VOX--An Extensible Natural Language Processor

Amnon Meyers

Artificial Intelligence Project
Technical Report #85-02
April, 1985

(Revised August, 1985)

ABSTRACT

VOX is a Natural Language Processor whose knowledge can be extended by interaction with a user.

VOX consists of a text analyzer and an extensibility system that share a knowledge base. The extensibility system lets the user add vocabulary, concepts, phrases, events, and scenarios to the knowledge base. The analyzer uses information obtained in this way to understand previously unhandled text.

The underlying knowledge representation of VOX, called Conceptual Grammar, has been developed to meet the severe requirements of extensibility. Conceptual Grammar uniformly represents syntactic and semantic information, and permits modular addition of knowledge.

This work is supported by the Naval Ocean Systems Center, under NOSC Grant N66001-83-C-0255.

1. INTRODUCTION

The ability to learn is one of the most important characteristics of intelligent systems. To approach such an ability, we must first build systems that can automatically accept new knowledge. By continually enhancing the extensibility capabilities of such systems, we can begin to address the problems of general learning.

Critical to extensibility is the underlying knowledge representation. The more powerful and flexible the knowledge representation, the more easily extensibility capabilities can be built and improved.

VOX (Vocabulary Extension System) is a Natural Language Processing system that emphasizes automatic extensibility. In VOX, extensibility capabilities are developed hand-in-hand with the knowledge representation. The knowledge representation, called Conceptual Grammar [*Meyers*], supports a bottom-up study of language, by representing both very general and very specific knowledge. As generalizations about language are discovered, they are simply incorporated into the representation.

Currently, VOX allows automatic addition of vocabulary and action-oriented events and scenarios. The user may build knowledge hierarchies of scenarios, events, nouns, verbs, adjectives, and other parts of speech, as well as specifying a variety of semantic and syntactic information about these objects. The VOX analyzer makes use of the information obtained in extensibility sessions to analyze novel text.

1.1 EXAMPLES

We will illustrate how VOX works by adding the sequence of events for a simple Naval 'attack' scenario:

ship searches for ship.
ship sights ship.
ship approaches ship.
ship attacks ship.
ship damages ship.

We will add the words, then the individual events, and then the entire scenario to the system. Finally, we will show the kinds of text the system can analyze using this knowledge.

MACRO NOUN Example: "ship"

Enter singular form of noun: **ship**
Enter plural form of noun: **ships**

Enter synonym or more general concept: **platform**

Macro noun is an extensibility capability for adding nouns. The words 'ship', 'ships', as well as the more abstract concepts ship(noun) and ship(np) are added to the knowledge base by macro noun. A phrase like "the 3 green ships" will be found to be equivalent to ship(np), for example. By specifying 'platform', the user places 'ship' into a conceptual hierarchy of nouns already containing 'platform'. ('Platform' is a Navy word for anything that a missile can be fired from. Thus, a base, a submarine, and an aircraft are all platforms.)

MACRO VERB Example: Add "search"

Enter the various forms of the verb.

Present.	(verb):	**search**
Third person.	(verb + s):	**searches**
Progressive.	(verb + ing):	**searching**
Past.	(verb + ed):	**searched**
Participle.	(verb + en):	**searched**

Verb kind (regular or prepositional): **prepositional**

Enter synonym or more general concept: __

Macro verb is similar to macro noun. In addition to concepts for the specific words, abstract concepts like search(vp), search(event), and search(frame) are added to the system. (search(frame) represents the generic search scenario.) By not specifying a synonymous concept for search, we place it at the top of a conceptual hierarchy. The macro will automatically have 'search' suggest a syntactic verb-phrase concept.

Assume that all the word-level items in the simple attack scenario have been added using macro noun, macro verb, and macros for other parts of speech. Next, we add an event:

MACRO EVENT Example: "ship search location for ship"

Enter an event:

ship search location for ship
1 2 3 4 5

Semantic information

Enter position of the following in the event-phrase:

actor = **1**
act = **2**
object = **5**
instrument = __
location = **3**
time = __

Enter a known concept that the event suggests: **search**

Syntax information

Enter position of subject: **1**
Enter voice of act (active or passive): **active**

Optionality information

Enter starting points of event: **1**
Enter end points of event: **5**

Enter skipping points for event.

Element 1 can skip to: __
Element 2 can skip to: **4**
Element 3 can skip to: __

Entry for new event = ship-search

Macro event lets the user add standard events to the knowledge base. These events are templates, and will match much more than the literal words "ship search location for ship". Macro event uses the abstract concepts ship(np), search(vp), rather than the word-level concepts. The event added is treated not just as a semantic restriction, but as a concept in its own right. The concept <ship-search-location-for-ship> is stored in the knowledge base under the entry 'ship-search'. We can use concepts such as this to add new scenarios, as will be shown below. This event concept was added to a hierarchy of events by suggesting 'search'. Macro event had this specific event suggest the generic 'search' event.

