MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1096 October 1988

# Using English for Indexing and Retrieving

**Boris Katz**

**Abstract**

This paper describes a natural language system START. The system analyzes English text and automatically transforms it into an appropriate representation, the *knowledge base*, which incorporates the information found in the text. The user gains access to information stored in the knowledge base by querying it in English. The system analyzes the query and decides through a matching process what information in the knowledge base is relevant to the question. Then it retrieves this information and formulates its response also in English.



This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institue of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defence under Office of Naval Research contract N00014-85-K-0124.

A shorter version of this paper appears in the *Proceedings of the Conference on User-oriented Content-based Text and Image Handling, RIAO '88.*

## 1. Introduction

This paper describes a natural language system START (SynTactic Analysis using Reversible Transformations). The system analyzes English text and automatically transforms it into an appropriate representation, the *knowledge base*, which incorporates the information found in the text. The user gains access to information stored in the knowledge base by querying it in English. The system analyzes the query and decides through a matching process what information in the knowledge base is relevant to the question. Then it retrieves this information and formulates its response also in English.

Researchers at MIT, Stanford University, and the Jet Propulsion Laboratory, have used the START system for creating knowledge bases from English text and for information retrieval in domains as diverse as medicine, politics, space, vision, and commonsense physics. (See, for instance, Winston [1982; 1984; Winston *et al.*, 1983], Doyle [1984], Katz and Brooks [1987], McLaughlin [1987; 1988]). In sections 11, 12, and 13 of this paper we present examples of actual dialogs with START in several different domains.

## 2. Understanding Language

Before we provide a detailed description of the START system we should make clear what we mean by "understanding." What does it mean for a machine to understand language?

Let us consider a situation where a mother gives instructions or tells a story to her daughter Jill. Hopefully, this child has "stored" the new information/knowledge in her memory. In this case, we say that the child understood her mother.

How can this be verified?

- If Jill heard a story, her mother can ask questions relevant to the story. If after searching her memory, Jill is able to utilize the acquired knowledge and answer the questions correctly, then she understood the story.
- Suppose instead of a story Jill heard a set of instructions for a task she is supposed to perform. If Jill is able to retrieve the knowledge given by the instructions and accomplish the task, then she understood the instructions.

We will use these two criteria as "Turing tests" to help us define what it means for a computer to understand English:

A. English text is typed into the computer and on the basis of this text a knowledge base is created. The user queries the knowledge base in English. If the computer's responses are correct, then we can say the computer understood the text.

B. A sequence of English commands or instructions is entered in the computer. If the computer carries out appropriate actions in response to them, then we can say the computer understood the instructions.

Remarkably, START passes these two tests in a variety of situations although it possesses no explicit theory of meaning.

As suggested by these two criteria, knowledge bases created by START can either be used in question-answering situations or they can provide input data for other computer systems. The START knowledge base is employed by both the language understanding and generating modules. These two modules also share the same Grammar (see Katz [1980], Katz and Winston [1982] for a detailed description of the Grammar). Moreover, most of the definitions, techniques, and constructs of START may be discussed in both the understanding and the generating mode. As it seems appropriate, we will use either of the two modes to explain how START works.

## 3. Kernel Sentences

Most English sentences break up into units that we will call *kernel sentences.* For instance, the following aphorism can be broken up into five smaller units:

(1) If the orator wants to persuade people, he must speak the things people wish to hear.

(S1) The orator$_i$ wants S2.

(S2) The orator$_i$ persuades people.

(S3) He$_i$ must speak the things$_j$.

(S4) People$_k$ wish S5.

(S5) People$_k$ hear the things$_j$.

One of the goals of the START system is to identify these smaller units and to determine how they assemble together in a larger sentence.

Before we can formally define kernel sentences let us examine how START represents sentence elements internally. The system uses three types of building blocks for constructing a kernel sentence, the *noun-template*, the *verb-*

*template*, and the *adverb-template*:

noun-template (NT) = [prep det mod adj* noun]

verb-template (VT) = [aux1 neg aux2 aux3 verb]

adverb-template (AT) = [mod adverb]

Here *prep, det, mod, adj, aux, neg* are, respectively, abbreviations for preposition, determiner, modifier, adjective, auxiliary, and negation. The superscript * indicates that a string of one or more symbols or their conjunction is allowed. All the elements in the templates are optional. Noun phrases and prepositional phrases in English can be constructed by reading off the slot values in instantiated noun-templates. In a similar way, verbs and their auxiliaries can be obtained by reading off slot values in verb-templates. For example, the well-formed instantiation of the noun-template, NT = [(noun **Mary**)], where most elements of the NT are omitted, produces a simple noun phrase *Mary*, but the same template with all its elements filled,

NT = [(prep **after**) (det **a**) (mod **very**) (adj **long**) (noun **flight**)]

generates a full-fledged prepositional phrase *after a very long flight*. Similarly, the verb-template may produce either one main verb, *launch*, when using VT = [(verb **launch**)], or a more complex string like *could have been watching*:

VT = [(aux1 **could**) (aux2 **have**) (aux3 **been**) (verb **watching**)]

Templates can be combined to represent a conjunction, for instance, *Jessica, Gabriella and Miriam.*

Notice that in our grammar the noun-templates are used for constructing both noun phrases and prepositional phrases. The existence of languages with nominal case marking, like Russian, indicates that the idea of having a prepositional element within a noun-template is not as far-fetched as it may seem (see Katz and Winston [1982]).

Now let us define a *kernel structure* as the following sequence of templates:

(2) $NT^{initial}\ NT^{subject}\ VT\ NT^{object}\ NT^{final}$

Here $NT^{subject}$ and $NT^{object}$ are noun-templates that represent the subject and the object in the sentence; $NT^{initial}$ and $NT^{final}$ represent its initial and final prepositional phrases; $VT$ is a verb-template.

We should point out that most of the elements in the kernel structure are optional. Also for simplicity the adverb-templates have been omitted although they may appear in (2) in a number of places. In addition, transformational rules, introduced in section 6, are allowed to modify the kernel structure and change the order of templates.

A *kernel sentence* is an English sentence obtained by reading off from left to right all slot values in all templates in (2). Notice that the kernel structure contains only one verb-template and can therefore represent only a limited class of English sentences, without any embedded clauses. However, as we will see in sections 6 and 7, kernel structures can be recursively combined in several different ways to account for a large variety of English sentences, including complex sentences involving embedding.

We define *parsing* as a process of syntactic analysis which, given an English kernel sentence as input, produces the corresponding kernel structure as output.[1] For example, given the sentence below:

(3) After the launch the commander will give additional instructions to the astronaut

the *parser* produces the following instantiated kernel structure where the order of templates follows that of (2):

| | |
|---|---|
| $NT^{initial}$ | [(prep **after**) (det **the**) (noun **launch**)] |
| $NT^{subject}$ | [(det **the**) (noun **commander**)] |
| $VT$ | [(aux1 **will**) (verb **give**)] |
| $NT^{object}$ | [(adj **additional**) (noun **instructions**)] |
| $NT^{final}$ | [(prep **to**) (det **the**) (noun **astronaut**)] |

Many natural language understanding systems restrict themselves to the parsing process just defined and stop there. However, this is clearly not enough to satisfy our definition of understanding. It is not enough to teach the computer to recognize different syntactic categories and fill in the slots in the kernel structure. Our goal is to enable the computer to use the knowledge encoded in the kernel structure; in other words, to *index* and *retrieve* this knowledge efficiently.

