF ⟨PERSON LED GROUP⟩ is found to cover (WHO LED C), and the SEFs ⟨PERSON LOST CONTEST⟩ and ⟨GROUP LOST CONTEST⟩ are found to cover (C LOST BATTLE); the set of common terms thus consists of the intersection of the sets {GROUP} and {PERSON, GROUP}, or the single concept GROUP. The application of the inference rule therefore yields one expansion of SG1, consisting of (1) the subgoals SG3, or (WHO LED GROUP), and SG4, or (GROUP LOST BATTLE); (2) the agreement condition ((SG3 3)·(SG4 1)), which designates the correspondence that must hold between elements that substitute for

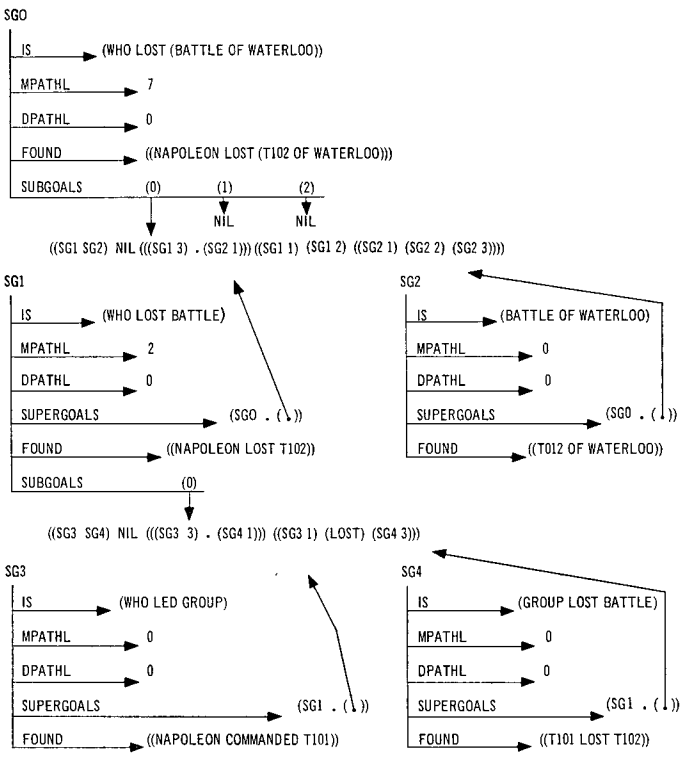

FIG. 3. Completed goal structure for example

GROUP in answers to the two subgoals; and (3) the answer form ((SG3 1) (LOST) (SG4 3)), which indicates how answers to SG1 are put together from pairs of answers to SG3 and SG4 that satisfy the agreement condition.

Upon their entry into the structure, now, generalized direct lookup is done on the two new subgoals—and yields answers to both, with (NAPOLEON COMMANDED T101) being found as an answer to SG3, and (T101 LOST T102) being found as an answer to SG4. The agreement condition on the two answers is satisfied (since T101 is identical with itself), and therefore (NAPOLEON LOST T102) is derived as an answer to SG1. Now SG1 and SG2 have both been answered, and the agreement condition on their answers is satisfied (since T102 is identical with itself); the answer (NAPOLEON LOST (T102 OF WATERLOO)) is thereby inferred for the top goal SG0. The values of DPATHL for SG1 and SG0 are changed to 0, of course, once these goals have been answered. The question has now been answered, and after the goal structure is erased from memory the answer is passed on to the sentence generator, which will express it as something like "Napoleon lost battle of Waterloo." The completed goal structure, just prior to its erasure, is shown in Figure 3.

The importance of semantic information in the question-answering process can be seen here in the use of SUPchains (which are computed from both SUP links and paraphrase equivalences) and SUBchains in generalized direct lookup and in the use of SUPchains, SUBchains, and SEFs in subgoal expansion by inference rules. The great power of generalized direct lookup (as opposed to a simple direct-match procedure) for drastically reducing the amount of search involved in inferential question-answering derives, in the main, from its use of SUPchains and SUBchains.

For what remains to be done of inference by explicit subgoal generation, the requirement of meaningfulness for the generated subgoals provides a quite powerful search-limiting heuristic. The examples in Section 4 illustrate the efficacy of these methods in answering both simple and complex natural English questions.

4. Performance

4.1 EXAMPLES OF SUCCESSFUL QUESTION ANSWERING. The three examples presented in this section represent a fair cross-section of the kinds of question answering that the system is capable of. The first two questions were among a set of 15 chosen randomly from a volume of Compton's Encyclopedia (others of the set that were not answerable are discussed in Section 4.2); the text from which each answer is derived is the relevant portion of the actual text from Compton's, paraphrased slightly in order that it might be analyzed with a simpler recognition grammar. The third example is adapted from an Advice Taker problem proposed by McCarthy [14].

1. *Text.* The stones and iron that fall to earth from outer space are called meteorites.

Analysis. [[((STONES ((FALL TO EARTH) FROM (SPACE MOD OUTER))****) TMOD THE) CALLED METEORITES] AND [((IRON ((FALL TO EARTH) FROM (SPACE MOD OUTER))****) TMOD THE) CALLED METEORITES]]

Semantic Information. (METEORITE EQUIV METEORITES), (WHAT EQUIV OBJECT), (STONES SUP

OBJECT), (IRON SUP OBJECT), (METEORITES SUP OBJECT), (CALLED SUP EQUIV), etc.[16]

Question. What is a meteorite?

Analysis. [(METEORITE TMOD A) EQUIV WHAT]

Answers. Meteorite be meteorites.[17] Meteorites be meteorite. Iron that fall to earth from outer space be called meteorites. Stones that fall to earth from outer space be called meteorites.

Explanation. The answers to this question are all found by generalized direct lookup. The first two answers result from the equivalence between "meteorite" and "meteorites", with the second answer appearing also because of the symmetry and right-collapsibility of EQUIV. In the third and fourth answers, the relevant tokens of "iron" and "stones" (onto which the sentence generator appends their modifications in the printout of the answers) are found on the SUBchain of "what", "called" is on the SUBchain of EQUIV, "meteorites" is on the SUBchain of "meteorite", and the inversion is realized by the symmetry of EQUIV.

2. *Text.* The deep-sea fishes are exposed to a danger that comes to no other animal. The danger is that of tumbling upwards.

Analysis.

[((FISHES MOD DEEP-SEA) TMOD THE) (EXPOSED TO ((DANGER (COMES TO ((ANIMAL MOD OTHER) MOD NO))****) TMOD A))****], [(DANGER TMOD THE)[18] EQUIV ((DANGER OF (TUMBLING MOD UPWARDS)) TMOD THE)]

Semantic Information. (DEEP-SEA SUP PLACE), (WHERE EQUIV PLACE), (TUMBLING SUP FALLING), (UPWARDS SUP UP), (FISHES SUP ANIMAL), (EXPOSED SUP CONDITION), (DANGER SUP CIRCUMSTANCE), (ANIMAL SUP THING), (CONDITION SUP THING), (CIRCUMSTANCE SUP THING); ⟨ANIMAL MOD PLACE⟩, ⟨THING LOC PLACE⟩, ⟨THING LOCAT THING⟩, ⟨THING CONDITION ****⟩, ⟨CONDITION TO CIRCUMSTANCE⟩, etc.

