A Question-answering System Based on the Kay Parsing Algorithm

John Seely Brown

TECHNICAL REPORT #28 - June 1973

This paper is a revised version of a working paper written in February, 1972, titled "Old Fashioned Question Answering Revisited"

A Question-answering System
Based on the Kay Parsing Algorithm

Introduction

In this paper we discuss a question answering system whose deductive component is based solely on the Kay Parsing Algorithm [1,2]. Our system accepts questions phrased in a limited subset of English. Each question is parsed and then transformed into a series of commands which retrieve from the data base potentially relevant data. This data is then sent off to the inference mechanism. In our system, the inferencer is the exact same procedure that parses the natural language input query. That is to say, the "parser" is being used both as an analyser for producing structural descriptions of the input query and as a theorem prover for deducing implicit information in the data base. Its grammar consists of two sub-grammars which are indistinguishable in form if not in content. The first is a context free grammar (with features) for characterizing our subset of English. The second is a context free representation of our rules of inference which characterize our particular domain of "knowledge".

We had two purposes in doing this research. First, we

wanted to explore some of the obvious parallels between parsing and theorem proving and likewise between grammars and inference rules. Second, we wanted to investigate if the Kay Parsing Algorithm could be useful for inference making. We were intrigued by the way this algorithm so efficiently encoded partial parses. Could some of these same techniques be used to circumvent the large amount of back-up inherent in many approaches to inference making by efficiently carrying along parallel "sub-proofs" even when such sub-proofs might never yield a complete proof? (Examples of related capabilities in both top-down and bottom-up parsers are also exemplified in Woods' well-formed substring feature [3] and in Kaplan's GSP [4].)

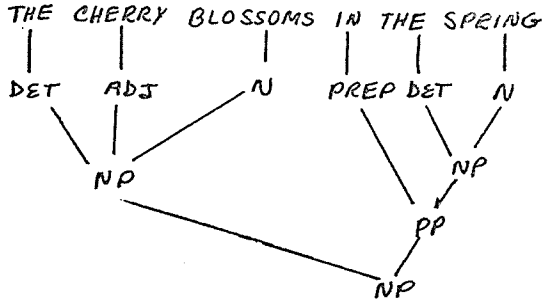
Before delving any deeper into our system, a short description of the Kay Algorithm is in order. We will refer interchangeably to the Kay Algorithm and the Chart Parser wherein the modelling structure underlying Kay's algorithm will be called a "chart". We will describe only the simplest context free aspect of the Parsing Algorithm and refer the interested reader to Kay [2] and Kaplan [4].

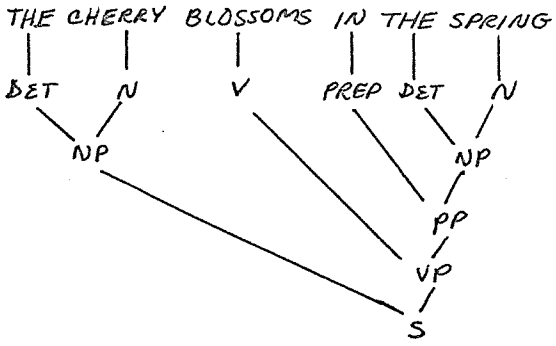
Let us consider the following utterance:

"The cherry blossoms in the Spring

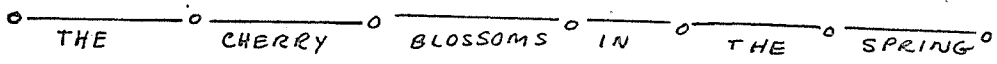
which is syntactically ambiguous yielding two obvious

readings:

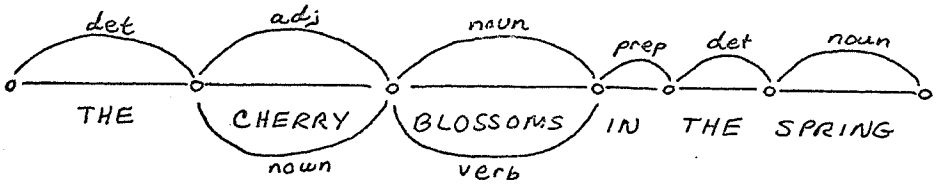




Superimposing these two parses one notes that some parts of the structural description are the same and other parts are different. The Kay parser provides a mechanism for circumventing the need to reparse any well-formed substring that is shared between any two parses. This feat is primarily accomplished by using a special modelling structure to represent the parsings. For the above sentence an initial chart is constructed as follows:



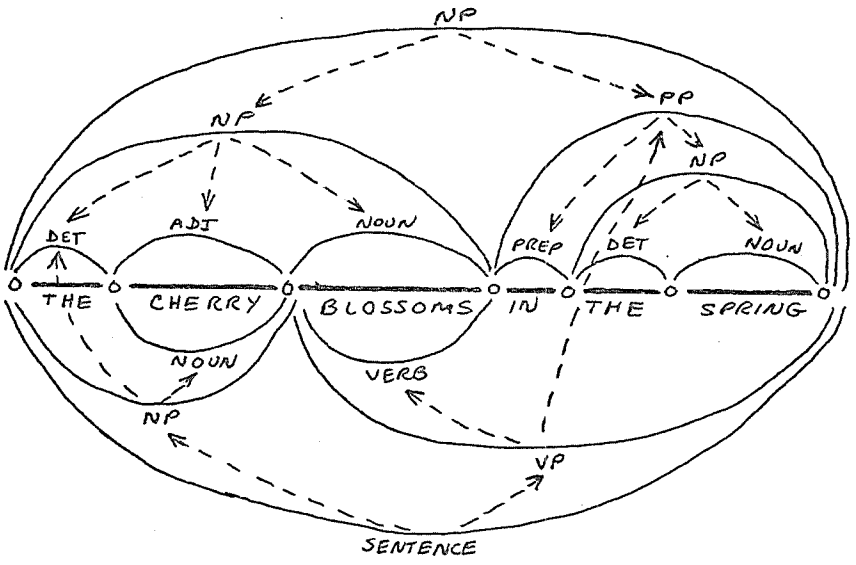
where the words of the utterance act as labels on the arcs. Arcs are then added covering these initial arcs according to what syntactic categories each word may be, i.e.