## 4. From Kernel Structures to T-expressions

Recall that in order to understand her mother, Jill had to perform two important operations. When listening to the story, Jill had to store (or *index*) the new knowledge in her memory. When answering the questions, she had to search her memory and *retrieve* the knowledge. These two operations, indexing and retrieving, are crucial in our model of understanding language. In this section we will describe the indexing procedure of START. The retrieval task is carried out by a matching procedure described in section 11.

[1] In sections 6 and 7 this notion of parsing is extended to a wider class of English sentences. See also sections 12 and 13 for examples from the "real world".

Suppose we type the following English sentence on a computer terminal:

(4) Jane will meet Paul tomorrow

and the parsing procedure constructs the appropriate kernel structure. Now, there are many things about this sentence that the computer should remember: that *Jane* is the subject of the sentence, *Paul* is the object, that *meet* is the relation between them. There is more to remember: the tense and the aspect of the sentence, its auxiliaries, its adverbs. Was this sentence embedded in a larger sentence? Does it have a relative clause? Was the verb in the active or passive form? We certainly want all this information about the sentence to be stored in the computer's memory. However, we also want to be able to retrieve this information efficiently.

We could store all these sentence features in one long list. This approach, however, would not account for the fact that some of the features in a sentence seem more salient than others. A simple list of features in sentence (4) would also fail to capture its structural affinity with the following two sentences:

(5) Yesterday Jane could have met Paul.

(6) Paul wasn't met by Jane.

And finally, this approach would turn the matching/retrieval task into a computational nightmare.

We could try to emphasize the hierarchical nature of the English sentence by using the kernel structure representation. However, since most elements of the kernel structure are optional, its shape is too unpredictable to allow the system to match the kernel structures efficiently.

Our system, START, rearranges the elements of the kernel structure by tying together the three most salient parameters of a sentence: the subject, the object, and the relation between them. These are combined into *ternary expressions (T-expressions)* of the form <**subject relation object**>. For example, sentence (4) will yield the *T*-expression

(7) <**Jane meet Paul**>

Note that the relation of this *T*-expression is the infinitive form of the main verb in the sentence.

Certain other parameters are used to create additional *T*-expressions in which prepositions and several special words serve as relations. The remaining parameters, adverbs and their position, tense, auxiliaries, voice, negation, *etc.*, are recorded in a representational structure called *history*. The history has a *page* pertaining to each sentence which yields the given *T*-expression. When we index the *T*-expression in the knowledge base, we cross-reference its three

components and attach the history $H$ to it: **<subject relation object>**$_H$.

For example, suppose START is analyzing a text containing sentence (5) and later sentence (6). The system would produce the $T$-expression

(8) **<Jane meet Paul>**$_H$

with a two-page history. One page would contain additional information about the first sentence, such as the fact that sentence (5) used perfective "have" and the modal "could", and that it starts with an adverb "yesterday". The second page would show the use of passive form and negation in sentence (6). One can thus think of a $T$-expression as a "digested summary" of the syntactic structure of a proposition and of its use within an English text.

The $T$-expression is the cornerstone of the representational hierarchy of the START system. It is the level of the hierarchy where the understanding and the generating modules meet. The understanding module analyzes English sentences and creates a set of $T$-expressions stored in the knowledge base. Given a set of $T$-expressions as an input, the generating module produces English text.

## 5. Referents for Noun Phrases

The subject and the object of $T$-expression **<Jane meet Paul>** are proper names which are taken directly from sentence (4). However, the process is more complex if, for example, the subject of a sentence is not a proper name but a complex noun phrase:

(9) Jane's good friend from Boston met Paul.

In sentence (9) the system needs to establish the referent for the head noun, *friend*. The system has to come up with a *unique name* for this noun in case a different instance of *friend* appears later in the analyzed text. In order to do this, START computes the *name environment* $E_1$ for this occurrence of *friend*. We define $E_1$ as a list of adjectives (in this case, *good*), possessive nouns (*Jane's*), prepositional phrases (*from Boston*), *etc.* modifying that noun in the present sentence. Then START associates with this environment a unique name, say *friend-1*, which we will call a *referent*[2] for the noun *friend* in the environment $E_1$. The main $T$-expression for sentence (9) will therefore take the form **<friend-1 meet Paul>**.[3]

---

[2] In calling a unique name a referent we deviate from standard usage, which reserves the term for an object in the world.

[3] The analysis of the subject noun-phrase in sentence (9) will produce three additional $T$-expressions (see section 7).

The analyzed noun, *friend*, its name environment $E_1$, and its referent, *friend-1* are then recorded in the computer's memory. This bookkeeping gives the system the ability to compute efficiently the referent of a noun given its name environment. If, for instance, the same noun phrase,

(10) Jane's good friend from Boston

occurs again in a different sentence later in the text, then its name environment would coincide with $E_1$ and hence the same referent, *friend-1*, would be retrieved and utilized in the $T$-expressions constructed for this sentence.

Suppose now that START encounters a new sentence where the noun *friend* appears in a slightly different noun phrase like

(11) Tracy's good friend from Pasadena.

The environment $E_2$ associated with the new noun phrase is different from $E_1$ and is not to be found in the computer memory. This means that there is no referent readily available for the noun *friend* in this sentence and the system needs to generate a new unique name, *friend*-2, to be associated with $E_2$.

START recursively employs the procedure just described to find a referent for *every* noun in the sentence. Thus, given the noun phrase

(12) The young woman's good friend from the big city

the system first determines the referents of the nouns *woman* and *city*. Only after that, once the computation of its name environment becomes possible, does the head noun, *friend*, get its referent.[4]

Sometimes, however, the information in the name environment is not sufficient to find referents for noun phrases. For instance, if a noun phrase is modified by a relative clause (see section 7) the entire knowledge base has to be consulted in order to determine the appropriate referent.

## 6. Transformational Rules

The standard kernel structure introduced in section 3,

(2) $NT^{initial}\ NT^{subject}\ VT\ NT^{object}\ NT^{final}$

allows the system to generate or parse only a limited variety of English sentences. To account for other kinds of sentences, START employs commutative *transformational rules* (see Chomsky [1957], Katz [1980]). For instance, consider how the kernel sentence

[4] In the remainder of this paper, for reasons of simplicity, we will use the nouns themselves in $T$-expressions rather than their referents.

(13) The probe reached Venus

is modified by several transformational rules, where each transformation is applied to the outcome of the previous one:

| Transformation | Sentence |
|---|---|
| | The probe reached Venus. |
| Question | Did the probe reach Venus? |
| Negation | Didn't the probe reach Venus? |
| Passive | Wasn't Venus reached by the probe? |

The transformations shown are executed by the generating module of START. In the understanding mode, the system's goal is to recognize which transformations were applied. In some cases, for instance, *Negation*, START simply makes the appropriate additions to the histories of the resulting $T$-expressions. In other cases, the system must actually *reverse* the effect of the transformation.

START uses a set of commutative transformational rules (two transformations are said to commute if they can be applied in either order and both orderings produce the same result). The fact of their commutativity is proved in Katz [1980]. Apart from its theoretical interest, the commutativity of the transformations provides the system with considerable computational advantages because no attention needs to be paid to the ordering of the transformations or to the so-called interactions between them (cf. Akmajian and Heny [1975]).

All the examples of English sentences considered so far have been very simple. We can make them a little more complex by allowing simpler sentences to be embedded in larger sentences, as shown below:

(14) Jessica wanted the computer to print the message.

(15) For Miriam to ignore the request would anger Jessica.