Inferential Information. [LOC PROPERTY TRANS], [LOCAT PROPERTY TRANS], [LOCAT PROPERTY SYMM], [[LOCAT C/P LOC] SUP LOC], [[MOD RSTR PLACE] SUP LOC], [[EXPOSED S*MOD TO] SUP LOCAT]

Question. Where is there danger of falling up?

Analysis. [(DANGER OF (FALLING MOD UP)) LOC WHERE]

Answer. Danger of tumbling upwards be in deep-sea.

Explanation. Of the three constituent triples of the question, the first two, (DANGER OF FALLING) and (FALLING MOD UP), are answered by generalized direct lookup. The third (DANGER LOC WHERE), is answered by the inference rule [[LOCAT C/P LOC] SUP LOC], which generates the subgoals (DANGER LOCAT THING) and (THING LOC WHERE). These two subgoals are both answered, in turn, by inference rules. (DANGER LOCAT THING) answers by the inference rule [[EXPOSED S*MOD TO] SUP LOCAT], which generates, as one of its expansions (there are two because of the symmetry of LOCAT), the subgoals (THING EXPOSED ****) and (EXPOSED TO DANGER), both of which are answered by generalized direct lookup. (THING LOC WHERE) answers by the inference rule [[MOD RSTR PLACE] SUP LOC], which generates the subgoals (THING MOD WHERE) and (PLACE SUB WHERE); both of which are answered by generalized direct lookup. The agreement conditions are all satisfied in the recombination of answers to the subgoals, and the answer above is what results. The computer printout in the Appendix shows a trace of a function called STOGOAL, which indicates the path of search in subgoal generation; a significant observation to be made here is that, because of the effectiveness as search-limiting heuristics of the meaningfulness check and the condition that no goal may generate itself as a subgoal at any level (so as to prevent looping in the search process), only one of six attempted expansions that were not part of the correct solution path was actually made.

3. *Text.* There is a monkey. There is a box. There are some bananas. The bananas are higher than the monkey. The bananas are higher than the box. The box is an elevated-object. The monkey move the box to the bananas. The monkey stand on the box. The monkey reach for the bananas.

Analysis.

[(MONKEY TMOD A) LOC THERE], [(BOX TMOD A) LOC THERE],
[(BANANAS TMOD SOME) LOC THERE],
[(BANANAS TMOD THE) HIGHER (MONKEY TMOD THE)],
[(BANANAS TMOD THE) HIGHER (BOX TMOD THE)],
[(BOX TMOD THE) EQUIV (ELEVATED-OBJECT TMOD AN)],
[(MONKEY TMOD THE) (MOVE TO (BANANAS TMOD THE)) (BOX TMOD THE)],
[(MONKEY TMOD THE) (STAND ON (BOX TMOD THE)) ****]

[16] Additional semantic information not given here is required for semantic analysis but not for answering the question.

[17] The barbarities in the English output of this and subsequent answers are due to an oversimplified generation grammar. To eliminate these barbarities, nouns and verbs would have had to have been subdivided into singular and plural categories, the definite article inserted before the subject of each sentence, and transitive verbs subdivided according to whether their objects took a definite or indefinite/zero article. Making these refinements, particularly since our grammars do not deal with features, would have resulted in an undesirable proliferation of the number of rules required for both analysis and generation.

[18] The TMOD THE, in this instance, links to the previous occurrence of DANGER.

[(MONKEY TMOD THE) ((REACH·ACT)[19] FOR (BANANAS TMOD THE)) ****]

Inferential Text. To be lower than is the inverse of to be higher than. To be moved is the inverse of to move. To be moved to is to be at. To be lower than and be at is to be under. To stand on an elevated-object under and reach for is to reach.

Analysis.[20]

[LOWER INVERSE HIGHER], [MOVED INVERSE MOVE],

[[MOVED S*MOD TO] SUP AT], [[LOWER S*AND AT] SUP UNDER],

[[[[[STAND S*MOD ON] RSTR ELEVATED-OBJECT] C/P UNDER] S*AND [(REACH·ACT) S*MOD FOR]] SUP (REACH·GET)]

Semantic Information. (MONKEY SUP ANIMAL), (ANIMAL SUP OBJECT), (BOX SUP OBJECT), (BANANAS SUP OBJECT), (OBJECT SUP CONCRETE), (MOVE SUP CHANGE), (TO SUP DIRECTION), (LOWER SUP LOCREL), (AT SUP LOCREL), (UNDER SUP LOCREL), (ON SUP LOCREL), (LOCREL SUP REL), ((REACH·ACT) SUP ACT), (FOR SUP PURPOSE), (PURPOSE SUP REL), ((REACH·GET) SUP GET); ⟨OBJECT LOCREL OBJECT⟩, ⟨ANIMAL CHANGE CONCRETE⟩, ⟨CHANGE DIRECTION *TOP⟩, ⟨ANIMAL ACT ****⟩, ⟨ACT REL OBJECT⟩, etc.

Question. Does the monkey get the bananas?

Analysis. [(MONKEY TMOD THE) GET (BANANAS TMOD THE)]

Answer. Monkey reach bananas.

Explanation. The monkey gets the bananas if he reaches them, and he can reach them if he stands on an elevated-object under them and reaches for them; the monkey stands on the box, which is an elevated-object, and reaches for the bananas, and so if the box is under the bananas the monkey has succeeded in reaching them; the box is under the bananas if it is lower than they, which it is, and also at the bananas; the box is at the bananas if some animal has moved it to the bananas—which of course the monkey has done; therefore, the monkey reaches the bananas. The question-answerer follows just this reasoning path in answering the question (though in terms, of course, of the formal concept structure); the recombination of subgoal answers into an answer to the question is shown by the trace of STOGOAL in the Appendix. It is interesting to note here, of course, the input of inverses (converses) and inference rules by means of English sentences; with a few additional grammar rules and pieces of lexical information, properties could be input in this manner also.

What should be clear from these examples is that the top-down goal-expansion process, when coupled with generalized direct lookup and limited in its search by SEF checks (which are, in essence, a formal way of positing meaningfulness as a criterion for goal acceptability) and nonrecursion checks, provides a direct and effective method for answering questions by deductive inference—at least in small-scale examples such as these. Since the system is core-bound (there are about 9000 words of free storage, or enough for about ten simple sentences with their associated semantic, syntactic, and inferential information, in the present system) and since the combination of the LISP compiler and the Q-32 time-sharing operating system produces excessively slow running times for a complex program (it normally takes between 20 and 30 minutes of console time to answer a question like the second or third example, which requires a reasonable amount of inference, and even longer to input and semantically analyze the question and its relevant text), there has been no opportunity to test the system on realistically large data bases.[21] It is in operating on such data-bases, one would expect, that the path length heuristic would begin to show its true worth and the limitations of some of the other features, such as generalized direct lookup, would begin to show up.

4.2 LIMITATIONS OF QUESTION-ANSWERING CAPABILITY. In Section 2.3 some of the limitations of the deductive formalism were discussed. Aside from these formal limitations, however, there are system limitations that render it impossible to answer a significant subset of questions that one would like, and even expect, the system to answer. In this section we give several examples of questions the system cannot answer (through whatever amount of inferential information) and discuss the reasons why.

1. *Text.* Lake Titicaca, more than 12,500 feet above sea level, is the highest body of water in the world traversed by steamships.