Assuming for the moment that our grammar is context free, then a single sweep from right to left is made applying all the reduction rules wherever possible.* For each reduction a new arc is added which dominates the sub-arcs that it reduces. (A broken line denotes explicitely which sub-arcs a new arc dominates.) The final chart resulting from the application of all possible rules for this utterance would look like:

---

*We caution the reader that this is a most superficial indication of the parsing scheme and recommend Kay [1] [2] for a thorough description of the chart parser as it applies to transformational grammar.



An inspection of the resulting chart reveals that exactly those phrases in common to both parses occur just once. In fact, not only do the phrases in common occur uniquely in the final chart but they are "considered" only once as we applied the Kay Algorithm from right to left.

Question-Answerer

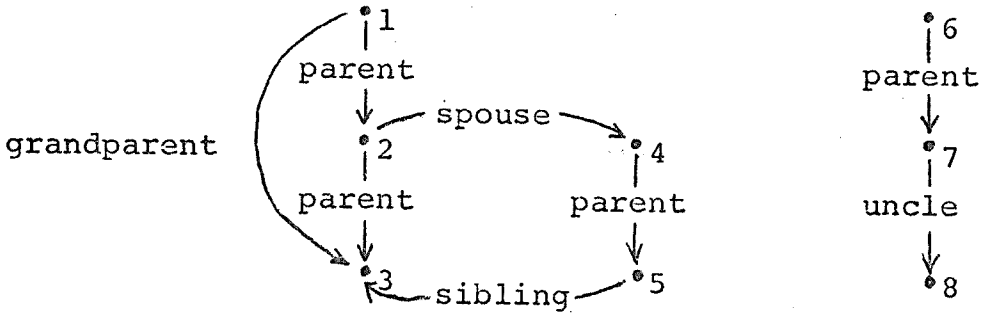
The domain of discourse chosen to exemplify our system is that classical area pertaining to kinship. Although we don't want to become sidetracked in defending the non-triviality of this domain, we suggest to the skeptical reader the seemingly innocent exercise of axiomatizing kinship structure in the first order predicate logic and

then using this formalization with a resolution theorem prover. A similar exercise in Planner will neatly show the phenominal back-up inherent in even this simple domain. We recognize that our system is in no sense a general theorem prover. Nevertheless it performs its class of inferences extremely efficiently and provides some interesting analogies.

Overview

The basic theory behind our system is relatively straightforward. Suppose our data base consists of a collection of facts of the form (x R y) where x and y are people and R is a kinship relation. Such a data base can be naturally represented as a directed labelled graph whose nodes represent the people and whose arcs are labelled with relation names. An example of a trivial data base involving seven people and five relations is seen in Figure 1.

FIGURE 1



Let us first consider the directed path p_1--Parent--p_2--Parent--p_3 in the above figure. Closely associated with this path is an abstracted structure called a labelled path sequence (LPS) which results from deleting the nodes of a path but preserving the labels. For the above path its associated LPS is simply "Parent-Parent". There are, however, several other directed paths from p_1 to p_3. We must consider how this entire collection of paths or LPS's relates to the various intensional definitions of the relations making up these LPS's:

The common intensional definition of a grandparent is a parent of a parent or:

DEF1: Grandparent <=> Parent/Parent

where "/" denotes composition

This definition involves a two way implication which expresses two interrelated rules which characterize the concept of grandparent. These rules are:

1.1) If x is the Grandfather of y then there is a z such that x is the Parent of z and z is the Parent of y

1.2) For every x,y,z such that x is the Parent of z and z is the Parent of y, then x must also be the Grandparent of y.

In terms of the above figure, rule 1.2 predicts that there must be at least one additional LPS from p_1 to p_3 (i.e.

"Grandparent"). Of course, this LPS might not explicitely exist in the data base but is implied from the existence of the LPS "Parent-Parent". Continuing our inspection of Figure 1 we consider the more complex path p_1-Parent-p_2-Spouse-p_4-Parent-p_5 or its equivalent LPS "Parent-Spouse-Parent". An intensional definition of Parent is:

DEF2: Parent <=> Spouse/Parent

which again says two things:

2.1) If x is the Parent of y then there is a z such that x is the Spouse of z and z is the Parent of y.

2.2) For any x,y,z such that x is the Spouse of z and z is the Parent of y, then x must also be the Parent of z.

Rule 2.2 predicts that if there exists an LPS "Spouse-Parent", then there must exist (implicitely or explicitely) another LPS "Parent". Each definition, like the above, can be recast as context free grammar rules for a language whose sentences consist of all the potential labelled path sequences. From a generative viewpoint the first two definitions form the generative context free rewrite rules:

1.1 Grandparent ==> Parent Parent
2.1 Parent ==> Spouse Parent

Since these rules embody part of their intensional definitions and are "meaning preserving", we can use these rules (plus others) as a grammar for the sublanguage of all LPS's whose meaning entails "Grandparent". Examples of this potentially infinite sublanguage are:

Grandparent
Parent-Parent
Parent-Spouse-Parent
Spouse-Parent-Spouse-Parent
Parent-Sibling-Parent
Parent-Sibling-Sibling-Parent
.
.
.

If we were to ask the question 'is x the Grandparent of y', then finding any LPS in the data graph (underlying a path from x to y) which is a sentence in the above sublanguage would imply that the given LPS had the same meaning as "Grandparent" and hence was true. (Of course not all LPS's from x to y entail Grandparent as for example Uncle-Parent and therefore only the LPS's that <u>necessarily</u> entail Grandparent should be part of this sublanguage.) The problem of determining if (x Grandparent y) is true then reduces to searching out all possible LPS's from x to y and seeing if any of them are contained in the above infinite sublanguage. But the problem of deciding this is precisely what a recognition grammar (parser) is for!

Returning to the above definitions we see that the second part of each of them is formulatable as a context free recognition rule, i.e.