The transformational rules responsible for sentence embedding form a special class called *connective transformations* (see Katz [1980]). Each connective transformation takes two kernel sentences as input; these correspond to the *matrix* clause and the *embedded* clause in the resulting English sentence. For example, sentence (14) above consists of a matrix clause, *Jessica wanted it*, and an embedded clause, *the computer printed the message*. We assume that one of the noun-templates in the kernel structure of the matrix clause always contains *it* as a *joining point* for glueing the two kernels together.

A connective transformation is fully determined by the values of three

parameters, each referring to a position in the kernel structure of the embedded clause: *COMP, NP,* and *INFL.* A different set of values for these parameters results in a different final shape of the English sentence generated using connective transformations. The complementizer *COMP* introduces the embedded clause; the binary parameter *NP* indicates whether the subject of the matrix clause is coreferent with the subject of the embedded clause; the inflectional element *INFL* specifies how the verb in the embedded clause should be inflected. For instance, in sentence (14) the value of *COMP* is *null* (while in (15) its value is *for*) and the value of *INFL* is *to.* The value of the parameter *NP* is set to indicate that the subject of the matrix clause, *Jessica,* is different from the subject of the embedded clause, *the computer.*

The positions of these three parameters in the kernel structure and their actual values are described in Katz [1980]. Table 1 shows examples illustrating the application of several different connective transformations.[5]

| Matrix clause | Embedded clause | Resulting Sentence |
|---|---|---|
| It angered Kirk | The computer ignored the message | That the computer ignored the message angered Kirk |
| Spock suggests it | McCoy is silent | Spock suggests that McCoy be silent |
| Spock watched it | Kirk read the message | Spock watched Kirk read the message |
| Kirk asked it | The computer repeated the message | Kirk asked the computer to repeat the message |
| Spock claims it | Spock has written the message | Spock claims to have written the message |
| It shocked Kirk | Spock ignored the command | Spock's ignoring the command shocked Kirk |
| Spock saw it | McCoy read the message | Spock saw McCoy reading the message |
| It angered Kirk | Kirk read the message | Reading the message angered Kirk |
| Kirk knew it | The computer ignored the message | Kirk knew whether the computer ignored the message |

Table 1. Examples of applications of connective transformations.

Notice that in each example the element *it* in the matrix clause indicates the location where the embedded clause is inserted. The embedded clause is adjusted according to the values of the parameters *COMP, NP,* and *INFL.*

The main verb of the matrix clause plays an important role in both the execution and recognition of connective transformations. It determines the shape which its embedded clause may take and specifies the kinds of connective transformations that may be applied in each particular case. It

[5] Since this family of transformations is completely defined by the values of parameters *COMP, NP,* and *INFL,* it can be considered a single *connective transformation* whose surface manifestation has several different forms depending on the values of these parameters.

is imperative, therefore, that the dictionary entry for any verb which may appear in a matrix clause (that is, can take a sentential complement) contain a list of permissible connective transformations.

Now let us examine how START analyzes embedded sentences. Recall that in section 2 we defined *parsing* only for kernel sentences. Connective transformations allow us to extend this notion to a wider class of English sentences which includes embedded sentences. The process consists of four steps:

1. The system determines the connective transformation involved.
2. The system reverses the connective transformation and splits the sentence into kernel sentences.
3. The system parses each kernel sentence separately and produces kernel structures.
4. The indexing procedure utilizes the lexical material provided by kernel structures to construct and index corresponding $T$-expressions.

In order to handle embedded sentences, START allows any $T$-expression to take another $T$-expression as its subject or object. Thus, sentence (16) leads to *right embedding*:

(16) Jessica wanted the computer to print the message.

(17) **<Jessica want <computer print message>>**

while the sentence (18) leads to *left embedding*:

(18) For Miriam to ignore the request would anger Jessica.

(19) **<<Miriam ignore request> anger Jessica>**

Connective transformations may be recursively applied without any restrictions on the depth of embedding. This means that START can analyze and generate sentences with arbitrarily complex embedded structures.

## 7. Complex Noun Phrases and Relative Clauses

We have seen how START analyzes a sentence and produces a $T$-expression which "summarizes" the syntactic structure of the sentence. In this section we examine sentences with complex noun phrases and show how such noun phrases result in the construction of several additional $T$-expressions. Consider sentence (20):

(20) Jane's good friend from Boston met Paul.

A complex noun phrase in English consists of three components: the *head* around which the other components cluster, the *premodification* which includes all the adjectives and possessive nouns placed before the head, and the *postmodification* which comprises all the items placed after the head, including prepositional phrases, non-finite clauses, and relative clauses (see Quirk and Greenbaum [1973]). In our example, the head of the noun phrase, *friend*, is premodified by *Jane's* and *good* and postmodified by the prepositional phrase *from Boston*. As a result, along with the main $T$-expression, **<friend-1 meet Paul>**, the system will construct three additional $T$-expressions: **<friend-1 is good>**, **<friend-1 related-to Jane>**, and **<friend-1 from Boston>**. In fact, every adjective or possessive noun in the sentence, as well as every prepositional phrase or relative clause, will cause new $T$-expressions to be built and stored in the knowledge base.

START can handle different types of relative clauses:

(21) The girl *who wants to become an astronaut* is young.
(22) The planet *which Voyager photographed yesterday* was shrouded in clouds.
(23) The man *we admire* walked on the Moon.
(24) The planet *the spacecraft flew behind* has a strong magnetic field.
(25) The satellite *to which the antennas were pointing* had an impact crater.
(26) The spacecraft *orbiting the Earth* photographed its surface.
(27) The satellite *launched by NASA* handles telecommunications.
(28) The space shuttle *whose protective tiles were damaged* underwent repairs.

These examples show sentences with full relative clauses, as in (21) or (25), with reduced relative clauses (26, 27), with subject relative pronoun (21), object relative pronoun (22), or even without it (23). The relative pronoun may be governed by case (28) or by a preposition (25). The preposition may precede (25) or follow (24) its complement. Let us analyze sentence (22), which involves a full relative clause with an object relative pronoun. First, the system has to find the relative clause boundaries and identify the location of the *gap* (denoted by $e$) which is coreferent to the head noun phrase:

(29) The planet$_i$ [which Voyager photographed $e_i$ yesterday] was shrouded in clouds.

Then the relative clause is "removed" from the sentence, the gap is filled with its antecedent and the modified clause is processed independently. As a result, the sentence (22) will be split into the following two sentences:

(30) Voyager photographed the planet$_i$ yesterday.

(31) The planet$_i$ was shrouded in clouds.

To indicate that these two sentences go together in a relative clause relation-

ship (cf. Woods [1985]), the history of the $T$-expression of the main clause (31) is provided with a pointer to the $T$-expression of the relative clause (30).

Relative clauses do not need to be simple kernel sentences (see example (21), for instance). In fact, any two sentences that may be analyzed by START, with arbitrarily complex embedded structures, can be combined into main and relative clauses of a larger sentence as long as they have a common noun phrase (see the sample passage in section 12). Moreover, several relative clauses may be recursively embedded inside one another.