Analysis.

[(LAKE-TITICACA MOD ((FEET MORE 12500) ABOVE (LEVEL MOD SEA))) EQUIV(((((BODY OF WATER) IN WORLD) TRAVERSED STEAMSHIPS) MOD HIGHEST) TMOD THE)]

Question. Where are steamers found 12,500 feet above sea level?

Analysis. [((STEAMERS FOUND ****) MOD ((FEET MOD 12500) ABOVE (LEVEL MOD SEA))) LOC WHERE]

Additional Information. (STEAMERS EQUIV STEAMSHIPS), (MORE SUP MOD), (TRAVERSED IN-

[19] This is to indicate the sense of "reach" that has "act" as its superclass; the other sense of "reach" used here is (REACH·GET).

[20] The TMOD TO determiner modification, which appears in the printouts in the Appendix, is omitted here for the sake of clarity.

[21] This deficiency is presently being corrected, however, in a new version of Protosynthex III that is being programmed in JOVIAL (by W. Schoene) and that uses disk as an auxiliary storage medium.

VERSE TRAVERSE), (TRAVERSE SUP LOC), (LAKE-TITICACA SUP WHERE), (FEET SUP DISTANCE), [[LOC C/P [MOD RSTR DISTANCE]] SUP MOD]

Explanation. There is no way of deriving the question triple (STEAMERS FOUND ****) from the analyzed text. The element **** would have to appear in the analyzed text and be related to STEAMERS (or something subordinate or equivalent) through a complex relation from which FOUND could be derived from an inference rule. This points up the weakness of the present system in handling intransitive verbs, including passives with agent deletion as in this example—it is really a case of the "deep structure" of our formal representation being not quite deep enough. Otherwise, the question is perfectly well answerable with the additional information given.

2. *Text.* Out come the mature cicadas, to make the air resound for their few weeks of life with their shrill ear-piercing song. The noisemaking apparatus is the most complicated musical organ in nature. It consists of little drumlike plates at the base of the abdomen, which are vibrated rapidly by tireless muscles.

Question. How does a cicada "sing"?

Analysis. [(CICADA TMOD A) (SING METHOD HOW) ****]

Explanation. "How" questions, which require inference from a sequence of actions to a result in such a way that the answer contains in the appropriate relation both the sequence of actions and the result, are unanswerable in the system unless the sequence of actions leading to the result is represented as a single concept and the action-result relation is either explicit in the data or inferrable from explicit relations in the data. To cite another example, the question "How does the monkey get the bananas?" would have been unanswerable from the data in Example 3 of the last section with any set of inferential information. To correct this deficiency and the comparable deficiency with "why" questions (as in the next example) would require an addition to the existing question-answering machinery that would recognize METHOD, REASON, etc., as special right-collapsible relations that would cause the inference-making machinery to include answers to subgoals as an integral part of the answer to the entire question. Such an answer might then be expressed by the sentence generator as something like "Method that monkey reach bananas is that monkey move box to bananas and stand on box and reach for bananas." for the monkey question or "Method that cicada sing is that little drumlike plates at base of abdomen be vibrated rapidly by tireless muscles." for the cicada question (assuming that the text could be analyzed in such a way and that appropriate inferential information could be given so that this answer could be derived).

3. *Text.* The beret was made of a circular piece of cloth gathered onto a band decorated with jewels or embroidery. Inside the band was a string, which could be tightened to fit any head. The tiny bow on the inside of men's hats today is a survival of that string.

Question. Why does a man's hat have a little bow on the inside?

Analysis. [[((HAT POSSBY MAN) TMOD A) HAVE (((BOW MOD LITTLE) TMOD A) ON (INSIDE TMOD THE))] REASON WHY]

Explanation. This question would pose three problems for the system. The first is the presence of "why", which raises the same problem that "how" raises and which, we presume, could be treated in the same manner. The second is equating the structure [HAT HAVE (BOW ON INSIDE)] in the question with its counterpart (BOW ON (INSIDE OF HATS)) in the analysis of the text. Although this could be accomplished through an inference rule like [[LOCREL C/P [OF RSTL PART]] SUP HAVE-converse], it still reflects the need, also seen in the first example, for a "deeper" semantic analysis of the text that would bring the two structures into correspondence. The third problem is the fact that since it must match an existing event triple (there is no provision in the system for creating "temporary" event triples in the data structure for things found by inference), a nested square-bracketed question triple such as [HAT HAVE BOW] in the example can be answered only in the very simplest of ways, by a direct lookup on the SUBchains of its elements for a matching event—thus even the use of the inference rule given above would have been precluded in this example.

5. Discussion

5.1 Technical Aspects. In the foregoing we have described the capabilities and limitations of Protosynthex III for deductive question answering in subsets of natural English on data bases consisting of relational network representations of small samples of English text. We have demonstrated that a question-answering system in which both the deductive formalism and the inference procedures are designed around the semantic organization of the data structure is capable of great power and efficiency in answering English questions by deductive inference. The use of a deductive formalism based on properties of relations and complex combinations of relations by means of relational operators, and the use of a question-answering procedure that works top-down from question to answer at two levels (generalized direct lookup and explicit subgoal generation) and that employs different aspects of the semantic organization to guide its search at each of the two levels are consequences of this "semantic" approach to question answering. We feel that such an approach will in the long run prove more useful for deductive question answering that is based on a meaningful analysis of natural language text and questions than will the more formalistic

approaches to question answering, such as those based on Robinson's [22] resolution principle and similar mechanical proof procedures.[22]

The representations and methods described here would, however, come up against severe limitations of processing efficiency if they were applied to data bases much larger than those we have tested so far—which would include, of course, any practical application of a question-answering system. To be sure, the number of subgoal expansions involved in answering a question depends not at all on the size of the data base but rather on the number of inference rules and SEFs. Moreover, it is likely that the path length heuristic would serve to reduce the dependence of the total number of subgoals generated on the depth of the expansion to a factor approaching linearity whatever the number of expansions generated for each subgoal, so that the average time expended in subgoal generation would increase only linearly with the average number of expansions per subgoal. But the amount of time spent in generalized direct lookup varies as a combinatorial function of the number of data triples containing elements of the SUBchains of the question elements, and the amount of time spent combining the answers to the subgoals of an expansion, checking agreement conditions, and deriving answers to the supergoal from these combinations varies as a combinatorial function of the number of answers to each subgoal. Therefore, in any realistically sized data base the cost of processing subgoals such as (THING LOC WHERE) in the second example of Section 4.1, or (OBJECT MOVED ANIMAL) in the third example of that section, or (WHO LED GROUP) in the example of Section 3.5 would become prohibitively high, for any of these questions would surely access large areas of the data base and produce a large number of answers.

There are two discernible tactics that one could employ in solving this problem. First, one could devise a method of partitioning the data base so that the partitions were each of manageable size and only the most relevant partitions would be searched in answering a question. One way of partitioning the data base would be by *discourse units*—units of, say, paragraph length, each of which could consist of a relatively independent set of interrelated concepts, tokens, and events. Just how such units would be determined, other than by simply using the paragraph boundary as a heuristic indicator of discourse boundary, is a matter for further research. Using the paragraph boundary would surely not work in certain cases—for example, in the many instances in which a word's antecedent is in a preceding paragraph. But given such a partitioning, one could employ methods such as those used by Simmons *et al.* [25] in the Protosynthex I text retrieval system to select an appropriate discourse unit or units for further lookup and inference in answering a question. Such a procedure could be rendered more effective by using the entire SUBchains (excluding tokens) of the concepts in the question as the set of index terms for the initial retrieval, rather than just the paraphrases as in Protosynthex I.