1.2 Parent Parent ==> Grandparent
2.2 Spouse Parent ==> Parent

and hence can be legitimately used as recognition rules for this sublanguage. In other words, the rules stemming from the intensional definitions of these relations can be used to form a grammar for either recognition or for generation.

The barest sketch of a question-answering strategy thus emerges: to determine if (x R y) is true, first find all LPS's which start at x and terminate at y. Then parse each sentence (LPS) to determine if it can be reduced to the relation R. If there is one such sentence that can be reduced to R, then (x R y) is necessarily true even if there is no direct arc from x to y labelled R. Thusly used, the parser is functioning as the inference maker and its grammar as the set of inference rules.

Admittedly, the above exposition has glossed over numerous logical problems, but before delving into these we think it is important to provide an intuitive explanation of the connections between this approach to inference generation and language recognition.

We now proceed to discuss the details of our question-answering system.

The Data Base

As mentioned earlier, the data base consists of (x R y) triples where R is a given binary relation. The data base will be sparse in the sense that not all of the facts about the given "world" will be contained explicitly in the data base. (Otherwise, no deductive capability whatsoever would be required of our question-answerer.)

It is crucial to our system that for any (x R y) triple stored in the data base its converse also be explicitly stored. This requirement assures us that whenever we wish to prove that a particular (x R y) fact is implicitly contained in the base we need only search along directed paths from x to y in order to obtain all "relevant" information.

The result of such a search is an LPS, or a sentence, connecting x to y. The search algorithm returns only paths which do not involve loops. The importance of not allowing loops is crucial. It provides us with the basic tool for handling axioms or rules that would otherwise require an explicit statement of inequality within the rule. For

example, suppose we have the small set of facts:

1) (Jack Brother John)
2) (John Brother Jim)

and we ask the question: "Is Jim Jack's brother?". Calling our search routine would produce the sentence: "Brother-Brother" which links Jack to Jim and our task would be to determine if this sentence implied "Brother". At first glance this would seem to be the case since Brotherhood is a transitive relation (i.e. Brother/Brother => Brother). However, this rule is not precisely true since Brotherhood is only "almost" transitive and what we really mean is:

$$(x)(y)\exists z\ [(x\ \text{Brother}\ z)\ \&\ (z\ \text{Brother}\ y)\ \&\ x\neq y) \Rightarrow (x\ \text{Brother}\ y)].$$

In other words, we must explicitly check to see that x and y are not equal before we perform the reduction of "Brother-Brother" => "Brother". Because our search algorithm does not permit loops, the check is implicit in the generation of a sentence and therefore relieves us from having to check for this property. (We shall see later that there are some additional problems with "equality" that are not so easily disposed of).

Rules and Definitions

The only rules of inference for this system are context free rewrite rules. These rules must characterize all the sentences that are paraphrases of any of the given relations.

The relations fall into two classes. The first class is atomic relations which are considered to be the primitive relations from which all the other relations can be intensionally defined. Table 1 gives a list of some of the atomic and non-atomic relations to be considered.

An important aspect of our system is that it requires rules interrelating the "atomic" relations and for the remaining relations it only needs their intensional definitions. (Multiple definitions are easily handled and, in some cases, can increase the system's efficiency). Since the number of atomic relations is often significantly smaller than the entire collection of relations we are freed from having to characterize all the possible interactions among all the relations. That is to say, all we must do is characterize the interactions among the atomic relations and require that the inferencing procedures be able to operate on intensional definitions of non-atomic relations in a sufficiently powerful way to free us from having to specify all possible interactions between these relations.

TABLE 1

Partial List of Inference Rules for the Atomic Relations

1)	SPOUSE/PARENT	=>	PARENT
2)	SPOUSE/PARENT	=>	PARENT
3)	PARENT/SIBLING	=>	PARENT
4)	SIBLING/SIBLING	=>	SIBLING
5)	OFFSPRING/PARENT	=>	SIBLING
6)	SIBLING/OFFSPRING	=>	OFFSPRING
7)	SIBLING/OFFSPRING	=>	OFFSPRING

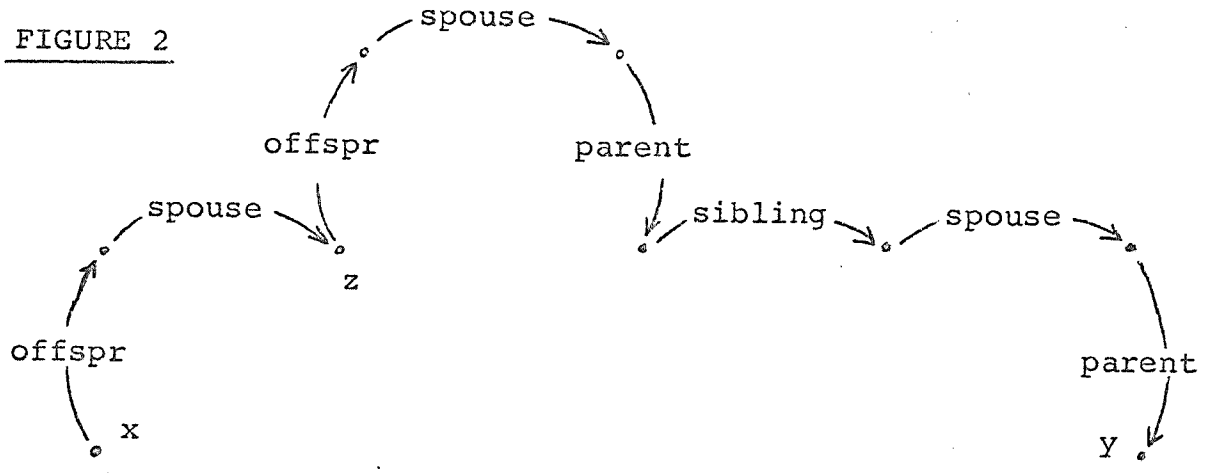
TABLE 2

Partial List of Intensional Definitions for the Non-atomic Relations

UNCLE	=	BROTHER/PARENT $\vee$ HUSBAND/SISTER/PARENT
COUSIN	=	OFFSPRING/SIBLING/PARENT
SISTER-IN-LAW	=	WIFE/SIBLING $\vee$ WIFE/SIBLING/SPOUSE $\vee$ SISTER/SPOUSE
GRANDFATHER	=	FATHER/PARENT