## 8. Lexical Ambiguity

Every word in the sample sentences discussed so far was assumed to belong to a unique part of speech. Thus, *Jill, friend*, and *man* are nouns, *read, write*, and *tell* are verbs, and *old* and *good* are adjectives. This assumption however is not always correct. Most words in English can receive several alternative category assignments (that is, can serve as different parts of speech); the particular choice depends on the context. For instance, in the following sentence from a detective story

(32) The gangsters *can supply uniform* alibis

the word *can* is used as a modal auxiliary, but it could also serve as a noun; the word *supply* is a verb, but it could also be a noun or a modifier in a noun-noun modification sequence; the word *uniform* is an adjective that could be used as a noun in a different context. Sentence (32) will be analyzed correctly only if the system selects the right category assignments for each word; any other assignment will result in an error.

Lexical entries in START (see section 10) are allowed to specify more than one category assignment. The system is equipped with a mechanism for category disambiguation which uses error feedback from the parser (including context information and type of error) to efficiently resolve ambiguities. As a result, along with sentence (32), START is able to process successfully another sentence from the same detective story:

(33) But the policeman found the *uniform* in the *supply can.*

Notice that each of the three ambiguous words in sentence (33) is a different part of speech from what it was in (32).

## 9. Forward and Backward S-rules

In sections 4 and 6 we showed how START builds $T$-expressions using the

pattern **<subject relation object>** at every level of embedding. As a consequence, $T$-expressions closely follow the syntax of analyzed sentences. This property incidentally is one reason why the language generator is frequently able to reconstruct the original English sentence almost verbatim. Unfortunately, this property also implies that sentences which have different surface syntax but are close in meaning will not be considered similar by the system.

An example will clarify this point. Given as input the sentence (34) START will create an embedded $T$-expression (35):

(34) Miriam presented Gabriella with a gift

(35) **<<Miriam present Gabriella> with gift>**

whereas a near paraphrase, sentence (36), will generate $T$-expression (37):

(36) Miriam presented a gift to Gabriella.

(37) **<<Miriam present gift> to Gabriella>**

Speakers of English know that sentences (34) and (36) both describe a transfer of possession. In both sentences, *the gift* is the transferred object, *Gabriella* is the recipient of this object, and *Miriam* is the agent of the transfer, despite different syntactic realizations of some of these arguments. It seem natural that this kind of knowledge be available to a natural language system. However, the START system, as described so far, does not consider $T$-expressions (35) and (37), which are associated with these sentences, to be similar.

The difference in the $T$-expressions becomes particularly problematic when START is asked a question. Suppose, for example, that the input text contains only one *present* sentence, (38), and the knowledge base contains only the corresponding $T$-expression, (39):

(38) Miriam presented Gabriella with a gift.

(39) **<<Miriam present Gabriella> with gift>**

Now suppose the user asked the following question:

(40) To whom did Miriam present a gift?

Although a speaker of English could easily answer this question after being told sentence (38), the START system, as described so far, would not be able to answer it. This question presents a problem for START because $T$-expression (41) produced by question (40) will not match $T$-expression (39).

(41) **<<Miriam present gift> to** *whom*>

START is unable to answer such questions because it is unaware of the interactions between the syntactic and semantic properties of verbs. This limitation is a serious drawback since interactions similar to the one just

described pervade the English language and, therefore, cannot be ignored in the construction of a natural language system.

The *present* example illustrates that START needs information that allows it to deduce the relationship between alternate realizations of the arguments of verbs. In this instance, we want START to know that whenever A presents B with C, then A presents C to B. We do this by introducing rules that make explicit the relationship between alternate realizations of the arguments of verbs. We call such rules *S-rules* (where *S* stands for both *S*yntax and *S*emantics). Here is the *S*-rule that solves the problem caused by the verb *present*:

(42) *Present S*-rule

**If** <<subject **present** object1> **with** object2>
**Then** <<subject **present** object2> **to** object1>

*S*-rules are implemented as a rule-based system. Each *S*-rule is made up of two parts, an antecedent (the IF-clause) and a consequent (the THEN-clause). Each clause consists of a set of templates for *T*-expressions, where the template elements are filled by variables or constants. For example, the *Present S*-rule contains three variables, *subject, object1, object2*, which are used to represent the noun phrases in the *T*-expressions. This rule also contains three constants, **present, with**, and **to**, shown in boldface. The *Present S*-rule will apply only to *T*-expressions which involve the verb *present* and which meet the additional structural constraints.

*S*-rules operate in two modes: *forward* and *backward*. When triggered by certain conditions, *S*-rules in the forward mode allow the system to intercept *T*-expressions produced by the understanding module, transform or augment them in a way specified by the rule, and then incorporate the result into the knowledge base. For instance, if the *Present S*-rule is used in the forward mode, as soon as its antecedent matches *T*-expression (43) produced by the understanding module, it creates a new *T*-expression (44) and then adds it to the knowledge base:

(43) **<<Miriam present Gabriella> with gift>**

(44) **<<Miriam present gift> to Gabriella>**

Now question (40) can be answered since *T*-expression (41) associated with this question matches against *T*-expression (44). The generating module of START responds:

(45) Miriam presented a gift to Gabriella.

All additional facts produced by the forward *S*-rules are instantly entered in

the knowledge base. The forward mode is especially useful when the information processed by START is put into use by another computer system because in such a situation START ought to provide the interfacing system with as much data as possible.

In contrast, the backward mode is employed when the user queries the knowledge base. Often for reasons of computational efficiency, it is advantageous not to incorporate all inferred knowledge into the knowledge base immediately. $S$-rules in the backward mode trigger only when a request comes in which cannot be answered directly, initiating a search in the knowledge base to determine if the answer can be deduced from the available information. For example, the *Present* $S$-rule used in the backward mode does *not* trigger when sentence (38) is read and $T$-expression (39) is produced by START. The $S$-rule triggers only when question (40) is asked since this question cannot be answered directly.

In a more complex situation, $S$-rules are allowed to trigger each other and to ask each other for help. At any given moment hundreds of rules may be hidden in the computer's memory examining the output flow generated by START and waiting for their turn to participate in the deduction process. $S$-rules fundamentally expand the power of our language understanding system; they open a window into the intricate world of syntax-semantic interactions.

## 10. The Lexical Component

In order to understand an English sentence, the system needs to have morphological, syntactic, and semantic information about the words in the sentence. All the words that the system is aware of, along with information about their part of speech, inflection, gender, number, *etc.* are stored in the *Lexicon.* Virtually every branch of our system resorts to the Lexicon to accomplish its task. In the understanding mode, the Lexicon is used to recognize embedded clauses, to construct kernel structures, to build $T$-expressions. In the generating mode, the Lexicon is consulted when a noun or verb phrase is built, when a connective transformation is applied, when a question is answered.

The Lexicon extends the system's abilities for semantic interpretation. Consider two sentences:

(46) The man wipes the table clean

and

(47) The man eats the fish raw.

We use START's $T$-expressions to show that although these sentences seem

to have apparently identical surface syntactic structure, their semantic interpretations are quite different. The first sentence (46) has a causative interpretation:

(48) **<<man wipe table> effect <table is clean>>**

while the second sentence (47) is depictive:

(49) **<<man eat fish> when <fish is raw>>**

In order to determine the appropriate interpretation (causative or depictive) for each sentence the system needs to examine lexical entries of verbs and adjectives involved in sentences like (46) or (47). The additional information required to make this decision must be provided by the Lexicon.

Let us examine how lexical information about verbs and verb classes may be utilized by the *S*-rules. A verb denotes an action, state, or process involving one or more participants, which we refer to as the *arguments* of the verb. Some verbs may express their arguments in more than one way, sometimes with slightly different semantic interpretations. Such verbs participate in *argument alternations*. (See Atkins, Kegl, and Levin [1986], Hale and Keyser [1986], Levin [1985] for a description of various alternations.) In this section, we introduce the *property-factoring* alternation (Van Oosten [1980]). Suppose we typed the following sentence into the computer:

(50) Paul surprised the audience with his performance.