The user specifies the semantic case (actor, act, location, etc.) of each element in the event. The user specifies that the phrase starts with element 1 and ends with element 5. The syntactic component of the Conceptual Grammar handles incomplete forms such as "ship searched the area", so the user need not specify that element 3 is a possible end of the event phrase. The user specifies that element 3

can be skipped over; that is, "The ship searched for the submarine", omitting a location element, is correct English. The user specifies this because it varies on a case-by-case basis. For example, in the sentence "Ship conducted attack on submarine", "attack" could not be omitted.

Assume that all events of the simple attack scenario have been entered using macro event. Next, we invoke macro frame to add the entire scenario.

MACRO FRAME Example: An attack scenario

Enter the events of the frame:

ship-search ship-sight ship-approach ship-attack ship-damage
1 2 3 4 5

Briefly describe the frame: **ship attacks ship**

Semantic information

Enter the main event: **4**
Enter events needed in a complete text: **2 4 5**
Enter a known concept that the frame suggests: **attack**

Enter number of actor roles: **2**

Enter a word for actor 1: **ship**
Actor 1 is subject in which events? **1 2 3 4 5**
Actor 1 is object in which events? __

Enter a word for actor 2: **ship**
Actor 2 is subject in which events? __
Actor 2 is object in which events? **1 2 3 4 5**

Optionality information

Enter starting points for frame: **1 2 3 4 5**
Enter end points for frame: **4 5**

Enter skipping points for frame.

Element 1 can skip to: **3 4 5**
Element 2 can skip to: **4 5**
Element 3 can skip to: **5**

Entry for frame = ship-attack

Macro frame is similar in many ways to macro event. In particular, this specific

scenario is a concept unto itself, and is placed under the entry 'ship-attack' [Note: the entry 'ship-attack' holds both events and scenarios.] We entered it into a hierarchy of scenarios by having it suggest the generic attack (frame) concept.

Optionality information: The user specifies that the description of the scenario could start with any of the events in it. For example, a complete text might read "damaged sub". On the other hand, we are only certain that an attack scenario is being described if the attack or damage events are present. The skipping information indicates that any of the events in this scenario may be omitted in text.

Having entered this scenario into the system, we can make use of it to understand texts that deal with a 'ship attacking ship' scenario. Here is an example of the kind of text VOX analyzes using the scenario we have just entered. Note that the Constellation aircraft carrier is assumed by VOX to be sending a message with the given text.

VOX Text Analysis Example

(Constellation is message sender.)

Type message:

at 1235T had searched area. damaged sub.

INTERPRETATION 1 OF 1.

MESSAGE FEATURES

ERROR: missing event = attack
ERROR: missing event = sight
ERROR: missing object = sub
ERROR: missing actor = constellation
ERROR: missing actor = constellation

REWORDED MESSAGE

constellation had searched area for sub at 1235 t.
constellation damaged sub.

Frame = ship attacks ship

When analyzing text, VOX produces an internal representation of the concepts underlying the text, similar to other case-based or frame-based conceptual representations. The representation embodies VOX's understanding of the input text, and is used for checking and correcting semantic and syntactic problems in the text.

The VOX output shown above is geared to the task of sending Navy messages:

The VOX analyzer first lists the errors that it found in the given text. In particular, VOX verbalizes its inferences about events that were not mentioned in the text. Though an 'approach' event is reasonable to infer as well, it is not necessary for a complete description of the scenario, so its absence is not mentioned by VOX. VOX also makes inferences as to who the missing actors were. Other semantic and syntactic errors are reported similarly.

After cataloguing the errors, VOX produces a reworded text that is faithful to the original text, with errors corrected. Such a rewording is more useful to the Navy message sender than a paraphrase containing all the inferences obtained from the internal representation of the message. VOX can interact with the user to correct the message further, using the errors that were found by the system. We omit such an example, for lack of space. Note that VOX's understanding of the message is script-based or scenario-based, similar to the way scripts are used by Schank. The way this text was analyzed is a direct result of the way in which the scenario was entered using macro frame.

1.2. DISCUSSION

Semantics in VOX is expressed by function, rather than by definition. We do not define the words we enter with the macros for parts of speech. The meaning of words is determined by the conceptual hierarchies in which they are placed and by the events and phrases in which they are used. The semantics of events and scenarios is likewise determined by the conceptual hierarchies in which they are placed, and by the scenarios that use them. The more such knowledge the system has about an object, the better it is understood.

The syntactic coverage of the macros is minimal. We have tried to gain a deep understanding of the simplest syntactic cases, with the idea of building a firm foundation from which to add new syntactic capabilities. The section on Conceptual Grammar will illustrate the intensive, rather than extensive, nature of our work so far.

2. VOX SYSTEM

2.1. BACKGROUND

VOX is being applied to the analysis and correction of Navy tactical messages, like its predecessor, NOMAD [*Granger*]. These messages are characterized by terse and ungrammatical English and heavy use of abbreviations and jargon.

