TABLE 1

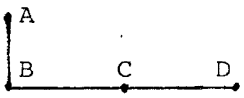
```
*BNF COMPONENT, INTRODUCTORY ENGLISH STRING GRAMMAR
    <SENTENCE> ::= <CENTER>'.' .                                          1.
    <CENTER>   ::= <ASSERTION> .                                          2.
    <ASSERTION>::= <SA><SUBJECT><SA><VERB><SA><OBJECT> <SA>.              3.
    <SA>       ::= <*D>|<*INT>|<PN>|<*NULL>.                              4.
    <PN>       ::= <*P><NSTG>.                                            5.
    <SUBJECT>  ::= <NSTG>.                                                6.
    <NSTG>     ::= <LNR>|<LPROR>|<NREP>.                                  7.
    <LNR>      ::= <LN><*N><RN>.                                          8.
    <LN>       ::= <TPOS><APOS>.                                          9.
    <TPOS>     ::= <*T>|<*NULL>.                                         10.
    <APOS>     ::= <*ADJ>|<*NULL>.                                       11.
    <RN>       ::= <PN>|<*NULL>.                                         12.
    <LPROR>    ::= <LPRO><*PRO><RN>.                                     13.
    <LPRO>     ::= <*NULL>.                                              14.
    <NREP>     ::=  WHAT'<SUBJECT><SA><VERB><SA>.                        15.
    <VERB>     ::= <LTVR>.                                               16.
    <LTVR>     ::= <LV><*TV><RV>.                                        17.
    <LV>       ::= <*D>|<*NULL>.                                         18.
    <RV>       ::= <*D>|<PN>|<*NULL>.                                    19.
    <LVR>      ::= <LV><*V><RV>.                                         20.
    <OBJECT>   ::= <NSTG>|<THATS>|<*NULLOBJ>|<TOVO>.                     21.
    <THATS>    ::=  THAT <ASSERTION>.                                    22.
    <TOVO>     ::= TO<LVR><SA><OBJECT><SA>.                              23.
*LISTS
ATTRIBUTE       =  SINGULAR, PLURAL, NCOUNT, NHUMAN, OBJLIST,NOMINATIV
                                                          ACCUSATIVE.
TYPE STRING     =  ASSERTION, PN, NREP, THATS.
TYPE STGSEG     =  ASSERTION .
TYPE ADJSET     =  SA, LN, LPRO,  LV, RN, RV.
TYPE LADJSET    =  LN, LPRO, LV.
TYPE RADJSET    =  RN, RV.
TYPE LXR        = LNR, LPROR, LTVR.
```

*EXAMPLES OF DEFINITIONS

1. -
2. -
3. Generally they show films on Saturday
4. Generally / moreover / on Saturday
5. on Saturday, at installations
6. -
7. useful programs for linguists / they / what they do
8. useful grammars for programmers
9. the useful, our, lengthy,
10. the, a, our
11. useful, lengthy
12. at our installation
13. we in the suburbs
14. -
15. what he generally says
16. says, writes clearly
17. just says (nothing)
18. just, nearly
19. poorly, in haste
20. be
21. (write) lengthy letters, that he arrived, (wants) to leave
22. that he arrived
23. to show films

3. THE TREE REPRESENTATION AND TOP-DOWN PARSING

Syntax analysis is the task of determining what sequence or sequences of productions in the grammar will generate a given sentence of the language. We represent the sequence of productions which constitute one parse of the sentence by a parse tree. At the top of our parse tree we place the root node, SENTENCE. If the derivation of the sentence includes the production <A> → <B><C><D> (possibly one option of the BNF definition <A> ::= <B><C><D>|<E><F><G>|<H>), we place under the node A in the parse tree the nodes B, C, and D, connected together thus:

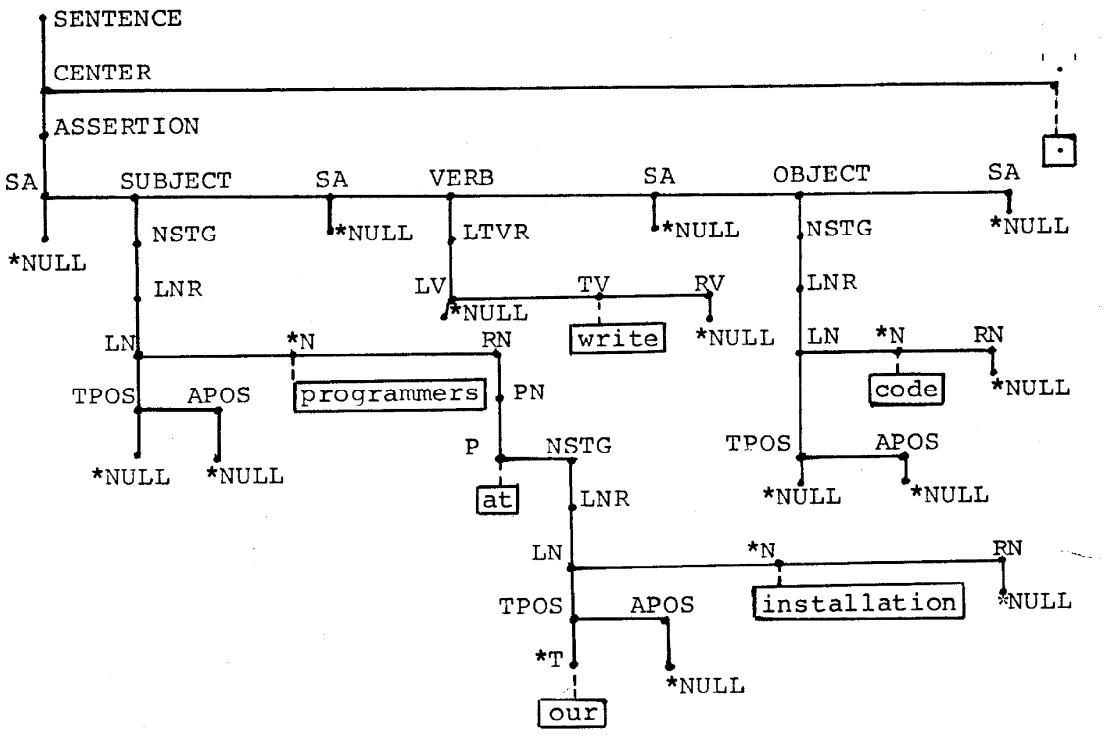


In this way, the parse tree for the sentence "Programmers at our installation write code." according to our small grammar appears as shown in Figure 1.

We analyze sentences and produce parse trees using an algorithm called a top-down parser. This algorithm is based on the idea of "analysis by synthesis." In its simplest realization, the parser successively generates all sentence strings of the language (by trying all possible combinations of choices of options) until it happens to generate the sentence to be parsed; the choices of options at that point are recorded as a parse tree. This approach is obviously absurdly inefficient. To speed things up, we compare the given sentence with the sentence strings we are generating during the generation process. For instance, if the parser has chosen an option which makes the first word of the generated string a NOUN, while the first

Fig. 1

Parse Tree for "Programmers at our installation write code."



word of the given sentence is a PRO, there is no point in generating all possible completions of the sentence with this particular option.

To see how the top-down parser works, we shall consider the analysis of the sentence "Tomorrow we parse." = D, PRO, (N|TV|V), "." with the super-trivial grammar

<SENTENCE> ::= <CENTER>'.'.

<CENTER> ::= <SA><SUBJECT><VERB>.

<SA> ::=⟪NULL⟫|<*D>.

<SUBJECT> ::= <*PRO>|<*N>.

<VERB> ::= <*TV>.

We are obliged to use such a small grammar because a complete exposition of the analysis of a sentence with even the simple English grammar would fill a large volume.

The parser begins by building a node corresponding to the root symbol [1]. It then attaches below the SENTENCE node the first element of the first (and only) option of SENTENCE, CENTER [2]. This process is repeated to add SA below CENTER and NULL below SA [3]. NULL is the special atomic symbol corresponding to the empty string, so it is always accepted without matching anything in the sentence.