An English speaker knows that sentence (50) can be paraphrased as:

(51) Paul's performance surprised the audience.

Notice that in (50), the subject brings about the emotional reaction (*surprise*) by means of some property expressed in the *with* phrase. Sentence (51) describes the same emotional reaction as in (50) but in (51) the property and its possessor are collapsed into a single noun phrase.

Suppose that after sentence (50) is typed into the computer, we ask:

(52) Did Paul's performance surprise the audience?

While a speaker of English would know that the answer to this question is *Yes*, this reply is not obvious to START since *T*-expressions related to sentence (50) and question (52) are very different:[6]

(53) **<<Paul surprise audience> with performance>**

(54) **<performance surprise audience>**

Extending the approach taken to the example with the verb *present* in section

[6] To simplify the exposition we do not show the *T*-expression describing the relation between the property (*performance*) and its possessor (*Paul*).

9, we could formulate a simple *S*-rule that could be used to answer question (52). The *Surprise* *S*-rule (55), like the *Present* *S*-rule, makes explicit the relationship between the alternate realizations of the arguments of the verb *surprise*:

(55) *Surprise* S-rule

**If** <<subject **surprise** object1> **with** object2>
**Then** <object2 **surprise** object1>

Formulating a special purpose *S*-rule which applies only to the verb *surprise* does not seem, however, to be the best solution to the problem. *Surprise* is only one of many verbs which exhibit the property-factoring alternation. This alternation holds of a large class consisting of over one hundred verbs, among them:

(56) amuse, anger, annoy, disappoint, embarrass, frighten, please, worry ...

These verbs share a certain semantic property as well: they all denote *emotional reactions.* For this reason we identify a class of *emotional-reaction* verbs and say that the syntactic property of the verb *surprise* responsible for the alternations shown in (50) and (51) holds for all verbs that comprise the *emotional-reaction* class.[7]

Now instead of writing a number of verb-specific *S*-rules, we can write a single general *S*-rule which triggers not only on the verb *surprise*, but on any verb from the emotional-reaction class:

(57) *Property-factoring* S-rule

**If** <<subject verb object1> **with** object2>
**Then** <object2 verb object1>
**Provided** verb $\in$ *emotional-reaction* class

The revised *S*-rule contains a PROVIDED-clause which specifies the class of verbs to which the rule applies, ensuring that it applies to the emotional-reaction verbs.

When question (52) is asked, the Property-factoring *S*-rule (used in the backward mode) will trigger, since the *T*-expression

(58) **<performance surprise audience>**

produced by the question matches the THEN-part of the rule, and furthermore, the verb *surprise* belongs to the emotional-reaction class. The correct answer

[7]These verbs have been the subject of extensive study in the linguistic literature because of these and other characteristic properties that set this class apart. (See Postal [1971], Pesetsky [1987], Belletti and Rizzi [1986], Grimshaw [to appear], and many others.)

to question (52) is deduced when the appropriately instantiated IF-part of the rule is matched to $T$-expression (53) found in the knowledge base. Here is how START responds:

(59) Yes, Paul's performance surprised the audience.

This example shows how the transparent syntax of the $S$-rules coupled with the information about verb class membership provided by the Lexicon facilitates fluent and flexible dialog between the user and the language understanding system. (See Katz and Levin [1988] for additional examples.)

Our current Lexicon contains several thousand entries. However, the process of *lexical acquisition* (adding new words to the Lexicon, with all relevant information about them) is very simple. In fact, introducing a new lexical item in START amounts to little more than appending it to a list of similar words, adding a few idiosyncratic features when necessary. Acquisition of $S$-rules is equally simple. Adding a new $S$-rule to the system requires typing in a set of English sentences (such as sentences (50) and (51)) which capture a specific instance of the rule. START will analyze the sentences, query the user for additional information regarding elements of corresponding $T$-expressions (ascertaining whether they are matching variables, constants, or predicates), and then build and generalize the $S$-rule automatically. All this makes the system *transportable*, *i.e.*, easily adaptable to new domains.

## 11. Answering Questions

In this section we will concentrate on the question-answering machinery in START. Suppose the system has analyzed and indexed a text containing sentence (60). As a result, the knowledge base contains $T$-expression (61):

(60) Jessica wanted the computer to print the message.

(61) **<Jessica want <computer print message>>**$_H$

Suppose now that a user asks:

(62) What did Jessica want the computer to print?

The first step in answering this question is to reverse the effect of the *wh*-movement transformation that is used to create English *wh*-questions. In order to accomplish this, START must find the place in sentence (62) that the *wh*-word *what* came from and then insert the *wh*-word in this position:[8]

[8]This situation is very similar to the treatment of *relative clauses* discussed in section 7. In fact, the same computational machinery is used to handle these two phenomena.

(63) Jessica wanted [the computer to print *what*].

Next, the START system leads sentence (63) through the same flow of control as any other declarative sentence and produces the following $T$-expression which serves as a pattern used to query the knowledge base:

(64) **<Jessica want <computer print** *what*>>

Treating *what* as a matching variable, the system attempts to determine whether there is anything in the knowledge base that matches (64). In this case, it finds the $T$-expression

(61) **<Jessica want <computer print message>>**$_H$

The language generation system could then take this $T$-expression with its associated history and produce the English response to question (62):

(65) Jessica wanted the computer to print the message.

The matching process treats other types of English questions, including *Yes/No*-questions, *when*-questions, *where*-questions, *why*-questions, *etc.* in a similar fashion.

In this example we implicitly assumed that the tense and the aspect of question (62) were identical to the tense and aspect of sentence (60) in the text. We also assumed that sentence (60) was used in the text only once and that it was not embedded in another sentence. All these assumptions need not necessarily hold, however. For instance, one might ask:

(66) Does Jessica want the computer to print the message?

or

(67) Did the computer print the message?

A person answering these questions in the context of (60) would probably say "I don't know" since sentence (60) just states that Jessica wanted a certain action to happen at one time in the past. Sentence (60) does not imply that Jessica wants this action to happen in the present nor does it imply that this action actually happened.

To illustrate a different case, suppose that it is known from the text that

(68) The telescope is orbiting the Earth.

Now someone may ask the following questions:

(69) Has the telescope been orbiting the Earth?

(70) Can the telescope orbit the Earth?

In spite of the fact that in the original sentence (68) the auxiliaries and the form of the main verb are different from those in questions (69) and (70), a

person would most likely answer *Yes* in both cases. Somehow people know when they can or cannot answer such questions.

What about computers, then? Although clearly world knowledge plays an important role here, the text itself may often provide sufficient data to determine whether the information in the system's knowledge base implies a definitive answer to the question. Matching the embedded $T$-expressions described earlier is only a "rough" first step in answering a question; some further reasoning is required in order to determine the proper response.

One way of thinking about the problem is to divide the question-answering task into two parts: (1) finding relevant information within the knowledge base, and (2) determining the implications of this information. Finding relevant information in this case means retrieving the appropriate $T$-expression, either through matching alone or with the help of $S$-rules. The second subtask requires a more subtle analysis of the histories attached to the $T$-expressions returned by the matcher. Given a question and a matching $T$-expression, the system must use the information contained in the histories to calculate the most accurate, helpful response possible.

Let us consider an example. Suppose that the analyzed text contains sentence (71) and later sentence (72):

(71) Miriam will be reading the book.