The other tactic one could employ here is to base the structure of data and questions on a unit larger than the relational triple. Such a move would result in a decrease in the number of subgoals generated and the number of answers to each subgoal, thereby reducing the time spent in subgoal generation and answer combination. Generalized direct lookup would, of course, take longer for each subgoal, but this increase ought to be more than compensated for by the decrease in the number of subgoals. This move turns out to have a linguistic motivation as well: Fillmore [8, 9] has proposed as the basic unit of linguistic structure (i.e. as the deep structure representation of a simple sentence) a verb with its various objects (including its subject) linked to it by explicitly marked case relations (a noun and its modifiers would constitute a similar structure). The Fillmore structure also supplies the "deeper" level of semantic analysis that we have seen to be necessary to answer some questions, such as the first and third examples of the last section; and this structure affords the additional flexibility of permitting one to write inference rules expanding n-ary relations for arbitrary n (which are, in fact, what the concepts heading a Fillmore structure represent) rather than only binary relations. Thus the substitution of the Fillmore structure for the structure of event triples would yield substantial benefits for natural-language deductive question answering.

Another change that would be required to transform Protosynthex III into a question-answering system of practical utility would be the introduction of *question-answering operators*—such as count, list, name, and yes-no—and to allow some specification of the number of answers desired for the question (one, five, several, all, etc.). At present only a nested event triple to be matched is presented to the question-answerer; the question-answering procedure then does lookup and subgoal expansion on the pattern's constituent triples until it either infers one or more instances of the triple from the data or can proceed no further, and passes the instances it gets unaltered to the sentence generator. English questions, however, not only specify a pattern of which instances are to be found, but also indicate whether some, all, or a specific number of instances are to be found, which part or parts of the complete structure constitute the information requested, and what operation (straight output, count, sum, average, etc.) is to be done with the set of answers so specified. A formalism representing these quantification, specification, and operation aspects of questions needs to be made part of the entire conceptual formalism and treated appropriately by the semantic analyzer and question-answerer.

[22] One reason for this is that existing logical formalisms are not powerful enough to represent all the meanings of natural English, particularly with respect to noncriterial attributes of concepts—this has been pointed out elsewhere by Schwarcz [22]. Therefore, more general heuristic methods should be more readily extendible to an adequate semantic formalism than algorithmic methods developed for a particular logical formalism that falls short of adequacy.

Finally, there is much linguistic work yet to be done to make natural language question answering succeed on a practical basis. Modality, quantification, time and tense relations, and identification of anaphoric and other discourse relationships are areas of linguistic research that are still on shaky ground, yet are vitally important to question answering. Conversely, no linguistic theory can be considered formally adequate or complete until it can be shown to support a model of deductive question answering that can make all the inferences that intelligent native speakers are generally capable of. The value of linguists and computer scientists working together to solve these problems should be most obvious.

5.2 Summary and Conclusions. The Protosynthex III question-answering system was designed and implemented as a prototype model of verbal understanding and inference by computer. It is based on a data structure composed of event triples consisting of concepts, and instances of concepts, connected by binary relations. The data structure is semantically organized on the basis of superset chains and equivalence classes of concepts, and semantic event forms (SEFs) which, in conjunction with the superset and equivalence relations, delineate the set of semantically acceptable relations between concepts. The deductive-inference properties of the data structure are defined by means of properties on relations and complex combinations of relations formed with relational operators. The system is programmed in Lisp 1.5 on the SDC Q-32 time-sharing system, and is designed to operate interactively with a user at a teletype.

Question answering in the system is performed by a top-down procedure that incorporates generalized direct lookup on elementary subquestions (including all inferences performable solely through relational properties and the converse operator on relations), explicit inference by means of subgoal generation from complex relation-operator combinations, and heuristics that direct search according to meaningfulness, proximity of elements in the data base, and other criteria. The question-answering procedure is given a structure representing a pattern of concepts and relations for which it is to look for instances in the data base. It returns representations of the instances it finds to a sentence generator, where they may be expressed in full or abbreviated form or paraphrased under user control.

Experimentation with the program has been limited by the slowness of the system under time-sharing and by the relatively small amount of core storage available to it. Nevertheless, the system has demonstrated the capability of making a wide range of simple and complex inferences to derive answers to questions in a reasonable time and with relatively little—if any—waste of search effort. The system can also determine in a reasonable time that an answer to a question cannot be found. The system is limited by its inability to represent negation and limited ability to represent quantification, its inability to answer certain questions because of the too syntactically bound nature of its data structure, and its inability to answer all but the simplest "how" and "why" questions. Remedies for all these limitations, however, have been conceived and are presently being implemented in an expanded version of Protosynthex III.

Even the limited success of Protosynthex III has demonstrated that meaningful natural language processing by computer, including "intelligent" question answering, has arrived. To be sure, much further research into the formalization of meaning in natural language, including further exploration of the logic of question-answer relationships, is required. And even if an adequate formalization is found, many man-years of lexicographic and other linguistic effort would be required to encode the meanings of English into the formalization (although through computational aids the difficulty of this task might be significantly reduced) and additional man-years of programming effort to develop and refine the algorithms to the point where they are practically useful. But once it has been recognized that the task is feasible, the huge and mushrooming information-handling requirements of today's technological society and the increasingly felt need for personalization of the relations between men and the systems with which they interact make it imperative that the task be done.

Appendix

This appendix contains the actual teletype inputs and outputs for the examples of successful question answering presented in Section 4.1. In all three examples, the superclass and word-class information for the appropriate senses of the words that appear in the text was input beforehand. In Example 1, the SEFs were input beforehand also; in Examples 2 and 3, they are input immediately following the sentences in whose interpretation they are first used.

After each sentence or question is input, its analysis or analyses are printed out in the form of nested triples. Here each pair (word . superclass) denotes a word sense, and the numbers of spaces of indentation of each line from the left margin indicates the level of nesting at that point. Following the analysis of each declarative sentence, the actual data structure that is stored to represent that sentence is printed. The G numbers at the left of each line are the symbolic names of event triples; the value of the property DEPN—I or D—represents square-bracketing or round-bracketing, respectively; the values of the properties U*X and U*Y, where these occur, are the names of other event

(Text continues on page 182.)