Let us consider an application of these rules. Suppose we wish to determine if x is the cousin of y where there is no explicit link between x and y. Our first step is to search the data base for a directed path from x to y. In so doing, suppose we discover the following path:

FIGURE 2



which translates into the sentence:

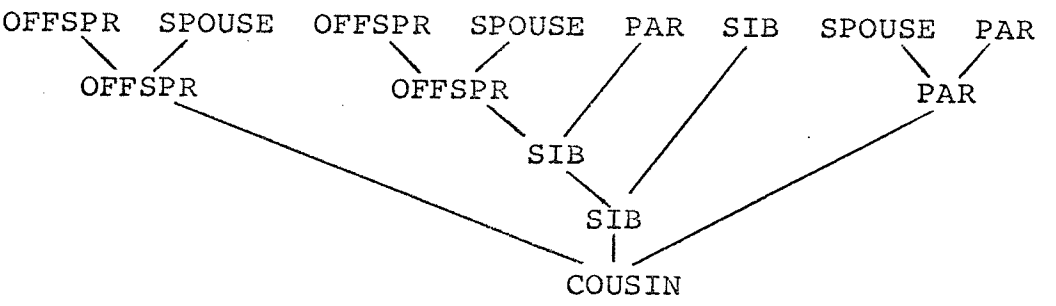
(Offspring Spouse Offspring Spouse Parent Sib Spouse Parent)

We must attempt to reduce this sentence to determine if it implies the cousin relation. Retrieving the intensional definition of cousin (see Table 2) we adjoin it to the context free inference rules as:

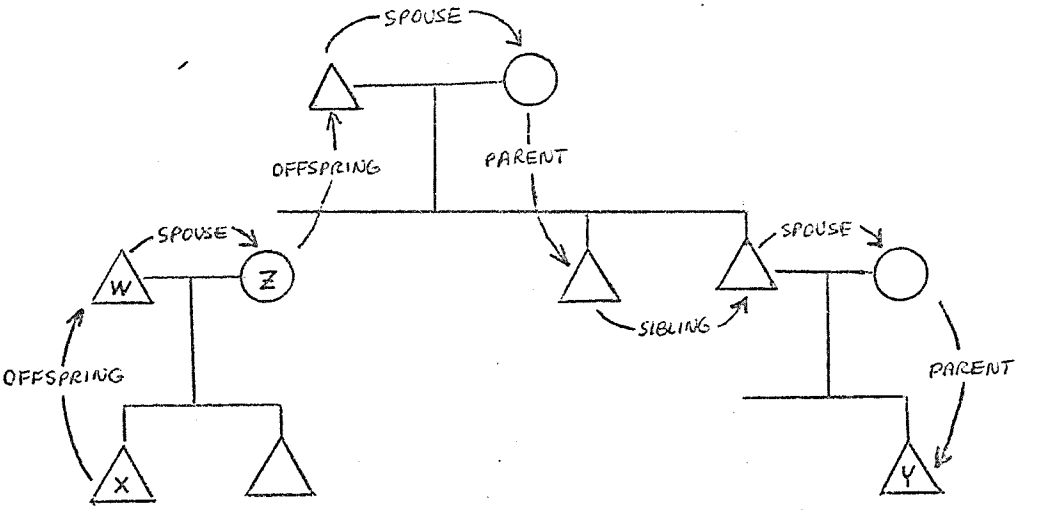
OFFSPRING SIBLING PARENT => COUSIN

We can now attempt to parse this sentence. Figure 3 illustrates such a parse. Since the sentence can be reduced to "COUSIN" it implies that x and y are indeed cousins.

FIGURE 3

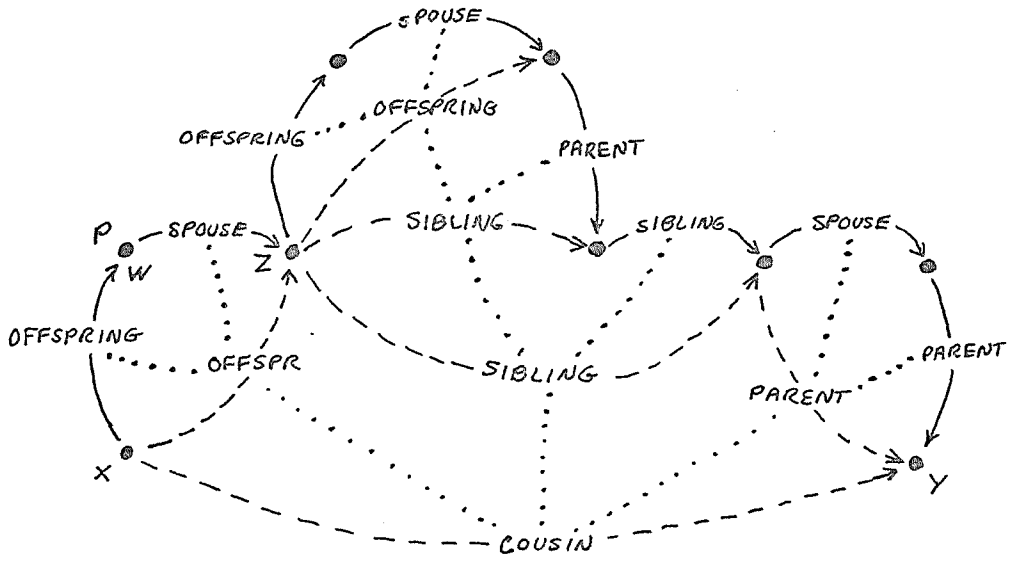


To help in understanding the semantics of this example we provide a family tree consistent with this path.