(72) Miriam was reading the book.

The $T$-expression

(73) **<Miriam read book>**

will thus have a two-page history. Now we ask:

(74) Has Miriam been reading the book?

Recall that when START analyzes a question, it reverses the effect of the movement transformation that is used to create English questions and converts it into $T$-expression form, like any other sentence. We will call the question in the assertional form the *Q-assertion*, so that

(75) Miriam has been reading the book

is the $Q$-assertion for question (74). The task for the system is then to determine whether the known statements, (71) and (72), individually, imply $Q$-assertion (75).

The information from histories that START must use to calculate the implications falls roughly into three categories. Ties to embedded or embedding $T$-expressions (if any) give the proposition's *intrasentential context.* Tense

and aspect place the proposition in a certain *time frame.* Modals, including negation as a special instance, show the *mood* of the proposition.

Accordingly, START employs an ordered set of filters, one for each type of information. A kernel sentence may receive one of three labels, YES, NO, or UNKNOWN, as it passes through the filter set. The UNKNOWN label is further broken down to show which filter assigned it, so that this information can be used in responding to the question in the most helpful way.

The embedding filter must be applied first. Thus, the sentence "Miriam read the book" which is embedded in (76) is labeled UNKNOWN*(embedding)*:

(76) Gabriella thinks that Miriam read the book.

A sentence that is not labeled UNKNOWN by the embedding filter is passed to the tense-aspect filter. This filter determines whether the time interval referred to by the kernel sentence overlaps with that given by the question. If they do not overlap, the sentence is labeled UNKNOWN*(time)*. For example, in attempting to answer the question "Has Miriam been reading the book," this filter would label the sentence "Miriam will be reading the book" UNKNOWN*(time)*. The sentence "Miriam was reading the book" would pass through the time-aspect filter, to be considered by the modal filter. The modal filter compares the modals (including negation) used in the kernel sentence and in the question. Each combination results in one of the three labels YES, NO, or UNKNOWN*(modals)*. (In Gaulding and Katz [1988] the operation of the tense-aspect and modal filters is discussed in more detail.)

If a sentence has internal embeddings, the reasoning process is repeated for each $T$-expression involved, producing a decision at each level of embedding. These decisions are then arbitrated in the following manner. If at any point the time frame referred to is inappropriate, or the modal combination is inconclusive, the entire sentence is labeled UNKNOWN*(time)* or UNKNOWN*(modals)*. Otherwise the "logical combination" of affirmative and negative responses is calculated, and the overall sentence is labeled with the result. For example, suppose the system had indexed the sentence

(77) Gabriella wishes that Miriam had read the book

and were asked

(78) Does Gabriella wish that Miriam will be reading the book?

The outer clause would receive the label YES but the inner clause would not pass the tense-aspect filter, and as a result the entire sentence would be relabeled UNKNOWN*(time)*. Although this solution gives correct answers for many internal-embedding situations, it seems that the general case requires more complex reasoning (see Gaulding and Katz [1988] for possible approaches).

Once a final label has been calculated for each sentence contained in the history, START arbitrates the resulting combination of YES's, NO's, or UNKNOWN's and decides upon the final response. Because the overall goal is to be as informative as possible, the response does not consist merely of a "yes", "no", or "unknown". All the sentences corresponding to the matching $T$-expression are generated for the user, and the explanations are given when necessary.

The following interaction illustrates that the system's final responses, made on the basis of this analysis, echo people's judgments in answering such questions. Note that inputs to START are given to the prompt "==⟩".

==⟩ ED EXPECTS TO LIKE ROCK MUSIC, BUT HE DOES NOT LIKE IT NOW.
==⟩ MOM HOPES ED WILL NOT LIKE IT.

==⟩ WILL ED LIKE ROCK MUSIC?

I don't know. However, I do have some information:

ED EXPECTS TO LIKE ROCK MUSIC.

MOM HOPES THAT ED WILL NOT LIKE ROCK MUSIC.
(The proposition 'ED LIKES ROCK MUSIC' was embedded within these sentences.)

ED DOESN'T LIKE ROCK MUSIC NOW.
(This sentence did not refer to the time interval about which you asked.)

## 12. A Dialogue with START

The following is a slightly shortened and modified passage from the *Testing Reading Skills* chapter of the book *How to Prepare for the Graduate Record Examination* (Brownstein and Weiner [1985]), as entered into the computer:

> *The establishment of the Third Reich influenced events in American history by starting a chain of events which culminated in war. The complete destruction of democracy, the persecution of Jews, the cruelty of the Nazis, and the plans of Germany for the conquest of the world caused great indignation in the United States. While speaking out against Hitler's atrocities, the American people generally favored isolationist policies and neutrality. The Neutrality Act of 1935 prohibited trade with any belligerent nation. In 1937 the President was empowered to declare an arms embargo in wars between nations at his discretion.*

*American opinion changed after President Roosevelt's speech at Chicago in which he severely criticized Hitler's policies. Germany's seizure of Austria and the Munich Pact also aroused the American people. The conquest of Czechoslovakia in 1939 was another rude awakening to the menace of the Third Reich. The United States attempted to maintain neutrality in spite of sympathy for the European democracies. The Neutrality Act of 1939 repealed the arms embargo and permitted exports of arms to belligerent nations. In 1940 a strong national defense program was begun and a draft act was passed to strengthen the military services. In 1941 Japan launched the unprovoked attack on the United States at Pearl Harbor. Immediately thereafter, Germany declared war on the United States.*

The understanding module of START analyzes this text and indexes the knowledge contained within it into a knowledge base. The user is then able to interact with the system in English. Shown below is an actual dialogue with START. The user asks questions and the computer answers them. Note that the English sentences generated by START may differ from the original text. Any stylistic deficiencies are the responsibility of START and not of this author.

The astute reader will observe that in addition to syntactic analysis, indexing, matching, retrieving the information, and generating English sentences, the START system was quietly performing additional tasks, such as: resolving lexical ambiguities (*e.g.*, the word *attack* was used as a verb and as a noun); working with synonyms (*begin-start, permit-allow, etc.*); employing $S$-rules (*e.g.*, in answering the question about Japan attacking the USA); and providing further information (in parentheses) when the correct answer could not be found due to embedding, or disagreements in time intervals or modality.

## Querying the GRE Knowledge Base

What did Germany's plans cause?

> Germany's plans for the world's conquest caused great indignation in the United States.

Did the American people favor neutrality?

> Yes—The American people generally favored neutrality while speaking out against Hitler's atrocities.

What did the Neutrality Acts do?

1935's Neutrality Act prohibited the trade with any belligerent nation.
1939's Neutrality Act repealed the arms embargo.
1939's Neutrality Act permitted the arms' exports to the belligerent nations.

Who declared the Arms embargo?

*I don't know. However, I do have some information:*

The President was empowered to declare the arms embargo in the wars between the nations at his discretion in 1937.

(*The proposition* 'The President declares the arms embargo' *was embedded within this sentence.*)

When did the American opinion change?

The American opinion changed after President Roosevelt's speech at Chicago in which President Roosevelt severely criticized Hitler's policies.

Who criticized Hitler's policies?

President Roosevelt severely criticized Hitler's policies in his speech.

Who maintained neutrality?

*I don't know. However, I do have some information:*

The United States attempted to maintain neutrality in spite of the sympathy for the European democracies.

(*The proposition* 'The United States maintains neutrality' *was embedded within this sentence.*)

Were exports of arms allowed?

