VOX—An Extensible Natural Language Processor

Amn on Meyers

Artificial Intelligence Project
Computer Science Department
University of California
Irvine, California

ABSTRACT

\ OX is a Natural Language Processor whose knowledge ran 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.

1. INTRODUCTION

The ability to learn is one of the most important charac teristics of intelligent systems. To approach such an ability, we first must build systems that can accept newknowledge automatically. By continually enhancing the extensibility capability 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 ('onceptual Grammar [5], supports a bottom-up study of language, by representing both very general and very specific knowledge. As generalizations about language are liscovered, they are 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 uses 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:

slop searches for ship,
ship sights ship,
ship approaches ship,
ship attacks ship,
ship damages ship

We will add the words, the individual events, and the entire scenario to the system. Then, we will show a text analysis example that uses this knowledge [User inputs are in boldface, in the examples below,]

MACRO NOUN EXAMPLE:

Enter singular form of noun: ship
Pinter plural form of noun: ships
Enter synonym or more general concept platform

Macro noun is an extensibility capability for adding nouns. The words 'ship[1], '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 gre^n ships" will be found to be equivalent to ship(np), for example. By specifying 'platform[1], the user plan's 'ship[1] into a conceptual hierarchy of nouns already containing 'platform'. (Platform[1] is a Navy word for anything that a missile can be fired from. Thus, a base, a submarine, and an aircraft an-[1] all platforms.)
Macro verb is similar to macro noun. In addition to concepts for words, verb, and verb phrase levels, macro verb creates concepts for the event and frame level. For example, in adding the verb search, the concepts search (event) and search (frame) will be created.

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:

This work is supported by the Naval Ocean Systems Center, under NO SO C ran I N 66001 - 83- C- 0255.

MACRO EVENT EXAMPLE:

```
Enter an event:

        ship search  location for ship
          1      2          3      4    5

Semantic information

    Enter position of the following:
        actor = 1
        act = 2
        object = 5
        instrument = __
        location = 3
        time = __
    Enter a 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 full-fledged concept. The concept <ship-sean h-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 specific event-concept is added to a hierarchy of events by suggesting 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 (dement 5. The syntactic component of VOX's 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 (dement '3 can be skipped over; that is, "The ship searched for the submarine', omitting a location element, is correct Fnglish. 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:

```
Enter the events of the frame:
    1  ship-search
    2  ship-sight
    3  ship-approach
    4  ship-attack
    5  ship-damage

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 concept suggested by frame: 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 given full concept status, 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 and skip over any events. For example, a complete text might read "damaged sub". On the other hand, we require that an attack scenario end with an attack or damage event (event 4 or 5).