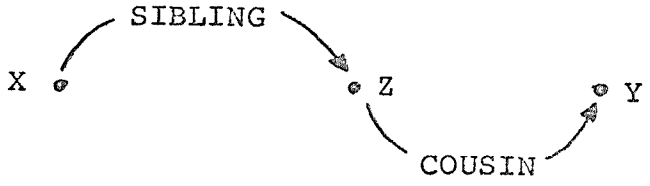
Examining just the application of the first rule, we see that rewriting "OFFSPRING SPOUSE" as "OFFSPRING" is tantamount to specifying an <u>implicit</u> new link between x and z, i.e. if x is the offspring of w and w is the spouse of z, then x must also be the offspring of z. This fact is true regardless of whether or not (x, Offspring, z) is

explicitly contained in the data. We therefore have:



In the chart above, the dashed lines are implicit (added) links and the dotted lines from these implicit links define which links caused their generation. Note that a "complete" data base would necessarily have <u>all</u> these implicit links represented as actual facts or, equivalently, as explicit links. This example seems more complicated than it really is. Note that every arc underlying the initial sentence is an atomic relation. This means that the sentence is already tailored to coincide with the rules of inference. Of course it is possible for a path to be generated which involves a non-atomic relation. What will happen in such a case? If the rules of inference do not include a reduction of the given non-atomic relation, then that part of the sentence must be irreducible.

Again, suppose we are trying to determine if x is the cousin of y and, in searching our data base, we discover:



Since a sibling of a cousin is a cousin, one might expect that if our rules of inference are complete, they should contain the rule:

SIBLING COUSIN => COUSIN.

However, we have agreed earlier that knowing the definitions of non-atomic relations in terms of the atomic relations (which are characterized by the rules of inference) should free us from having to spell out all the rules governing all possible interactions among the non-atomic relations.

In the previous example we used the definition:

COUSIN = OFFSPRING/SIBLING/PARENT

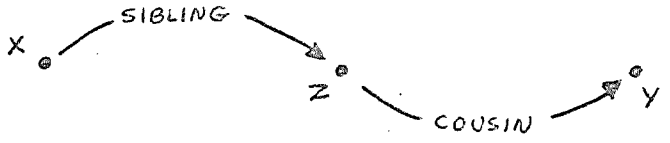
to arrive at the rule:

OFFSPRING SIBLING PARENT => COUSIN.

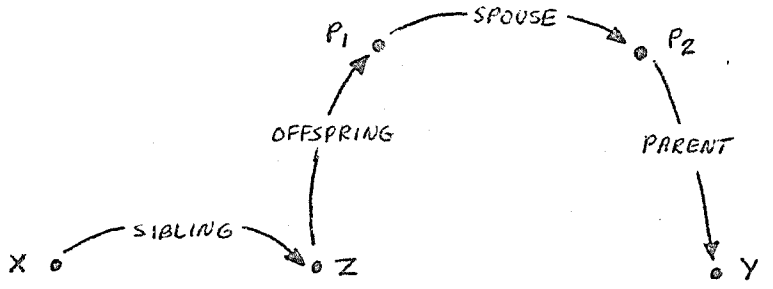
since definitions actually represent two-way implications, we know that if x is the cousin of y there must exist at least two other people (i.e. p_1 and p_2, which are not necessarily referenced in the data base) forming a hypothetical path:

OFFSPRING-(p_1)-SIBLING (p_2)-PARENT.

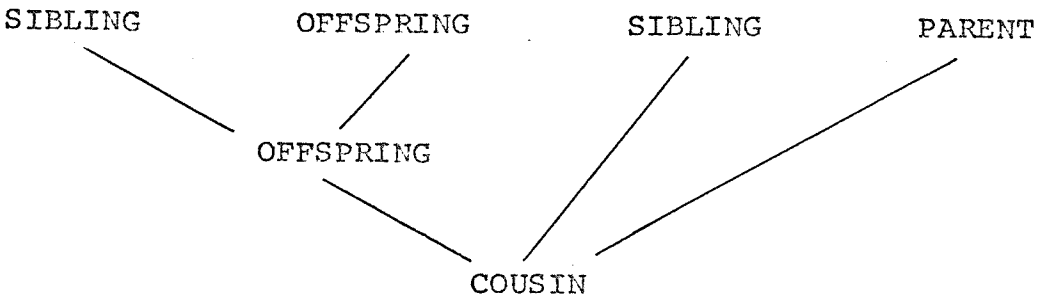
From another point of view what we have done is to take the intensional definition of cousin and split it into a context free rewrite rule and into a lexical rewrite rule. The lexical rewrite rule operates on the initial chart (path) before any of the reduction rules are applied. Thus the lexical rewrite rule transforms the initial chart:



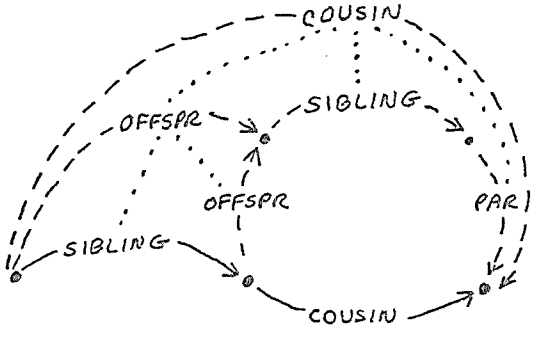
into the new chart:



If we now apply the parser (inference mechanism) to this chart we achieve the following parses:



In terms of a phrase tree the above chart is simply:



We note that this parse is achieved without the need for the explicit inference rule:

SIBLING COUSIN => COUSIN

The above use of lexical rewrite rules circumvents some of the limitations of question-answering systems which force

the binding of all of their existential variables in a given rule to explicit data. For example, in the TRAMP system [5], even with the inference rule "SIBLING/OFFSPRING => OFFSPRING", the above reduction could not be achieved unless the data base explicitly mentioned p_f. Likewise the obvious way to use Micro-planner would run into the same problem although the use of Skolem functions would enable us to circumvent such problems.

Unfortunately, the problems inherent in not knowing the identity of the person through which these hypothetical paths (i.e. those generated by the lexical rewrite rules) pass are not so easily solved. The following set of examples sheds considerable light on the nature of these problems and provides us with an introduction to questions which refer to possible states of the world.