NOMAD was developed along the lines of CA [*Birnbaum & Selfridge*], that is, *word-expert* systems, with a routine for every vocabulary word. Our experience with NOMAD led us to conclude that word-expert systems are difficult to extend, for several reasons:

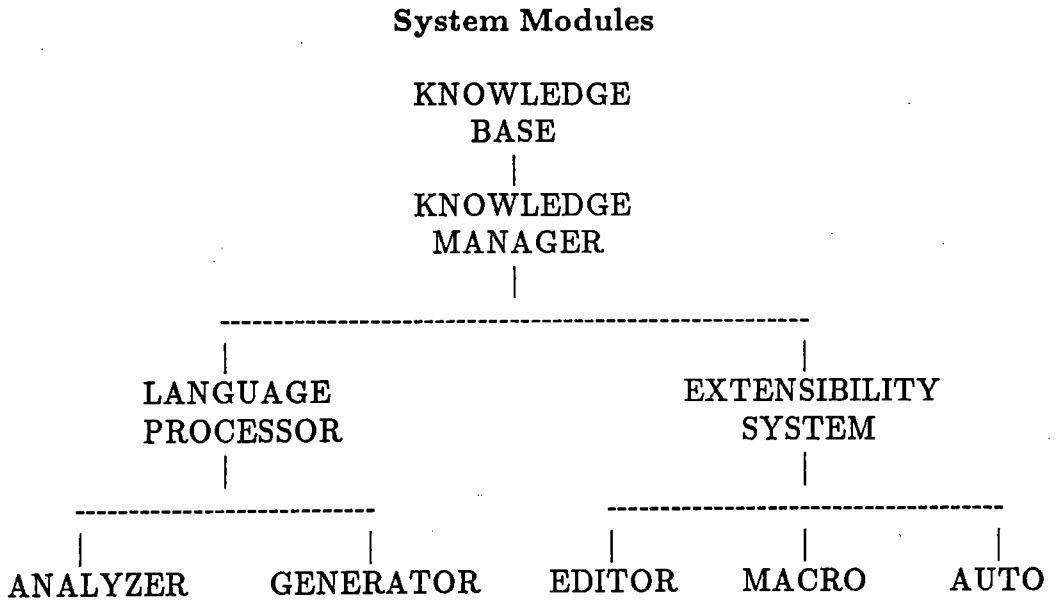
a) Every new vocabulary word requires a new routine to be coded.
b) Standardization of the code in routines is difficult.
c) Adding a word often requires reprogramming existing routines.
d) Automation of coding is difficult, because of (b) and (c).
e) Syntactic knowledge is decentralized.
f) Handling of English phrases and idioms, especially discontinuous phrases, is cumbersome for word experts.

We felt that phrase-based systems, along the lines of PHRAN [*Wilensky, 1980*], would be more amenable to automatic extensibility, by virtue of their uniform knowledge base. The ability to represent very general and very specific knowledge in phrases was also appealing, as was the clean separation of knowledge from processing mechanisms.

VOX and the underlying Conceptual Grammar representation have grown out of our attempt to organize phrases more systematically and to incorporate standard syntax theory into the phrasal knowledge scheme. In Conceptual Grammar, phrases are regarded as *concepts* in their own right, rather than being associated with Conceptual Dependency structures.

It is interesting that Conceptual Grammar and KODIAK [*Wilensky, 1984*], both of which grew out of phrase-based systems, support the notion of "proliferation of concepts", though they differ greatly in other respects: Conceptual Grammar, unlike other knowledge representations, is concerned with *surface language constructs* and the mapping of these into the underlying concepts, in addition to providing a framework for the representation of concepts.

2.2. DISCUSSION



The VOX NLP system is dedicated to the complete analysis of English text. Ill-

formed input is handled: Syntactic and semantic errors of many kinds are detected. The system produces a reworded text that is faithful to the original text. Syntactic and semantic ambiguity are handled as well.

VOX consists of three systems: a language processor devoted to language analysis, an extensibility system, and a knowledge base. Each of these will be discussed in the following sections.

2.3. EXTENSIBILITY SYSTEM

The extensibility system has three components: a primitive data structure editor, a macro extensibility system, and an auto- extensibility system.

The editor provides an organized framework for manipulation of the data structures of the knowledge base. It consists of primitives to add, remove, and examine objects of the knowledge base. A slightly more structured set of primitives assures that objects of the knowledge base are manipulated in a consistent or legal manner.

The macro extensibility system allows the user to manipulate knowledge in a more structured way, allowing users who are largely ignorant of the details of the knowledge base to use the extensibility system. The user can add vocabulary (nouns, verbs, adjectives, and so on), events, and scenarios as exemplified at the start of this paper. Each macro facility invokes the editor to actually manipulate the knowledge base.