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 just entered.

```
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 builds a frame-based representation of the underlying meaning, which is used for cheeking and correcting syntactic and semantic problems. VOX reports errors found and produces a reworded version of the input text. VOX's use of extensibility session information in the above example is fairly straightforward.

2. CONCEPTUAL GRAMMAR

We describe the Conceptual Grammar knowledge representation, which forms the foundation of the extensibility and analysis capabilities of VOX.

2.1 INTRODUCTION

Conceptual Grammar (CG) is a framework for the representation of conceptual information. 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

In rule (A), a phrase suggests its atomic representation. The phrase "aircraft carrier"[1] 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 organizing principle of CG. CG has semantic hierarchies of 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 seenario, 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. CG treats all of these facets as explicit concepts. The lower-level semantic concepts correspond to the syntactic concepts word, verb, verb phrase, and so on. The higher semantic levels of event and frame (or scenario) have no precise syntactic equivalent. A frame could correspond to a sentence, a paragraph, or even a novel

The different levels of semantic concepts allow semantic phrases to be represented unambiguously in CC. Note how the concept of 'ship' is used in

(1) <ship(np)>	<attaek(vp)>	<submarine(np)>
(2) <ship(noun)>	<ahoy(word)>	<T>
(3) <ship(word)>	<hyphen>	<shape(word)>

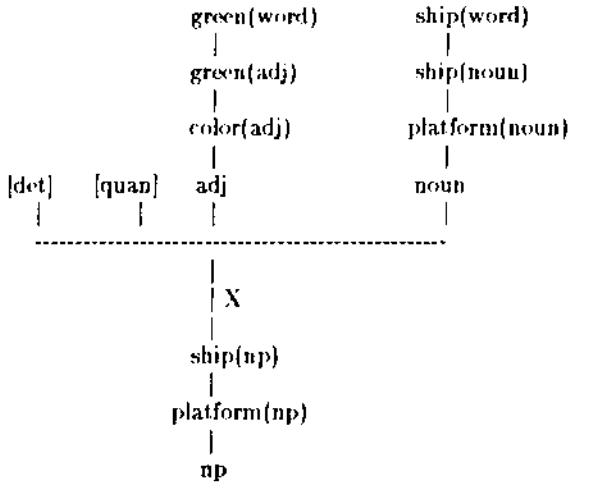
Phrase (1) would match a text like "The *3 U*S destroyers will attack the enemy sub". Phrase (2) matches only 'ship ahoy' ," "destroyers ahoy!", etc. Phrase (3) matches only "ship-shape". Conceptual levels allow semantic phrases to be represented with a high degree of precision.

An important class of rules in CG is concerned with the transitions between semantic and syntactic phrases. In Figure 1 above, rule (I)) 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.

2.2. EXAMPLE

We show, step-by-step, how CG analyzes

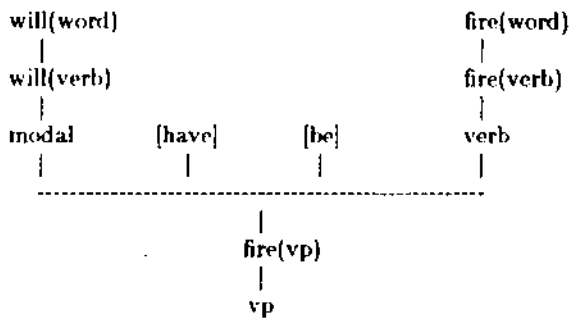
"Green ship will fire 2 missiles at 1230pm at submarine".



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

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

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



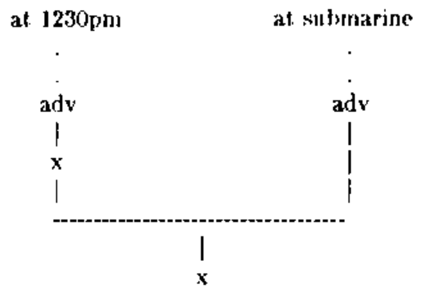
Here again, we are using a semantic restriction rule:

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

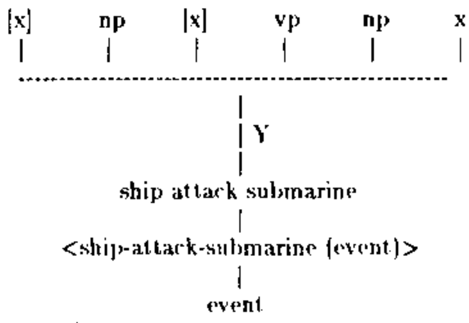
This rule finds the most specific possible semantic instance of the verb, and suggests its corresponding vp-level.

"2 missiles" is analyzed similarly 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 try to merge the elements found for the given sentence:



Rule Y looks like

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

The task of rule V is to find specific events in the data-base; in this case, it found the 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 npx prep rip x --> <specific event>

B

x np x vp-regular np x —> < specific event >

If the verb can be prepositional, rule Y looks for rules such as

<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 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 has no information about attacks on 'time'.

Rules like Y form a critical part of CG. They not only provide a mapping from surface text to underlying concepts, but also handle syntactic ambiguity in a unified and non-cornbinatorially explosive fashion. (Using rules A and R 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)> 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 is able to search for active-voice events, thus allowing most event knowledge to be stored in active voice. If desired, this rule can 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 events, and to try to determine the causal relationship of all the events. This rule embodies some of the analyzer's frame-selection mechanisms.

3. RELATED WORK

VOX is a revised version of NOMAD [1].

CG developed from our attempt to reconcile the I'll RAN approach [6] with traditional syntax theories, and to system ize the representation of phrasal knowledge.

We know of several efforts to build automatically extensible NLP systems. The Teacher component of UC [7], LIFER [4], KLAUS [3], TEAM [2], and others. The success of these efforts depends, ultimately, on the underlying knowledge representation, including both conceptual and linguistic knowledge. We find that CG provides a framework for both productive and nonproductive linguistic knowledge, while the other systems mentioned tend to concentrate on one type or the other. Also, CG alone provides for automatic addition of scenarios.

ACKNOWLEDGMENT

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

REFERENCES

[1] Granger, R.H. (1984). The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Text. *American Journal of Computational Linguistics,* v.9, no.3-4.

[2] Grosz, Barbara J. (1983). TEAM: A Transportable Natural Language Interface System. *Conference on Applied Natural Language Processing,* Santa Monica.

[3] Grosz, Barbara J. and Mark E. Stickel (1984). Research on Interactive Acquisition and Use of Knowledge. SRI Technical Report.

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

[5] Meyers, Amnon (1983). Conceptual Grammar. AI Project, ICS Department, Irvine, California. UC Irvine Technical Report 215.

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

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