Example 1

```
       ANALYSIS MØDE //

 READY--
THE STØNES AND IRØN THAT FALL TØ EARTH FRØM ØUTER SPACE ARE CALLED
METEØRITES.

 1
P
 [[[[(STØNES . ØBJECT)
     (((FALL . MØVE) (TØ . DIRECTIØN) (EARTH . PLACE))
      (FRØM . DIRECTIØN)
```

```
    ((SPACE . PLACE) (MØD . PRIMIT) (ØUTER . RELLØC)))
    (**** . PRIMIT))
   (TMØD . PRIMIT)
   (THE . QUANT))
  (CALLED . EQUIV)
  (METEØRITES . ØBJECT))]
 (AND . BØØLEAN)
 ((((IRØN . ØBJECT)
    (((FALL . MØVE) (TØ . DIRECTIØN) (EARTH . PLACE))
     (FRØM . DIRECTIØN)
     ((SPACE . PLACE) (MØD . PRIMIT) (ØUTER . RELLØC)))
    (**** . PRIMIT))
   (TMØD . PRIMIT)
   (THE . QUANT))
  (CALLED . EQUIV)
  (METEØRITES . ØBJECT))]
1
ØK DØNE
((G09250 (DEPN I MEANS (G09245 C09173 G09249)))
  (G09249 (U*Y (G09250) DEPN I MEANS (T09246 C09123 C09118)))
  (G09248 (DEPN D MEANS (T09246 T09238 ****)))
  (G09247 (DEPN I MEANS (T09246 TKØF C09168)))
  (G09245 (U*X (G09250) DEPN I MEANS (T09242 C09123 C09118)))
  (G09244 (DEPN D MEANS (T09242 T09238 ****)))
  (G09243 (DEPN I MEANS (T09242 TKØF C09178)))
  (G09241 (DEPN D MEANS (T09238 C09143 T09235)))
  (G09240 (DEPN D MEANS (T09238 C09153 C09148)))
  (G09239 (DEPN I MEANS (T09238 TKØF C09158)))
  (G09237 (DEPN D MEANS (T09235 MØD C09138)))
  (G09236 (DEPN I MEANS (T09235 TKØF C09183))))

1
READY--
WHAT IS A METEØRITE?
1
P
 (((METEØRITE . ØBJECT) (TMØD . PRIMIT) (A . QUANT))
  (EQUIV . PRIMIT)
  (WHAT . *TØP))
 1
ØK
  4 ANSWERS.
  ANS 1.
S
  METEØRITE BE METEØRITES.
N S
  ANS 2.
  METEØRITES BE METEØRITE.
N S
  ANS 3.
  IRØN THAT FALL TØ EARTH FRØM ØUTER SPACE BE CALLED METEØRITES.
N S
  ANS 4.
  STØNES THAT FALL TØ EARTH FRØM ØUTER SPACE BE CALLED METEØRITES.
```

Example 2

```
       ANALYSIS MØDE //

READY--
THE DEEP-SEA FISHES ARE EXPØSED TØ A DANGER THAT CØMES TØ NØ ØTHER
ANIMAL.
.
ANIMAL MØD PLACE, THING CØNDITIØN ****, CØNDITIØN TØ CIRCUMSTANCE,
CIRCUMSTANCE HAPPEN ****, HAPPEN TØ THING, THING MØD QUANT.
  2 INTERPRETATIØNS.
1
P
 ((((FISHES . ANIMAL) (MØD . PRIMIT) (DEEP-SEA . PLACE))
   (TMØD . PRIMIT)
   (THE . QUANT))
  ((EXPØSED . CØNDITIØN)
   (TØ . REL)
   (((DANGER . CIRCUMSTANCE)
     ((CØMES . HAPPEN)
      (TØ . REL)
      (((ANIMAL . THING) (MØD . PRIMIT) (ØTHER . QUANT))
       (MØD . PRIMIT)
       (NØ . QUANT)))
     (**** . PRIMIT))
    (TMØD . PRIMIT)
    (A . QUANT)))
  (**** . PRIMIT))]
 1
ØK DØNE
 ((G09683 (DEPN I MEANS (T09667 T09680 ****)))
   (G09682 (DEPN D MEANS (T09680 C09464 T09677)))
   (G09681 (DEPN I MEANS (T09680 TKØF C09489)))
   (G09679 (DEPN D MEANS (T09677 T09674 ****)))
   (G09678 (DEPN I MEANS (T09677 TKØF C09479)))
   (G09676 (DEPN D MEANS (T09674 C09464 T09670)))
   (G09675 (DEPN I MEANS (T09674 TKØF C09469)))
   (G09673 (DEPN D MEANS (T09670 MØD C09459)))
   (G09672 (DEPN D MEANS (T09670 MØD C09454)))
   (G09671 (DEPN I MEANS (T09670 TKØF C09424)))
   (G09669 (DEPN D MEANS (T09667 MØD C09504)))
   (G09668 (DEPN I MEANS (T09667 TKØF C09499))))

 1

 READY--
THE DANGER IS THAT ØF TUMBLING UPWARDS.
.
CIRCUMSTANCE ASSØC ACTIØN, ACTIØN MØD DIRECTIØN.
 1
P ØK DØNE
 (((DANGER . CIRCUMSTANCE) (TMØD . PRIMIT) (THE . QUANT))
  (EQUIV . PRIMIT)
  (((DANGER . CIRCUMSTANCE)
    (ØF . ASSØC)
    ((TUMBLING . FALLING) (MØD . PRIMIT) (UPWARDS . UP)))
   (TMØD . PRIMIT)
   (THE . QUANT))]
 1
 ((G09692 (DEPN I MEANS (T09677 EQUIV T09689)))
   (G09691 (DEPN D MEANS (T09689 C09562 T09686)))
   (G09690 (DEPN I MEANS (T09689 TKØF C09479)))
   (G09688 (DEPN D MEANS (T09686 MØD C09552)))
   (G09687 (DEPN I MEANS (T09686 TKØF C09557))))
 1

 READY--
WHERE IS THERE DANGER ØF FALLING UP?
.
THING LØC PLACE, THING LØCAT THING.
 1
P
 (((DANGER . CIRCUMSTANCE)
    (ØF . ASSØC)
    ((FALLING . ACTIØN) (MØD . PRIMIT) (UP . DIRECTIØN)))
  (LØC . PRIMIT)
  (WHERE . SPATIAL))]
 1
E UTILITY()

       UTILITY MØDE //

 RULE INFER [LØC PRØPERTY TRANS] IN
  DØNE

 RULE INFER [LØCAT PRØPERTY TRANS] IN
  DØNE

 RULE INFER [LØCAT PRØPERTY SYMM] IN
  DØNE

 RULE INFER [[LØCAT C/P LØC] SUP LØC] IN
  DØNE

 RULE INFER [[MØD RSTR PLACE] SUP LØC] IN
  DØNE

 RULE INFER [[EXPØSED S*MØD TØ] SUP LØCAT] IN
  DØNE

 RUN
  T
  1
 E TRACE((ST0G0AL))
  (ST0G0AL)
  1
 ØK
  ARGS ØF ST0G0AL
  SG09702
  ((SG09703 SG09706)
    (SG09709)
    (((SG09706 3) SG09703 1) ((SG09709 1) SG09706 1))
    (((SG09706 1) (SG09706 2) ((SG09703 1) (SG09703 2) (SG09703 3)))
     (SG09709 2) (SG09709 3)))

  VALUE ØF ST0G0AL
  2

  ARGS ØF ST0G0AL
  SG09709
  (NIL (SG09712 SG09717)
    (((SG09712 3) SG09717 1)) ((SG09712 1) (LØC) (SG09717 3)))

  VALUE ØF ST0G0AL
  2

  ARGS ØF ST0G0AL
  SG09702
  ((SG09703 SG09706)
    (SG09709)
    (((SG09706 3) SG09703 1) ((SG09709 1) SG09706 1))
    (((SG09706 1) (SG09706 2) ((SG09703 1) (SG09703 2) (SG09703 3)))
     (SG09709 2) (SG09709 3)))

  VALUE ØF ST0G0AL
  2

  ARGS ØF ST0G0AL
  SG09712
  ((SG09720)
    (SG09719)
    (((SG09719 2) SG09720 1)) ((SG09719 1) (C09655) (SG09720 3)))

  VALUE ØF ST0G0AL
  3

  ARGS ØF ST0G0AL
  SG09712
  ((SG09721 SG09722)
    NIL (((SG09721 2) SG09722 1)) ((SG09722 3) (C09655) (SG09721 1)))

  ARGS ØF ST0G0AL
  SG09709
  ((SG09712)
    (SG09717) (((SG09712 3) SG09717 1)) ((SG09712 1) (LØC) (SG09717 3)))

  ARGS ØF ST0G0AL
  SG09702
  ((SG09703 SG09706)
    (SG09709)
    (((SG09706 3) SG09703 1) ((SG09709 1) SG09706 1))
    (((SG09706 1) (SG09706 2) ((SG09703 1) (SG09703 2) (SG09703 3)))
     (SG09709 2) (SG09709 3)))
```

```
VALUE ØF STØGØAL
1

VALUE ØF STØGØAL
1

VALUE ØF STØGØAL
0

ARGS ØF STØGØAL
SG09717
((SG09724 SG09725)
  NIL (((SG09724 3) SG09725 3)) ((SG09724 1) (LØC) (SG09725 3)))

ARGS ØF STØGØAL
SG09709
((SG09717 SG09712)
  NIL (((SG09712 3) SG09717 1)) ((SG09712 1) (LØC) (SG09717 3)))

ARGS ØF STØGØAL
SG09702
((SG09709 SG09703 SG09706)
  NIL (((SG09706 3) SG09703 1) ((SG09709 1) SG09706 1))
  (((SG09706 1) (SG09706 2) ((SG09703 1) (SG09703 2) (SG09703 3)))
   (SG09709 2) (SG09709 3)))

VALUE ØF STØGØAL
0

VALUE ØF STØGØAL
0

VALUE ØF STØGØAL
0

 1 ANSWERS
 ANS 1.
S
 DANGER ØF TUMBLING UPWARDS BE IN DEEP-SEA.
```