Macros for other kinds of phrases are also being constructed. For example, *macro pv* lets a user add phrases like "search for" and "fire at" to the knowledge base, thus informing the system that these are standard English idioms.

2.3.1. AUTO-EXTENSIBILITY SYSTEM

Because all of VOX's knowledge is added by interaction with the extensibility system, we have found it profitable to store all of the English and domain knowledge in the form of command files that invoke the extensibility system. These files simply list the commands the user would type if he were interacting with the extensibility system. We refer to the entire set of command files as the *auto-extensibility system.*

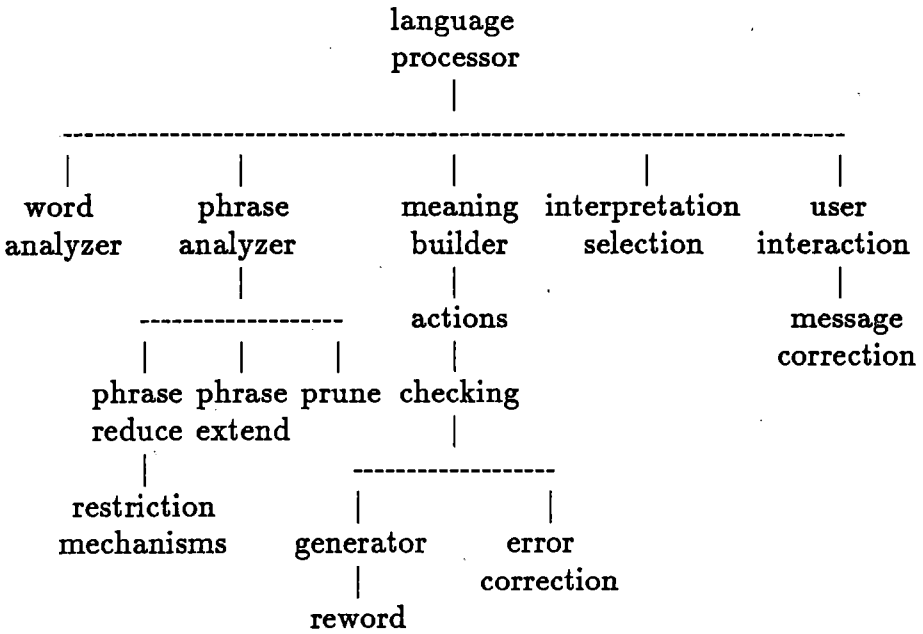
The auto-extensibility system provides many advantages. It

a) is a compressed form of the knowledge.
b) is a readable form of the knowledge.
c) is an easily corrected form of the knowledge.
d) documents the order of knowledge addition.
e) organizes the domains of knowledge.

f) facilitates addition of large amounts of knowledge by simple editing changes. One can make many copies of the interaction with macro noun, for example, and edit them for new nouns to be added to the system.
g) facilitates correction of knowledge.
h) provides a core of knowledge that can be restored at any time. Testing can be done without harming the system.
i) serves as a backup for the knowledge base.
j) allows several tutors to update the same version of the knowledge base. Tutors work by editing command files, and one worker uses these to update the knowledge base.

Since the macro extensibility system is interactive, an analogous set of noninteractive macro commands is used by the auto-extensibility system. The commands are named *auto noun*, *auto verb*, and so on. The auto commands are guaranteed to provide a fixed dialogue, while the interaction with macro commands may vary from session to session.

2.4. LANGUAGE PROCESSOR



The language-processing component of VOX is dedicated to language analysis, with emphasis on ill-formed text. Generation consists mainly of rewording incomplete or erroneous text.

The input text is read left to right. First, a primitive word analyzer groups characters into lexical units. If a word is unknown to VOX, the user may invoke the extensibility system to add the word before the analysis continues. The word analyzer submits lexemes to the phrase analyzer, one at a time.

The phrase analyzer, or rule-based analyzer, uses a parallel parsing algorithm: it

collects all the accessible suggestions of every concept, starting with the concepts suggested by the current lexeme. Each concept is then checked to see if it extends phrases collected while processing the previous text. The analyzer uses restriction mechanisms [*such as the ones discussed in Section 3.2*] to find specific knowledge corresponding to the input text. Pruning mechanisms are also used to limit combinatorial explosion. For example, once the end of a sentence has been verified, phrases that have not been used to match the entire text so far are pruned from the system.

While the analysis is essentially a brute-force parallel algorithm, Conceptual Grammar guides the analysis along meaningful and noncombinatorially explosive paths. For example, if a text is found to correspond to a specific scenario, this interpretation is preferred to one which merely recognizes the text as a sequence of unrelated events. Another example is the collection of lists of concepts: a new list cannot be started if a list of the same type can be extended.

After the rule-based analysis, an initial selection process eliminates all but the most meaningful parse trees constructed for the input text.

A meaning construction process then operates on the selected parse trees, building an internal conceptual representation for each interpretation. The conceptual representation is used for detecting, documenting, and correcting errors in the text. Rewording of the text is also performed for each interpretation, in this phase.