The Handling of "Can" Questions

A characteristic of all the above examples is that each question has a unique answer. That is, there is no uncertainty as to whether the sibling of a cousin is a cousin. If, however, we pose the question:

"Is a cousin of a cousin a cousin?"

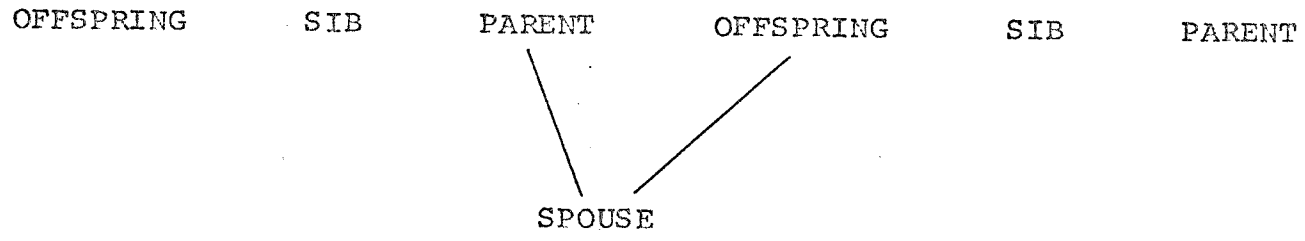
a significantly different situation is encountered. A cousin of a cousin might be a cousin, but then again it might not be. Additional information about the particular case at hand is clearly required. In other words, the rule:

COUSIN COUSIN => COUSIN

sometimes leads to incorrect conclusions and therefore is not a sound inference rule. Expanding each occurrence of cousin by its intensional definition indicates where the uncertainty resides. The expanded path of COUSIN-COUSIN is:

OFFSPRING-SIBLING-PARENT-OFFSPRING-SIBLING-PARENT

which cannot be parsed any further than:

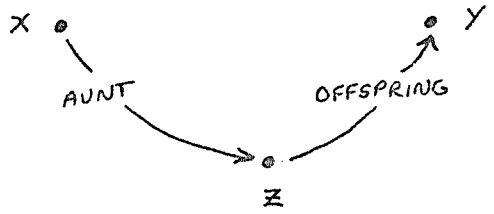


Suppose we make explicit the existentially quantified persons that occur in the definition of cousin:

x Offspr-(p_1)-Sib-(p_2)-Par-(p_3)-Offspr-(p_4)-Sib-(p_5)-Par y

(p_2 Spouse p_4)

This reduction makes the assumption that persons p_2 and p_4 are distinct and therefore the inference rule asserts that they are married. However, we have no guarantee that this assertion is correct since a parent of an offspring would be a spouse only if no loops had been permitted in the generation of this sentence (LPS). In this particular case, the expanded sentence was <u>not</u> generated by an explicit transversal of the data graph. Therefore, even though the original sentence was generated under this restriction, the mechanism which grants such a guarantee has not been invoked on the expanded sentence. A moment's reflection reveals that when a cousin of a cousin is still a cousin, a loop is involved; p_2 equals p_4 and PARENT-OFFSPRING reduces to the identity. An example taken from Lindsay (6) better illustrates this problem. Suppose we encounter the path:

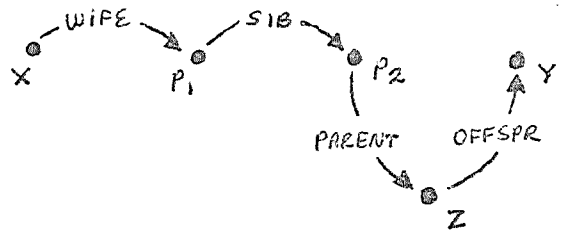


Can x be the sister of y?

The underlying structure of these two examples can best be understood by considering "possible states of the world". Given the two facts:

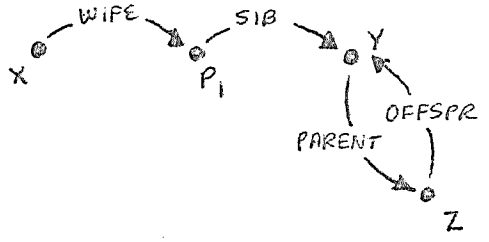
(x, Aunt, y)
(y, Offspring, z)

There are four possible states consistent with these facts:

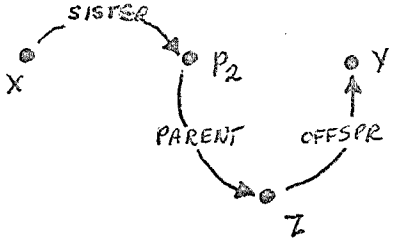


States

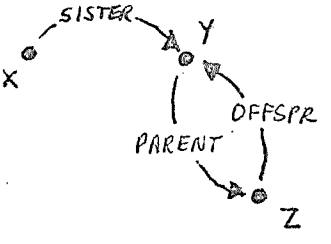
1) $x \neq p_1$
   $y \neq p_2$



2) $x \neq p_1$
   $y = p_2$



3) $x = p_1$
   $y \neq p_2$



4) $x = p_1$
   $y = p_2$

Given no additional information, we have no way of deciding which of the four states reflects the actual situation. Expressed differently, our data base certainly contains explicit references to x and y, but might contain no information about the existentially quantified persons p_1 and p_2. Lacking this information, it is impossible to determine if y equals p_2 or if x equals p_1. Nevertheless, there clearly exists a state in which the aunt of an offspring is a sister (i.e. state 4). A similar analysis holds for the cousin of a cousin. What we would like is some natural way of handling these possible states in case no further information is forthcoming. We emphasize that this problem occurs only when encountering a path involving non-atomic relations for which no additional information is available.

One possible approach to this problem might be to construct a purposely ambiguous grammar (a set of inference rules) which would give multiple parsings of a sentence with each parse reflecting a possible state of the world.