Example 3

```
       ANALYSIS MØDE //

 READY--
THERE IS A MØNKEY.  . CØNCRETE LØC PLACE.
 1
P
 [((MØNKEY . ANIMAL) (TMØD . PRIMIT) (A . QUANT))
  (LØC . PRIMIT)
  (THERE . PLACE)]
 1
ØK
 ((G09329 (DEPN I MEANS (T09327 LØC C09120)))
   (G09328 (DEPN I MEANS (T09327 TKØF C09105))))
 1
DØNE

 READY--
THERE IS A BØX.
 1
P
 [((BØX . ØBJECT) (TMØD . PRIMIT) (A . QUANT))
  (LØC . PRIMIT)
  (THERE . PLACE)]
 1
ØK
 ((G09332 (DEPN I MEANS (T09330 LØC C09120)))
   (G09331 (DEPN I MEANS (T09330 TKØF C09129))))
 1
DØNE

 READY--
THERE ARE SØME BANANAS.
 1
P
 [((BANANAS . ØBJECT) (TMØD . PRIMIT) (SØME . QUANT))
  (LØC . PRIMIT)
  (THERE . PLACE)]
 1
ØK
 ((G09335 (DEPN I MEANS (T09333 LØC C09120)))
   (G09334 (DEPN I MEANS (T09333 TKØF C09135))))
 1
DØNE

 READY--
THE BANANAS ARE HIGHER THAN THE MØNKEY.  . ØBJECT LØCREL ØBJECT.
 1
P
 [((BANANAS . ØBJECT) (TMØD . PRIMIT) (THE . QUANT))
  (HIGHER . LØCREL)
  ((MØNKEY . ANIMAL) (TMØD . PRIMIT) (THE . QUANT))]
 1
ØK
 ((G09337 (DEPN I MEANS (T09333 C09173 T09327))))
 1
DØNE

 READY--
THE BANANAS ARE HIGHER THAN THE BØX.
 1
P
 [((BANANAS . ØBJECT) (TMØD . PRIMIT) (THE . QUANT))
  (HIGHER . LØCREL)
  ((BØX . ØBJECT) (TMØD . PRIMIT) (THE . QUANT))]
 1
ØK
 ((G09338 (DEPN I MEANS (T09333 C09173 T09330))))
 1
DØNE

 READY--
THE BØX IS AN ELEVATED-ØBJECT.
 1
P
 [((BØX . ØBJECT) (TMØD . PRIMIT) (THE . QUANT))
  (EQUIV . PRIMIT)
  ((ELEVATED-ØBJECT . ØBJECT) (TMØD . PRIMIT) (AN . QUANT))]
 1
ØK
 ((G09341 (DEPN I MEANS (T09330 EQUIV T09339)))
   (G09340 (DEPN I MEANS (T09339 TKØF C09181))))
 1
DØNE

 READY--
THE MØNKEY MØVE THE BØX TØ THE BANANAS.  . ANIMAL CHANGE CØNCRETE,
 CHANGE DIRECTIØN *TØP.
 1
P
 [((MØNKEY . ANIMAL) (TMØD . PRIMIT) (THE . QUANT))
  ((MØVE . CHANGE)
   (TO . DIRECTIØN)
   ((BANANAS . ØBJECT) (TMØD . PRIMIT) (THE . QUANT)))
  ((BØX . ØBJECT) (TMØD . PRIMIT) (THE . QUANT))]
 1
ØK
 ((G09347 (DEPN I MEANS (T09327 T09344 T09330)))
   (G09346 (DEPN D MEANS (T09344 C09203 T09333)))
   (G09345 (DEPN I MEANS (T09344 TKØF C09213))))
 1
DØNE

 READY--
THE MØNKEY STAND ØN THE BØX.  . ANIMAL ACT ****, ACT REL ØBJECT.
 1
P
 [((MØNKEY . ANIMAL) (TMØD . PRIMIT) (THE . QUANT))
  ((STAND . ACT)
   (ØN . LØCREL)
   ((BØX . ØBJECT) (TMØD . PRIMIT) (THE . QUANT)))
  (**** . PRIMIT)]
 1
ØK
 ((G09353 (DEPN I MEANS (T09327 T09350 ****)))
   (G09352 (DEPN D MEANS (T09350 C09226 T09330)))
   (G09351 (DEPN I MEANS (T09350 TKØF C09231))))
 1
DØNE

 READY--
THE MØNKEY REACH FØR THE BANANAS.
 1
P
 [((MØNKEY . ANIMAL) (TMØD . PRIMIT) (THE . QUANT))
  ((REACH . ACT)
   (FØR . PURPØSE)
   ((BANANAS . ØBJECT) (TMØD . PRIMIT) (THE . QUANT)))
  (**** . PRIMIT)]
 1
ØK
 ((G09357 (DEPN I MEANS (T09327 T09354 ****)))
   (G09356 (DEPN D MEANS (T09354 C09243 T09333)))
   (G09355 (DEPN I MEANS (T09354 TKØF C09248))))
 1
DØNE

 READY--
TØ BE LØWER THAN IS THE INVERSE ØF TØ BE HIGHER THAN.
 1
P
 [((LØWER . LØCREL) (TMØD . PRIMIT) (TØ . QUANT))
  (INVERSE . PRIMIT)
  ((HIGHER . LØCREL) (TMØD . PRIMIT) (TØ . QUANT))]
 1
ØK
 NIL
 1
DØNE

 READY--
TØ BE MØVED IS THE INVERSE ØF TØ MØVE.
 1
P
 [((MØVED . CHANGE) (TMØD . PRIMIT) (TØ . QUANT))
  (INVERSE . PRIMIT)
  ((MØVE . CHANGE) (TMØD . PRIMIT) (TØ . QUANT))]
 1
ØK
 NIL
 1
DØNE

 READY--
TØ BE MØVED TØ IS TØ BE AT.
 1
P
 [[[(MØVED . CHANGE) (S*MØD . PRIMIT) (TØ . DIRECTIØN)]
   (TMØD . PRIMIT)
   (TØ . QUANT))
  (SUP . PRIMIT)
  ((AT . LØCREL) (TMØD . PRIMIT) (TØ . QUANT))]
 1
ØK
 ((G09359 (DEPN I MEANS (G09358 SUP C09295)))
   (G09358 (U*X (G09359) DEPN I MEANS (C09279 S*MØD C09203))))
```

```
 1
DØNE

 READY--
TØ BE LØWER THAN AND BE AT IS TØ BE UNDER.
 1
P
 [[[[LØWER . LØCREL) (S*AND . PRIMIT) (AT . LØCREL)]
   (TMØD . PRIMIT)
   (TØ . QUANT))
  (SUP . PRIMIT)
  ((UNDER . LØCREL) (TMØD . PRIMIT) (TØ . QUANT))]
 1
ØK
 ((G09361 (DEPN I MEANS (G09360 SUP C09290)))
   (G09360 (U*X (G09361) DEPN I MEANS (C09265 S*AND C09295))))
 1
DØNE

 READY--
TØ STAND ØN AN ELEVATED-ØBJECT UNDER AND REACH FØR IS TØ REACH.
  8 INTERPRETATIØNS.
 1
P
 [[[[[[[STAND . ACT) (S*MØD . PRIMIT) (ØN . LØCREL)]
       (RSTR . PRIMIT)
       (ELEVATED-ØBJECT . ØBJECT)]
      (C+P . PRIMIT)
      (UNDER . LØCREL)]
     (S*AND . PRIMIT)
     [(REACH . GET) (S*MØD . PRIMIT) (FØR . PURPØSE)]]
    (TMØD . PRIMIT)
    (TØ . QUANT))
   (SUP . PRIMIT)
   ((REACH . GET) (TMØD . PRIMIT) (TØ . QUANT))]
 1
N P
 2
 [[[[[[[STAND . ACT) (S*MØD . PRIMIT) (ØN . LØCREL)]
       (RSTR . PRIMIT)
       (ELEVATED-ØBJECT . ØBJECT)]
      (C/P . PRIMIT)
      (UNDER . LØCREL)]
     (S*AND . PRIMIT)
     [(REACH . ACT) (S*MØD . PRIMIT) (FØR . PURPØSE)]]
    (TMØD . PRIMIT)
    (TØ . QUANT))
   (SUP . PRIMIT)
   ((REACH . GET) (TMØD . PRIMIT) (TØ . QUANT))]
 2
ØK
 ((G09367 (DEPN I MEANS (G09366 SUP C09313)))
   (G09366 (U*X (G09367) DEPN I MEANS (G09364 S*AND G09365)))
   (G09365 (U*Y (G09366) DEPN I MEANS (T09354 S*MØD C09243)))
   (G09364 (U*X (G09366) DEPN I MEANS (G09363 C/P C09290)))
   (G09363 (U*X (G09364) DEPN I MEANS (G09362 RSTR T09339)))
   (G09362 (U*X (G09363) DEPN I MEANS (T09350 S*MØD C09226))))
 2
DØNE

 READY--
DØES THE MØNKEY GET THE BANANAS?
 1
P
 [((MØNKEY . ANIMAL) (TMØD . PRIMIT) (THE . QUANT))
  (GET . CHANGE)
  ((BANANAS . ØBJECT) (TMØD . PRIMIT) (THE . QUANT))]
 1
E TRACE((STØG0AL))
 (ST0G0AL)
 1
ØK
 ARGS ØF ST0G0AL
 SG09428
 ((SG09429 SG09430)
   NIL (((SG09429 2) SG09430 1)) (((SG09429 1) (C09295) (SG09430 3)))

VALUE ØF ST0G0AL
0

ARGS ØF ST0G0AL
SG09426
((SG09427 SG09428)
  NIL (((SG09427 1) SG09428 1) ((SG09427 3) SG09428 3))
  (((SG09427 1) (C09290) (SG09428 3)))

VALUE ØF ST0G0AL
0

ARGS ØF ST0G0AL
SG09418
((SG09423 SG09424 SG09425 SG09426 SG09431 SG09432)
  NIL (((SG09423 2) SG09424 1)
   ((SG09424 3) SG09425 3)
   ((SG09425 3) SG09426 1)
   ((SG09423 1) SG09431 1)
   ((SG09426 3) SG09432 3) ((SG09431 2) SG09432 1))
  (((SG09423 1) (C09313) (SG09432 3)))

VALUE ØF ST0G0AL
0

ARGS ØF ST0G0AL
SG09417
((SG09418) NIL NIL ((SG09418 1) (SG09418 2) (SG09418 3)))

VALUE ØF ST0G0AL
0

 1 ANSWERS.
 ANS 1.
A
 MØNKEY REACH BANANAS.
```