Yes—1939's Neutrality Act permitted the arms' exports to the belligerent nations.

To which nations did the Neutrality Act permit exports?

1939's Neutrality Act permitted the arms' exports to the belligerent nations.

Why was the draft act passed?

The draft act was passed in order to strengthen the military services.

Will a national defense program be started?

*I don't know. However, I do have some information:*

The strong national defense program was begun in 1940.

*(This sentence did not refer to the time interval about which you asked.)*

Did Japan attack the USA?

Yes—Japan launched the unprovoked attack on the United States at Pearl Harbor in 1941.

## 13. Spacecraft Sequencing

In this section we demonstrate the application of the START system in a real-world situation: interplanetary exploration. The scientists and engineers who plan to perform experiments aboard an interplanetary spacecraft compete for a limited amount of time and resources. Their requests are coordinated and integrated into a sequence of activities for the spacecraft through a *spacecraft sequencing* process. This process programs involves designing, scheduling and programming the onboard activities, as well as controlling its functions. It is a complex, tedious, and time consuming process which is carried out by a team of experts called the *Sequence Team.* In order to perform this task, the Sequence Team uses a set of computer programs, the *Mission Sequence Software (MSS).* These programs do everything from simulating the geometry of the encounter to detailed constraint checking of the proposed sequence and actual command simulation and generation. Unfortunately, this software is very difficult to use. Producing MSS which is more "user friendly" could reduce the cost of a mission dramatically.

In Katz and Brooks [1987] we have identified three possible roles that the START system could play in improving the MSS performance. First, START could function as a translator from English conceptual descriptions of activities into inputs for the MSS, providing the long-needed link between early design work and integration of the sequence. Secondly, START could act as an interface between a user and the various components of the MSS during integration, allowing, but not requiring, the user to operate the MSS by means of English commands instead of the cryptic operands used presently. And finally, START could be employed as a query tool. The researchers could ask questions in English about the spacecraft or the state of its submodules and the system would analyze the query, retrieve the relevant information from the knowledge base and formulate its response also in English.

The Mars Observer Mission plans to use small modular sequence components called Sequence Segments. These segments, which are based on the geography of Mars, will be used during mission operations to build the sequences of activities to be executed onboard. Shown below is a sample of the types of observations which will be specified in a typical segment. This document is automatically transformed by the START system into a knowledge base which incorporates the information found in the text. Following that, we show how the user obtains the information about the events which are taking place in the sequence by querying the knowledge base in English.

MOC, VIMS, TES, and PMIRR are all scientific instruments on the Mars Observer spacecraft (MO). IR is an abbreviation for Infra-red. All other capitalized words (ALBA PATERA, ASCRAEUS MONS, TANTALUS FOSSAE, etc.) are names of targets on the planet's surface.

## Mars Observer Sequence Segment
(as entered in the computer)

00:04:20 ASCRAEUS MONS is at Nadir.
00:04:35 MOC takes 5 pictures of ASCRAEUS MONS.
00:04:35 TES performs experiment number 16 on ASCRAEUS MONS.
00:04:35 PMIRR performs IR study of ASCRAEUS MONS.
00:04:35 VIMS takes 1 picture of ASCRAEUS MONS.
00:08:20 Entering CERAUNIUS FOSSAE region from south side.
00:10:25 PMIRR performs IR study of CERAUNIUS FOSSAE.
00:12:30 Exiting CERAUNIUS FOSSAE region from north side.
00:14:30 Entering TANTALUS FOSSAE region from south side.
00:14:35 MOC takes 4 pictures of TANTALUS FOSSAE.
00:16:40 +40-deg latitude crossing pulse occurs northbound.
00:17:00 Entering ALBA PATERA region from south-east side.
00:17:00 Exiting TANTALUS FOSSAE region from north side.
00:18:40 Entering ALBA FOSSAE region from south side.
00:18:40 Exiting ALBA PATERA region from north-east side.
00:18:45 Take 5 pictures of ALBA FOSSAE with MOC.
00:20:40 Entering VASTITAS BOREALIS region from south side.
00:20:40 Exiting ALBA FOSSAE region from north side.
00:22:55 Take 2 pictures of VASTITAS BOREALIS with MOC.
00:22:55 VIMS takes 1 picture of VASTITAS BOREALIS region.
00:27:05 +65-deg latitude crossing pulse occurs northbound.
00:29:10 MOC takes 3 pictures of the north-polar region.
00:29:10 Take 1 picture of north-polar region using VIMS.
00:31:15 PMIRR begins continuous IR study of north-polar region.
00:33:20 TES begins study of north-polar region.
00:34:10 Entering north-polar region.
00:37:30 +90-deg latitude crossing pulse occurs.
00:37:30 MOC takes 6 pictures of the North Pole.
00:37:30 PMIRR performs internal experiment # 21 on North Pole.
00:37:30 TES does internal experiment #11 on North Pole.
00:37:30 VIMS executes internal experiment number 1 on North Pole.
00:41:40 TES ends study of north-polar region.
00:43:45 Exiting VASTITAS BOREALIS region from north side.
00:43:45 Exiting north-polar region.
00:43:45 PMIRR ends study of north-polar region.
00:47:55 +65-deg latitude crossing pulse occurs southbound.
00:52:05 Entering UTOPIA PLANITIA from north side.
00:54:10 Use MOC to take 2 pictures of VASTITAS BOREALIS.

00:54:10 VIMS takes 1 picture of VASTITAS BOREALIS.
00:58:20 +40-deg latitude crossing pulse occurs southbound.
01:00:25 Entering NILOSYRTIS MENSAE from north side.
01:00:25 Exiting UTOPIA PLANITIA from south side.
01:01:15 Take 1 picture of NILOSYRTIS MENSAE using MOC.
01:01:15 Take 1 picture of NILOSYRTIS MENSAE with VIMS.
01:04:35 Entering NILI FOSSAE region from north side.
01:04:35 Exiting NILOSYRTIS MENSAE from south side.
01:06:40 Entering SYRTIS MAJOR PLANITIA region from north side.
01:06:40 Exiting NILI FOSSAE region from south side.
01:06:40 PMIRR begins IR study of SYRTIS MAJOR PLANITIA.
01:06:40 VIMS takes 1 picture of NILI FOSSAE.

## Querying the Mars Observer Knowledge Base
(actual dialog with START)

How many pictures of ASCRAEUS MONS does MOC take?

At 00:04:35, MOC takes 5 pictures of ASCRAEUS MONS.

When does PMIRR begin continuous IR study of north-polar region?

At 00:31:15, PMIRR begins the north-polar region's continuous IR study.

When does VIMS take pictures of VASTITAS BOREALIS region?

At 00:54:10, VIMS takes 1 picture of VASTITAS BOREALIS.
At 00:22:55, VIMS takes 1 picture of VASTITAS BOREALIS.

Which instruments take pictures of ASCRAEUS MONS?

At 00:04:35, MOC takes 5 pictures of ASCRAEUS MONS.
At 00:04:35, VIMS takes 1 picture of ASCRAEUS MONS.

When does PMIRR execute the experiment #21 on North Pole?

At 00:37:30, PMIRR performs experiment21 on North Pole.

What does PMIRR do between 00:10:00 and 00:50:00?

At 00:10:25, PMIRR performs CERAUNIUS FOSSAE'S IR study.
At 00:31:15, PMIRR begins the north-polar region's continuous IR study.
At 00:37:30, PMIRR performs experiment21.
At 00:43:45, PMIRR ends the north-polar region's continuous IR study.