The conceptual representation consists of data structures for frames, events, actors, acts, locations, words, and so on. The data structures each contain slots for the various cases. For example, the event data structure has slots for the agentive actor, affected actor, action, location, time, instrument, and other semantic cases. A full discussion of the internal representation used in VOX is beyond the scope of this paper. (The concept tree built during the analysis of text is an important component of the internal representation, because it contains semantic as well as syntactic nodes.)

Once the meaning construction and text correction phase is complete, a final selection process ranks the interpretations according to meaningfulness and lack of errors. The interpretations are displayed to the user in order of preference. The user selects one of the interpretations and can then interact with the system to further improve the message and to undo bad corrections by the system.

2.5 KNOWLEDGE BASE

The knowledge base of VOX consists of a database, a dictionary, and a knowledge manager.

The dictionary is merely a list of words with pointers into the database. It is implemented as a file system, and contains many words that have not yet been

defined.

The database is also implemented as a file system, and contains the entire implemented Conceptual Grammar. The database contains many objects and is organized hierarchically. A discussion of the database and how Conceptual Grammar information is represented is beyond the scope of this paper.

The knowledge manager controls the flow of information between the database, dictionary, analyzer, and extensibility system. It provides facilities for updating and accessing the knowledge base, and for determining the status of information, i.e., whether it exists, is in core or in the knowledge base, whether it is slated for addition or deletion by the extensibility system, and so on. The manager also provides facilities for structured searches of the knowledge base, that are mainly used by the extensibility system.

3. CONCEPTUAL GRAMMAR

3.1 INTRODUCTION

Conceptual Grammar (CG) is a framework for the representation of conceptual information. We loosely define anything that can be verbalized as 'conceptual'.

The unit of knowledge in CG is the *concept.* A concept is an atomic representation of anything that can be verbalized. In our notation, a concept is depicted by a description of the concept enclosed in angle brackets. For example,

<aircraft carrier (noun)>

is an atomic representation of the concept "aircraft carrier". We often omit the angle brackets and hyphenate the description, for simplicity:

aircraft-carrier or <aircraft-carrier>

Concepts can be combined to form *phrases.* Most phrases have associated concepts to represent their meaning. Concepts and phrases *suggest*, or *reduce to*, other concepts by means of grammar rules. Some typical rules in CG are shown in Figure 1.

(A) aircraft carrier	----->	aircraft-carrier
(B) aircraft-carrier	----->	ship
(C) attack (vp)	----->	attack (event)
(D) ship	----->	noun
(E) det quan adj noun	----->	<specific np>

FIGURE 1

Rule (A) has a phrase suggesting its atomic representation. The phrase "aircraft

carrier" has no corresponding single English word, yet it is a well-defined object, so we represent it with an atomic concept.

Rule (B) is an example of a hierarchical rule. The *hierarchy* is an important semantic organization scheme in Conceptual Grammar. CG has hierarchies for nouns, adjectives, and some other parts of speech, as well as events and scenarios. For example, a scenario where a ship attacks a submarine is a more specific instance of one where a platform attacks a platform, which is a more specific version of the generic attack scenario, and so on. ('platform' is a Navy word for anything that missiles can be fired from.)

Rule (C) illustrates a second organizing principle of CG -- *conceptual levels.* When we speak of 'attack', we may be talking about the word itself, the verb, the action, the event, or an entire scenario. Conceptual Grammar treats all of these facets as explicit concepts. These semantic concepts correspond to the syntactic concepts word, verb, verb phrase, and so on. The semantic levels of event and frame (or scenario) have no syntactic equivalents. A frame could correspond to a sentence, a paragraph, or even a novel. Objects have a similar set of conceptual levels corresponding to word, noun, and noun phrase.

The semantic concepts at differing conceptual levels allow semantic phrases to be represented unambiguously in Conceptual Grammar. Note how the concept of 'ship' is used in

(1) <ship (np)> <attack (vp)> <submarine (np)>
(2) <ship (noun)> <ahoy (word)> <exclamation mark>
(3) <ship (word)> <hyphen> <shape (word)>

Phrase (1) would match a text like "The 3 US destroyers will attack the enemy sub". Phrase (2) matches only "ship ahoy!", "destroyers ahoy!", etc. Phrase (3) matches only "ship-shape". Conceptual levels allow semantic phrases to be represented with a high degree of precision.