TABLE 1. QUESTION-ANSWERING GOALS IN EXAMPLE 2

GOAL	IS	FOUND
SG09702	((DANGER OF (FALLING MOD UP)) LOC WHERE)	((T09689 OF (T09686 MOD UP)) LOC DEEP-SEA)
SG09703	(FALLING MOD UP)	(T09686 MOD UP)
SG09706	(DANGER OF FALLING)	(T09689 OF T09686)
SG09709	(DANGER LOC WHERE)	(T09677 LOC DEEP-SEA)
SG09712	(DANGER LOCAT THING)	(T09677 LOCAT T09667)
SG09717	(THING LOC WHERE)	(T09667 LOC DEEP-SEA)
SG09719	(DANGER EXPOSED ****)	None
SG09720	(EXPOSED TO THING)	(T09680 TO T09677)
SG09721	(THING EXPOSED ****)	(T09667 T09680 ****)
SG09722	(EXPOSED TO DANGER)	(T09680 TO T09677)
SG09724	(THING MOD WHERE)	(T09667 MOD DEEP-SEA)
SG09725	(PLACE SUB WHERE)	(PLACE SUB DEEP-SEA)

triples in which the event triple is used as X or Y respectively; the value of the property MEANS is the event triple itself.

In the questions of Examples 2 and 3, the function STOGOAL is traced to show the path of subgoal generation and recombination. The functional arguments of STOGOAL are (1) the internal name of a goal, and (2) the representation of a subgoal expansion of that goal, which is of the form ((⟨answered subgoals⟩) (⟨unanswered subgoals⟩) (⟨agreement conditions⟩) ⟨answer form⟩), as in Figures 2 and 3 of Section 3.3. The value of STOGOAL is the path length of the expansion that is input to it. Tables 1 and 2 below give the meanings (IS) and answers (FOUND) for each of the goals in Examples 2 and 3 (with lexical words used, as in the text, to represent concepts) so that the reader may follow the actual inference paths with the aid of the explanation in Section 4.1.