This has been achieved by augmenting the grammar to incorporate the possibility of loops by the following technique: Whenever a path is expanded, a special flag "F" is placed around the inserted definition. For example, the expansion of COUSIN-COUSIN is:

OFFSPRING SIBLING PARENT F OFFSPRING SIBLING PARENT

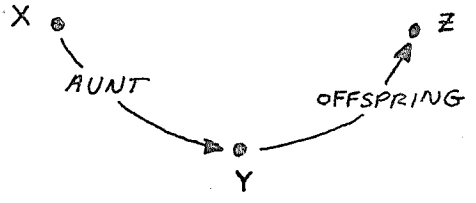
The augmented grammar contains rules of the form:

```
SPOUSE F SPOUSE     => I Married (I is the Identity Relation)
PARENT F OFFSPRING => I
FATHER F OFFSPRING => I Male
MOTHER F SIBLING    => I Female
       F            => Nil
       .
       .
       .
```

(This last rule permits the flag to be turned off so that the initial set of inference rules may be applied)

The application of any of these rules introduces an implicit loop in the derived data graph and thereby allows us to cover implicitly all the possible consequences of a particular sentence. An example will help to clarify how these augmented rules introduce implicit loops.

Let us reconsider the problem of determing if an aunt of an offspring can be a sister. The initial data path was:



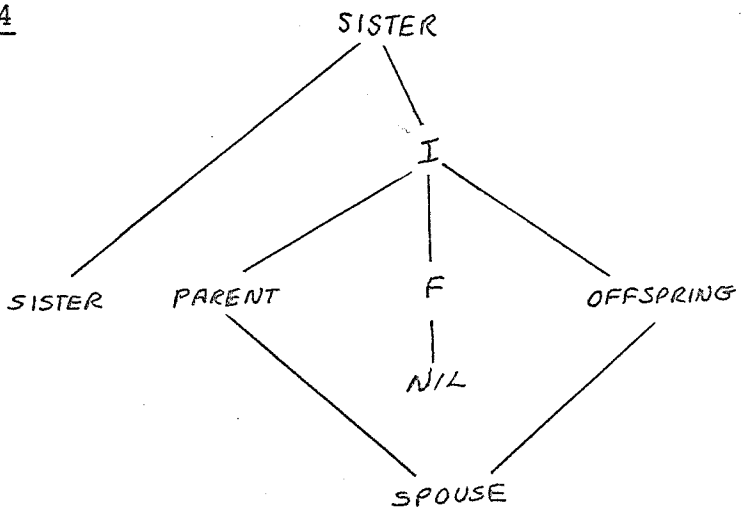
or AUNT-OFFSPRING.

Expanding this path we encounter the following two sentences:

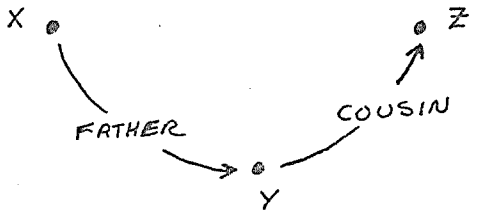
Sentence 1: WIFE SIBLING PARENT F OFFSPRING
Sentence 2: SISTER PARENT F OFFSPRING

Restricting our attention to the latter sentence, we find two possible reductions. These are listed in Figure 4 along with the data paths consistent with them.

Figure 4



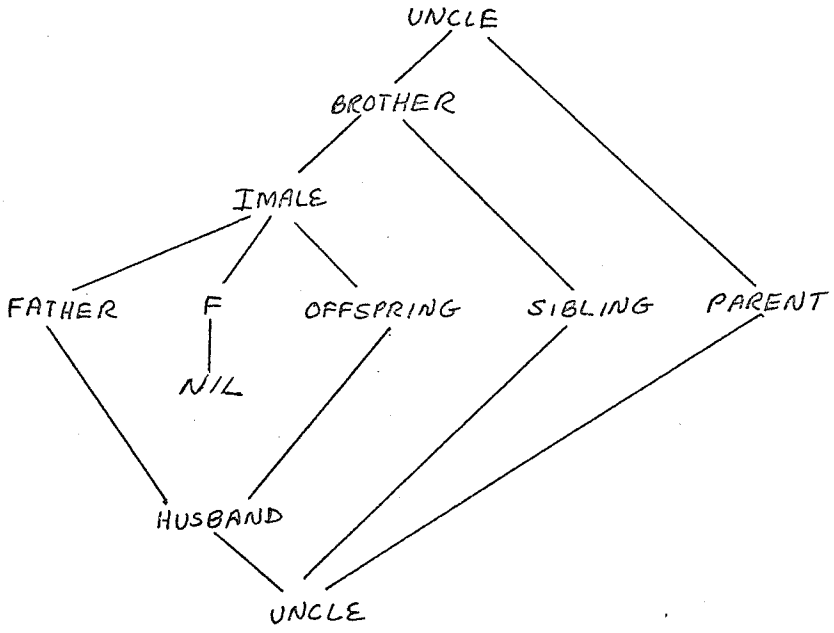
The only basic question that remains unexplored is how to derive conclusions which are certain but which involve sentences containing non-atomic relations. For example, suppose we encounter a data path:



Is x the uncle of z? The expansion of this path is:

FATHER F OFFSPRING SIBLING PARENT.

Because of the flag F we now find that there are two possible parses. One of these reduces the sentence to "BROTHER PARENT" and then to "Uncle"; the other reduces first to "HUSBAND SIBLING PARENT" and then to "UNCLE".



Since both parses reduce to Uncle, and since one involves using the flag and the other does not, we are assured that in either case this sentence implies the "Uncle" relation

(i.e it is true in all possible "states" of the world).

The Question-Answerer

The above sections have highlighted some of the more interesting aspects of this approach to question-answering. We now provide a brief description of our system which consists of several basic blocks of LISP procedures. The first of these parses an input statement, determining whether it is a fact or a question. If the statement is a fact, it stores the statement and its converse in the dynamically grown data graph. If the statement is a question, the first block formats it as an (x, R, y) triple and passes it on to the second block. This block is responsible for searching the data graph and thus constructing all possible LPS's that bridge the (x, y) 2-tuple mentioned in the statement.