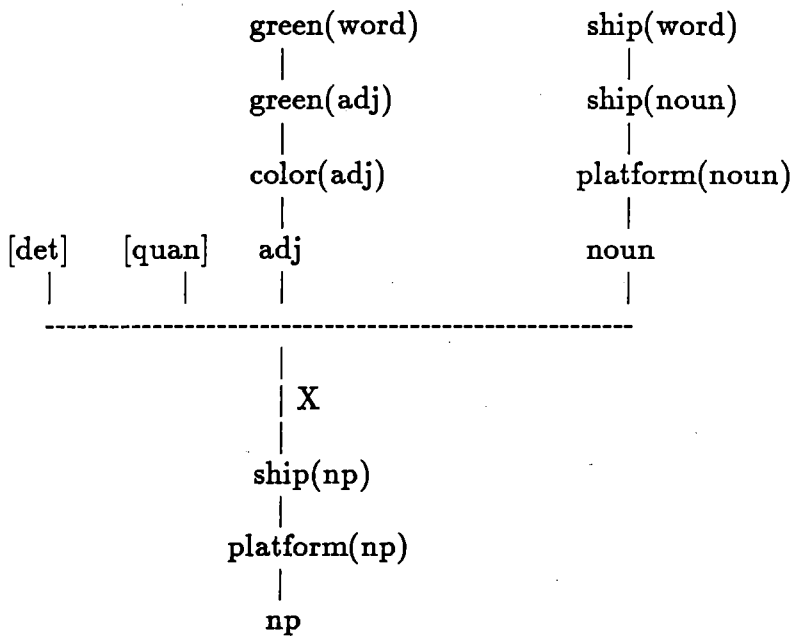
An important class of rules in Conceptual Grammar is concerned with the transitions between semantic and syntactic phrases. In Figure 1 above, rule (D) shows a semantic concept suggesting a syntactic concept, while rule (E) shows the reverse. Rule (E) is an example of a *restriction rule.* It suggests a specific noun-phrase concept corresponding to the noun on the left-hand-side of the rule. We will discuss this kind of rule in more detail in the next section.

3.2. EXAMPLE

We will illustrate how Conceptual Grammar analyzes

"Green ship will fire 2 missiles at 1230pm at submarine"

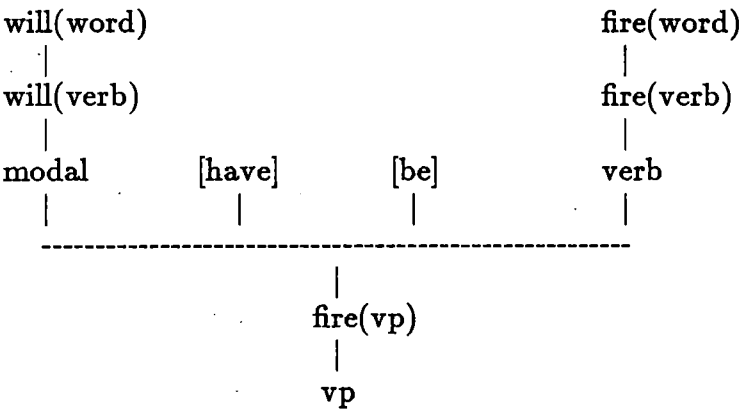
"Green ship" is analyzed as follows:



Most important here is step X, which uses the grammar rule:

det quan adj noun -------> <specific np>

Since <ship (noun)> gave rise to the noun in the left-hand-side, this grammar rule reduces to <ship (np)>. In essence, "green ship" has been condensed to the *semantic* concept <ship (np)>.



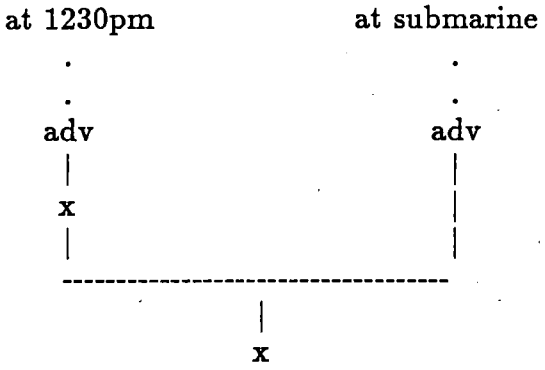
Here again, we are using a semantic restriction rule:

modal have be verb ------> <specific vp>

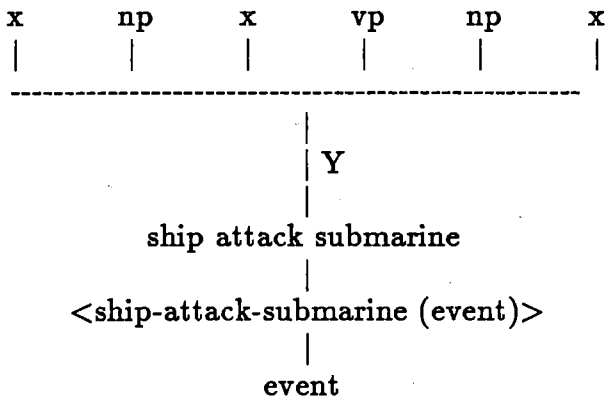
This rule finds the most specific possible semantic instance of the verb, and suggests it.

"2 missiles" is analyzed almost identically to "green ship".