The meanings of the various user commands (P, N, OK, S, A, and DONE) and other details of the teletype record are fully explained in the Users' Guide and Program Description for Protosynthex III [2]; we therefore give only a brief account of them here. The P command is for printing out, in nested bracketed form, a semantic interpretation of a sentence or question. The N command is to select the next (in cyclical order) of a set of semantic interpretations or answers to a question for further processing. The OK command is to store a semantic interpretation of a sentence in the data structure or to attempt to answer a

TABLE 2. QUESTION-ANSWERING GOALS IN EXAMPLE 3

GOAL	IS	FOUND
SG09418	(MONKEY GET BANANAS)	(T09327 REACH T09333)
SG09423	(MONKEY STAND ****)	(T09327 T09350 ****)
SG09424	(STAND ON OBJECT)	(T09350 ON T09330)
SG09425	(ELEVATED-OBJECT SUB OBJECT)	(ELEVATED-OBJECT SUB ELEVATED-OBJECT), (T09330 SUB T09330), (T09339 SUB T09339)
SG09426	(OBJECT UNDER BANANAS)	(T09330 UNDER T09333)
SG09427	(OBJECT LOWER BANANAS)	(T09330 LOWER T09333)
SG09428	(OBJECT AT BANANAS)	(T09330 AT T09333)
SG09429	(OBJECT MOVED ANIMAL)	(T09330 *T09344 T09327)
SG09430	(MOVED TO BANANAS)	(T09344 TO T09333)
SG09431	(MONKEY REACH ****)	(T09327 T09354 ****)
SG09432	(REACH FOR BANANAS)	(T09354 FOR T09333)

semantic interpretation of a question. The S and A commands are to express a semantic interpretation or answer as a sentence, in full (with tokens being replaced by their full description) or abbreviated (with tokens being replaced by their head concept only) form, respectively. And the DONE command indicates that processing of the set of semantic interpretations or answers has been completed.

RECEIVED MARCH, 1969

REFERENCES

1. BOBROW, DANIEL G. A question-answering system for high school algebra word problems. Proc. AFIPS 1964 Fall Joint Comput. Conf. Vol. 26, Pt. 1, Spartan Books, New York, pp. 591–614.
2. BURGER, J. F., SCHWARCZ, R. M., AND SIMMONS, R. F. Users' guide and program description for Protosynthex III. TM-4068, System Development Corp., Santa Monica, Calif., Sept. 1968.
3. CHOMSKY, NOAM. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, Mass., 1965.
4. COLES, L. STEPHEN. Syntax directed interpretation of natural language. Ph.D. diss., Carnegie-Mellon U., Pittsburgh, Pa., 1967.
5. COLES, L. S. Talking with a robot in English. Proc. International Joint Conf. on Artificial Intelligence, Washington, D. C., May 7-9, 1969.
6. DARLINGTON, JARED L. Machine methods for proving logical arguments expressed in English. *Mech. Trans. Comput. Linguist. 8* (July 1965), 41–67.
7. ELLIOTT, ROGER W. A model for a fact retrieval system. Ph.D. diss., U. of Texas, Austin, Texas, May 1965.
8. FILLMORE, CHARLES J. A proposal concerning English prepositions. In *Report of the 17th Annual Round Table Meeting on Linguistics and Language Studies*, F. P. Dineen (Ed.). Georgetown U. Press, Washington, D. C., 1966, pp. 19–34.
9. FILLMORE, CHARLES J. The case for case. In *Universals in Linguistic Theory*, E. Bach and R. Harms (Eds.) Holt, Rinehart & Winston, New York, 1968, pp. 1–88.
10. GREEN, C. CORDELL, AND RAPHAEL, BERTRAM. Research on intelligent question answering system. Scientific Rep. 1, Stanford Research Institute, Stanford, Calif., May 1967.
11. KATZ, JERROLD J. Recent issues in semantic theory. *Foundations of Language 3* (1967), 124–194.
12. KATZ, JERROLD J., AND POSTAL, PAUL M. *An Integrated Theory of Linguistic Descriptions*. MIT Press, Cambridge, Mass., 1964.
13. KLEIN, SHELDON, LIEMAN, STEPHEN, AND LINDSTROM, GARY. DISEMINER: a distributional-semantics inference maker. CIP Rep., Carnegie-Melloen U., Pittsburgh, Pa., June 1966.
14. LEVIEN, ROGER, AND MARON, M. E. Relational data file: a tool for mechanized inference execution and data retrieval. RM-4793-PR, RAND Corp., Santa Monica, Calif., Dec. 1965.
15. MCCARTHY, JOHN. Situations, actions, and causal laws. Stanford Artificial Intelligence Project Memo No. 2, Stanford U., Stanford, Calif., July 1963.
16. NEWELL, ALLEN, SHAW, J. C., AND SIMON, H. A. Empirical explorations with the logic theory machine: a case study in heuristics. In *Computers and Thought*, E. Feigenbaum and J. Feldman (Eds.). McGraw-Hill, New York, 1963, 109–133.
17. NEWELL, ALLEN, AND SIMON, H. A. GPS, a program that simulates human thought. In *Computers and Thought*, E. Feigenbaum and J. Feldman (Eds.). McGraw-Hill, New York, 1963, pp. 279–293.
18. OLNEY, JOHN C., AND LONDE, DAVID L. An analysis of English discourse structure, with particular attention to anaphoric relationships. SP-2769, System Development Corp., Santa Monica, Calif., Nov. 1967.
19. RAPHAEL, BERTRAM. A computer program which "understands." Proc. AFIPS 1964 Fall Joint Comput. Conf., Vol. 26, Pt. 1, Spartan Books, New York, pp. 577–589.
20. ROBINSON, J. A. A machine-oriented logic based on the resolution principle. *J. ACM 12*, 1 (Jan. 1965), 23–41.
21. ROSENBAUM, PETER S. A grammar base question answering procedure. *Comm. ACM 10*, 10 (Oct. 1967), 630–635.
22. SCHWARCZ, ROBERT M. Towards a computational formalization of natural language semantics. Proc. Int. Conf. on Computational Linguistics, Stockholm, Sept. 1–4, 1969.
23. SIMMONS, R. F., BURGER, J. F., AND SCHWARCZ, R. M. A computational model of verbal understanding. Proc. AFIPS 1968 Fall Joint Comput. Conf., Vol. 33, Spartan Books, New York, pp. 441–456. (Also System Development Corp. doc. SP-3132).
24. SIMMONS, R. F., KLEIN, S., AND MCCONLOGUE, K. L. Indexing and dependency logic for answering English questions. *Amer. Documentation 15* (1964), 196–204.
25. SLAGLE, JAMES R., AND BURSKY, PHILIP. Experiments with a multipurpose theorem-proving heuristic program. *J. ACM 15*, 1 (Jan. 1968), 85–99.
26. TEITELMAN, WARREN. PILOT: A step toward man-computer symbiosis. MAC-TR-32 (Ph.D. th.), Project MAC, MIT, Cambridge, Mass., Sept. 1966.
27. THOMPSON, FREDERICK B., et al. Information systems research: the Deacon project. 65TMP-69, General Electric Co., Oct. 1965.