After each LPS is generated, a check is made to see if it contains any non-atomic relations. If it does, it is pushed onto a hold list and the search is continued for another LPS. Whenever an LPS consisting solely of atomic relations is encountered, that LPS is immediately passed to the inference mechanism (parser). If the inferencer fails to reduce this sentence to the desired relation, control is

passed back to the search routine which picks up where it left off. If a successful reduction is performed, control is immediately returned to the top level with the appropriate response.

If after all possible LPS's are discovered and no successful reduction has occurred for those LPS's involving only atomic terms, then the third block is entered which begins processing the above-mentioned hold stack. Each of these LPS's is, in turn, popped off this stack, expanded, (note that if the non-atomic relation has a disjunctive definition, then a sentence is created for each disjunct) and passed to the inferencer.

Limitations

There are numerous limitations to this simple question answering system and a few are worthy of special attention. We first note that our rules of inference are limited to compositional rules and thus cannot express certain structural properties that might require the intersection operator. An example of such a non-compositional rule is:

> "If x is the father of y and spouse of z, then x is the husband of z."

Another kind of limitation is our inability to express (in this formalization) constraints dealing with the number of distinct objects satisfying some relation. For example, a person has exactly four distinct grandparents (divorce, etc. has always been excluded). Some deductions clearly might require knowing this.

The most severe limitation is our inability to handle, in any interesting way, what things cannot be true. For example, suppose we ask if x is the Uncle of y and discover that x is the father of y. The only way we can answer "no" to this question is to have an explicit rule that says something like "FATHER => ¬ UNCLE". But such a solution bypasses the interesting and profound problems of "inferring" the above rule by only knowing that siblings can't marry. In other words, the only way that x could be the father and uncle of y is for x to be married to his own sister!

Conclusion

Appendix 1 contains some examples of this system. Most of these questions including parsing times were answered in about 100 msec per question with some taking as little as 17 msec. These times appear to be quite impressive especially

if compared with the response times obtained with more general theorem proving approaches.*

We stress that this approach to deduction is not suitable for truly general purpose question-answering systems. However we do believe that new avenues for achieving special types of deduction should be explored and we offer this scheme as such an example. In addition we hope that this treatment of question-answering will at least lead to a greater appreciation of the similarity between grammars and inference rules on the one hand, and parsers and theorem-provers on the other.

---

* These times were achieved on a PDP-10 running UCI LISP. The axiom base consisted of approximately 100 axioms not counting the "facts" making up the data base.

BIBLIOGRAPHY

1. Kay, Martin, "The MIND System" in Natural Language Processing, Randall Rustin, ed., Algorithmics Press, Inc., New York, 1973

2. Kay, Martin, Experiments with a Powerful Parser, The Rand Corp, Santa Monica, Calif. 1967, RM-5452-PR.

3. Woods, W.A., "An Experimental Parsing System for Transition Network Grammars," in Natural Language Processing, 1973

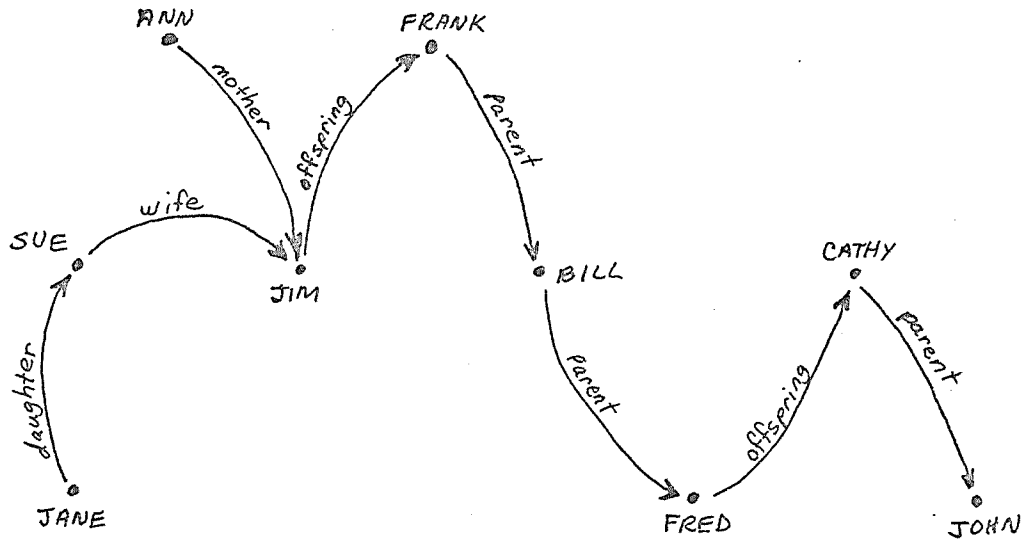
4. Kaplan, Ronald M., "A General Syntactic Processor," Natural Language Processing, 1963

5. Ash, W. L., and E. H. Sibley, "TRAMP: An Interpretive Associative Processor with Deductive Capabilities," Proceeding 23, Nat'l. Conference of the ACM, Las Vegas, August 1968.

6. Lindsay, R. K., "Inferential Memory as the Basis of Machines which Understand Natural Language," in Computers and Thought, Edward Fiegenbaum and Julian Feldman, eds., New York, McGraw-Hill

APPENDIX 1

Data Base:



Questions:

Q: How is Cathy related to Bill?
A: Spouse
TIME: 17 ms

Q: How is Jane related to John?
A: Cousin
TIME: 267 ms

Q: Is Jim the parent of Jane?
A: Yes
TIME: 50 ms

Q: What is a brother-in-law's wife?
A: Sister or sister-in-law
TIME: 217 ms

Q: What is a spouse's mother?
A: Mother-in-law
TIME: 17 ms

Q: How is Jane related to Ann?
A: Granddaughter or grandchild
TIME: 83 ms

Q: What is a sibling of a cousin?
A: Cousin
TIME: 50 ms

Q: What is an aunt of an offspring?
A: Sister or sibling or sister-in-law
TIME: 100 ms

Q: What is a daughter of a spouse of a daughter of a father?
A: Niece
TIME: 166 ms