The detailed analysis of the prepositional phrases "at 1230pm" and "at submarine" is omitted for simplicity. Both will suggest the concept <adv>, which corresponds to prepositional and adverbial phrases. Also, we represent a list of adverbial phrases by <x>, for short.



Now, we come to the most interesting part of the analysis:



Rule Y looks like

(Y) x np x vp np x ------> <specific event>

The task of rule Y is to find specific events in the database, and it found an event "ship attack submarine". Now, rule Y is highly sophisticated, and we will describe some of the actions that it took. First, note that "ship fired missile at submarine" is syntactically ambiguous. "At submarine" could be analyzed as a prepositional phrase, or "fire ... at" could be recognized as a prepositional verb with its associated particle. Rule Y knows about both of these possibilities. It checks the kind of verb, and uses one of the rules A or B accordingly:

(A) x np x <vp (prep)> np x prep np x --> <specific event>

(B) x np x <vp (regular)> np x --> <specific event>

If the verb can be prepositional, rule Y looks for phrases like

<fire (vp)> <weapon (np)> <at> --> <attack (vp)>

In our case, it will find and use this rule. Having found that "will fire 2 missiles at" corresponds to an 'attack' concept, rule Y then looks for the most specific possible event of the type

<ship> <attack> <submarine>

or

<platform> <attack> <platform>

and so on. Once the most specific possible event is found, rule Y suggests it. Note that rule Y has to search through the adverbial-list to find possible prepositional particles for the prepositional verb, and that it rejected the time adverbial because the knowledge base had no information about attacks on 'time'.

Rules like Y form a critical part of the Conceptual Grammar. They not only provide a mapping from surface text to underlying concepts, but also handle syntactic ambiguity in a unified and non-combinatorially explosive fashion. (Using rules A and B instead of Y would always result in two interpretations, whereas Y chooses the best one.)

Another example of such a rule is

x np x <vp (passive voice)> x --> <specific event>

which handles forms like

The ship was attacked by the submarine
Missiles were fired at ship by submarine
Missiles were fired by submarine at ship

again, in non-combinatorially explosive fashion. Furthermore, this rule knows how to search for active-voice events, thus allowing most event knowledge to be stored in active voice. If desired, this rule could also transform passive voice sentences to active voice. Another critical rule is

<event-list> -----> <specific frame>

The task of this rule is to find a single frame (or scenario) which will unify a sequence of actions, and to try to determine the causal relationship of all the events. This rule can embody the analyzer's frame-selection mechanisms.

3.3. DISCUSSION

Conceptual Grammar is seen to intimately combine syntactic and semantic knowledge in the analysis of English text. The mechanisms are designed so as to strip away syntactic details while extracting the underlying concepts from text.

We do not claim that CG alone provides a complete meaning representation for text. Rather, it provides a framework from which meaningful information can be obtained. To fully understand text, we must effectively build a real-world model of what the text describes, including the causal relationships between events, the actors and the reasons for their actions, and so on.

Any desired meaning construction system can easily be incorporated into the Conceptual Grammar framework. The construction actions are executed when a grammar reduction is performed.

Part of the work on Conceptual Grammar consists in incorporating more semantic information *about* the phrases in the knowledge base. For example, in a scenario phrase such as that of our example extensibility session [*Section 1.1*], we have knowledge of which actor roles are present in the scenario, which is the most important event of the scenario, and which events are critical to a complete description of the scenario.

We should note that the implemented Conceptual Grammar lags significantly behind the theory. Similarly, the extensibility system lags far behind the capabilities of the implemented grammar.

4. OTHER EXTENSIBILITY WORK

Automatic extensibility is a relatively unexplored area of Natural Language Processing. While many systems and knowledge representations may be domain-independent and extensible in principle, this has not usually been demonstrated by implementation.

The UC system of Wilensky [*1984*], which is based on PHRAN, has a UC Teacher component by which a user can add knowledge by telling it new facts. Unlike VOX, the UC Teacher makes use of simple definitional knowledge provided by the user. Most knowledge must still be added to PHRAN by editing complex LISP data structures, however. Also, the UC Teacher is passive -- it has no capacity to ask the user for information about a new definition.

The LIFER system [*Hendrix*], which uses a semantic grammar much like PHRAN patterns, has a component for adding new patterns. As in PHRAN, the role of pure syntax in LIFER is minimal, and new semantic information must be added programmatically.

KLAUS [*Haas 1980, Grosz 1984*], like the UC Teacher, functions by the user telling the system new facts. KLAUS also takes an active role to assure that new knowledge is being acquired completely and in linguistically correct fashion. Currently, KLAUS can interact with a more linguistically naive user than VOX.

Both KLAUS and UC attempt to interact in English with the user, though both interfaces are fairly crude, and require much knowledge about what language

forms the system can handle, in addition to knowledge about concepts the system understands. VOX currently interacts in a fairly canned manner, displaying menus and short-answer questions so as to minimize typing by the user. Both kinds of interface are important in order to maximize the user's convenience.