When do latitude crossing pulses occur?

At 00:58:20, the +40-deg latitude crossing pulse occurs southbound.
At 00:16:40, the +40-deg latitude crossing pulse occurs northbound.
At 00:47:55, the +65-deg latitude crossing pulse occurs southbound.
At 00:27:05, the +65-deg latitude crossing pulse occurs northbound.
At 00:37:30, the +90-deg latitude crossing pulse occurs.

Does Mars Observer enter the NILI FOSSAE region?

Yes—At 01:04:35, spacecraft enters NILI FOSSAE from the north side.

From which side does MO exit the regions?

At 00:12:30, spacecraft exits CERAUNIUS FOSSAE from the north side.
At 00:17:00, spacecraft exits TANTALUS FOSSAE from the north side.
At 00:18:40, spacecraft exits ALBA PATERA from the north-east side.
At 00:20:40, spacecraft exits ALBA FOSSAE from the north side.
At 00:43:45, spacecraft exits VASTITAS BOREALIS from the north side.
At 01:00:25, spacecraft exits UTOPIA PLANITIA from the south side.
At 01:04:35, spacecraft exits NILOSYRTIS MENSAE from the south side.
At 01:06:40, spacecraft exits NILI FOSSAE from the south side.

Which targets does VIMS look at?

At 01:06:40, VIMS takes 1 picture of NILI FOSSAE.
At 01:01:15, VIMS takes 1 picture of NILOSYRTIS MENSAE.
At 00:29:10, VIMS takes 1 picture of the north-polar region.
At 00:54:10, VIMS takes 1 picture of VASTITAS BOREALIS.
At 00:22:55, VIMS takes 1 picture of VASTITAS BOREALIS.
At 00:04:35, VIMS takes 1 picture of ASCRAEUS MONS.

How many pictures does MOC take after 00:20:00?

At 00:22:55, MOC takes 2 pictures of VASTITAS BOREALIS.
At 00:29:10, MOC takes 3 pictures of the north-polar region.
At 00:37:30, MOC takes 6 pictures of North Pole.
At 00:54:10, MOC takes 2 pictures of VASTITAS BOREALIS.
At 01:01:15, MOC takes 1 picture of NILOSYRTIS MENSAE.

Total number of pictures is 14.

## Acknowledgments

From the beginning Patrick H. Winston has been actively involved in the process of designing and developing START. He has also been a very active user of the system.

A number of students have participated in the project, in particular, David A. Chanen, Robert Frank, Jill Gaulding, and Jeff Palmucci contributed significantly to various parts of the system.

I am grateful to Mikhail Katz and Beth Levin for their time and numerous helpful suggestions concerning this paper. In addition, useful comments were provided by Richard Doyle, Robert Frank, Jill Gaulding, Michael Kashket, David Kirsh, Mitch Marcus, and Thomas Marill.

Robert N. Brooks from JPL developed the data for the Mars Observer mission which is used in section 13.

This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the Advanced Research Projects Agency under Office of Naval Research contract N0014-85-K-0124.

## References

1. A. Akmajian and F. Heny, *An Introduction to the Principles of Transformational Syntax*, MIT Press, Cambridge, MA, 1975.
2. B.T. Atkins, J. Kegl, and B. Levin, "Explicit and Implicit Information in Dictionaries," Lexicon Project Working Papers 12, Center for Cognitive Science, MIT, Cambridge, MA, 1986.
3. A. Belletti and L. Rizzi, "Psych-Verbs and Th-Theory," Lexicon Project Working Papers 13, Center for Cognitive Science, MIT, Cambridge, MA, 1986. To appear in *Natural Language and Linguistic Theory.*
4. S.C. Brownstein and M. Weiner, *How to Prepare for the Graduate Record Examination*, Barron's Educational Series, Inc., Woodbury, N.Y., 1985.
5. N. Chomsky, *A Theory of Syntactic Structures*, Moulton & Co., 1957.
6. R.J. Doyle, "Hypothesizing and Refining Causal Models," MIT Artificial Intelligence Laboratory Memo No. 811, December 1984.
7. J. Gaulding and B. Katz, "Using Word-Knowledge Reasoning for Question Answering," in *The Society of Text*, edited by E. Barrett, MIT Press, Cambridge, MA, to appear, 1989.
8. J. Grimshaw, *Argument Structure*, MIT Press, Cambridge, MA, to appear.
9. K.L. Hale and S.J. Keyser, "Some Transitivity Alternations in English," Lexicon Project Working Papers 7, Center for Cognitive Science, MIT, Cambridge,

MA, 1986.

10. B. Katz, "A Three-step Procedure for Language Generation," MIT Artificial Intelligence Laboratory Memo No. 599, December 1980.
11. B. Katz and R. Brooks, "Understanding Natural Language for Spacecraft Sequencing," *JBIS*, Vol. 40, No. 10, 1987.
12. B. Katz and B. Levin, "Exploiting Lexical Regularities in Designing Natural Language Systems," *Proceedings of the 12th International Conference on Computational Linguistics, COLING '88*, Budapest, 1988. (A version of this paper also appears as Lexicon Project Working Papers 22, MIT Center for Cognitive Science, and as MIT Artificial Intelligence Laboratory Memo No. 1041, 1988.)
13. B. Katz and P.H. Winston, "A Two-way Natural Language Interface," in *Integrated Interactive Computing Systems*, edited by P. Degano and E. Sandewall, North-Holland, Amsterdam, 1982.
14. B. Levin, "Introduction," in *Lexical Semantics in Review*, edited by B. Levin, Lexicon Project Working Papers 1, Center for Cognitive Science, MIT, Cambridge, MA, 1985.
15. B. Levin, "Approaches to Lexical Semantic Representation," in *Automating the Lexicon*, edited by D. Walker, A. Zampolli, and N. Calzolari, MIT Press, to appear, 1989.
16. W.I. McLaughlin, "Automated Sequencing," *Spaceflight*, Vol. 29, No. 1, January 1987.
17. W.I. McLaughlin, "Computers and Language," *Spaceflight*, Vol. 30, No. 8, August 1988.
18. D. Pesetsky, "Binding Problems with Experiencer Verbs," *Linguistic Inquiry* 18, 126-140, 1987.
19. P. Postal, *Cross-Over Phenomena*, Holt, Rinehart, and Winston, NY., 1971.
20. R. Quirk and S. Greenbaum, *A Concise Grammar of Contemporary English*, Harcourt Brace Jovanovich, New York, 1973.
21. J. Van Oosten, "Subjects, Topics and Agents: Evidence from Property-factoring," *Proceedings of the Berkeley Linguistics Society* 6, Berkeley, CA, 1980.
22. P.H. Winston, *Artificial Intelligence*, Addison-Wesley, Reading MA, 1984.
23. P.H. Winston, "Learning New Principles from Precedents and Exercises," *Artificial Intelligence*, vol. 19, no. 3, 1982.
24. P.H. Winston, T.O. Binford, B. Katz, and M.R. Lowry, "Learning Physical Descriptions from Functional Definitions, Examples, and Precedents," *National Conference on Artificial Intelligence*, Washington, D.C., 1983.
25. W.A. Woods, "What's in a Link: Foundations for Semantic Networks," in *Readings in Knowledge Representation*, edited by R.J. Brachman and H.J. Levesque, Morgan Kaufmann Publishers, Los Altos, 1985. (Originally published in *Representation and Understanding: Studies in Cognitive Science*, Academic Press, 1975.)