Some specialized systems use automatic extensibility tools as well. For example, TEAM [*Grosz, 1983*] provides natural language interfaces for database systems. The acquisition component of TEAM interactively gains knowledge about how users phrase their queries in natural language.

In the dialogues to acquire verbs, nouns, and other parts of speech, VOX, TEAM, and KLAUS are fairly similar, in principle. Yet, VOX alone has the capacity to acquire and make use of knowledge about scenarios in a systematic way.

5. KNOWLEDGE REPRESENTATION AND EXTENSIBILITY

Little is known about extensible knowledge representations. We have elaborated a tentative set of guiding principles, though we make no claims for necessity and sufficiency. A desirable knowledge representation is:

1. *Simple* - the knowledge representation must consist of a small number of primitive elements. The uniformity so provided can then be used by extensibility mechanisms. The greater the variety of data structures, the more difficult it is to construct extensibility mechanisms.

2. *Modular* - addition of knowledge can occur in small, discrete pieces. In our representation, there is no such thing as 'complete' knowledge. Rather, the more knowledge the system has about a concept, the richer the understanding of that concept.

3. *General* - the knowledge representation must be able to represent a wide range of concepts, and to relate such concepts to each other. Abstract and concrete knowledge must both be representable.

4. *Episodic* - the knowledge representation must distinguish between concepts and instances of concepts.

5. *Organized* - the knowledge representation must be able to express relationships between concepts.

6. *Verbal* - any piece of knowledge should be verbalizable.

7. *Declarative* - all concepts must be representable declaratively.

8. *Pragmatic* - the knowledge representation must be closely integrated with processing mechanisms that use it.

ACKNOWLEDGMENT

Thanks to Richard Granger for his support and for critiquing this paper and to Laura Yoklavich for formatting it.

APPENDIX: PROGRAM STATISTICS

VOX is written in UCI MLISP, an Algol-like dialect of LISP. The code occupies more than 12000 lines of UCI LISP. The extensibility system uses 6000 lines. The rest is split among the analyzer, database manager, and other utilities.

The knowledge base of VOX consists of a database file system and a dictionary file system. The database has about 14000 lines of LISP statements. The dictionary has 25000 entries, only a small percentage of which have database information. The auto-extensibility system has 14000 lines of commands for knowledge addition.

There are over 300 vocabulary-related database entries, each of which holds all the conjugations, parts of speech, and senses of a word. There are about 100 phrases (length $>$ 1) in the database. The number of grammar rules is about 3000.

The analyzer uses about half a minute of CPU time per word analyzed. This includes analysis, error detection, and text rewording. Little has been done to optimize the analyzer.

REFERENCES

Birnbaum, L. and Selfridge, M. (1980). Conceptual Analysis of Natural Language. In *Inside Computer Understanding*, Schank, R.C. and Riesbeck, C., eds. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Granger, Richard H., Amnon Meyers, Gregory B. Taylor, and Rika Yoshii (1983). NOMAD: A Naval Message Understanding System. AI Project, ICS Department, Irvine, CA. UC Irvine Technical Report 209.

Also in *Proceedings of the Conference on Artificial Intelligence*, April 26, 1983. Oakland University, Rochester, Michigan.

Granger, R.H. (1984). The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Text. UC Irvine Technical Report 226.

Also in *American Journal of Computational Linguistics.* v.9, no.3-4.

Grosz, Barbara J. (1983). TEAM: A Transportable Natural Language Interface System. *Conference on Applied Natural Language Processing*, San-

ta Monica.

Grosz, Barbara J. and Mark E. Stickel (1984). Research On Interactive Acquisition And Use Of Knowledge. SRI Technical Report.

Haas, N. and Hendrix, G. G. (1980). An Approach To Acquiring And Applying Knowledge. *First National Conference on Artificial Intelligence.*

Hendrix, Gary G. (1977). The LIFER Manual: A Guide to Building Practical Natural Language Interfaces. SRI Technical Note 138.

Meyers, Amnon (1983). Conceptual Grammar. AI Project, ICS Department, Irvine, CA. UC Irvine Technical Report 215.

Schank, R.C. and R. Abelson (1977). *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Associates, Inc., Publishers. Hillsdale, New Jersey.

Wilensky, Robert and Yigal Arens (1980).
a) PHRAN - A Knowledge Based Approach to Natural Language Analysis. UC Berkeley. Electronic Research Laboratory Memorandum No. UCB/ERL M80/34.

b) PHRAN - A Phrasal Natural Language Understander. *ACL 80.*

Wilensky, Robert, Yigal Arens, and David Chin (1984). Talking to UNIX in English: An Overview of UC. *CACM* vol. 27, no. 6, pp.574-593.

Wilensky, Robert (1984). Knowledge Representation - A Critique and a Proposal. *Proceedings of the First Annual Workship on Theoretical Issues in Conceptual Information Processing.* Atlanta, Georgia. pp. 148-159.