[1]
. SENTENCE

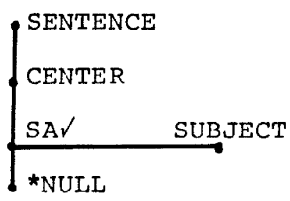
[2]
SENTENCE
|
CENTER

[3]
SENTENCE
|
CENTER
|
SA
|
*NULL

After the NULL is accepted, the parser checks whether there are any more elements in this option of SA.

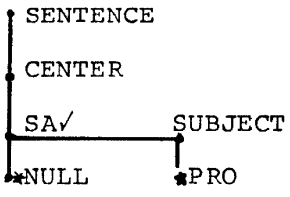
There aren't, so the parser marks the SA "completed" (indicated by a check mark on the tree). It then takes the next element in the definition of CENTER, namely SUBJECT, and attaches it to the right of SA [4].

[4]



Below SUBJECT the parser attaches the first element of its first option, PRO [5]. Since this is a non-NULL atomic, it is matched against the current (first) sentence word; the first word has only the category D, so the match fails.

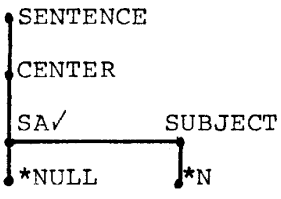
[5]



The parser must now "back up" to try to generate some alternate tree. It backs up by removing nodes in the tree in the reverse order from which they were added. Each time it detaches the first node in a level, it checks whether there are any as-yet untried options of the node above. If there are, it selects the next option and resumes building the tree. If there are no untried options, it continues backing up.

In this case, as soon as it has detached PRO it notices that SUBJECT has a second option, so it attachs the first element of this option, N, below SUBJECT [6]. This doesn't match the first word either, so the parser must back up again. This time

[6]

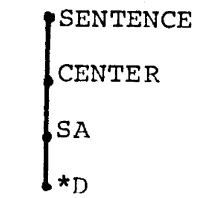


there are no more options of SUBJECT so the parser backs up further, detaching SUBJECT and NULL (note that when a node below SA is removed, the completion mark on SA is deleted). SA does have an untried option, so this is tried next [7]. The element just added is the atomic D, and this does match a category of the first sentence word. The parser can therefore accept the D, advance the "current word" pointer to the second word, and mark SA complete again [8].

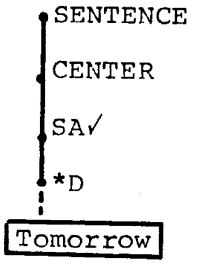
The parser can now reattach SUBJECT to the right of SA and PRO below SUBJECT. Since we are now at the second word, PRO matches and SUBJECT can be completed [9]. The parser may now proceed to attach VERB and TV below VERB. TV matches a category of the third word, so the parser can complete VERB and then, going up one level, complete CENTER [10].

Finally, the parser will attach the literal element '.' to the right of CENTER. A literal element is

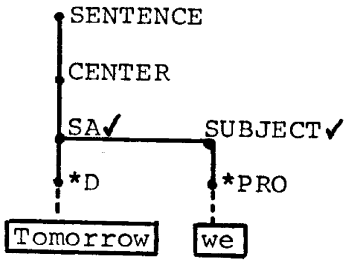
[7]



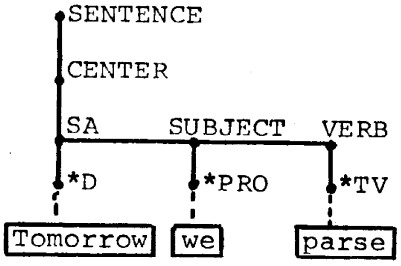
[8]



[9]

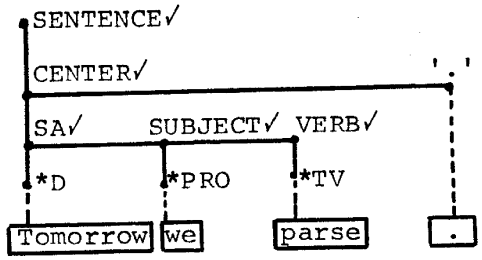


[10]



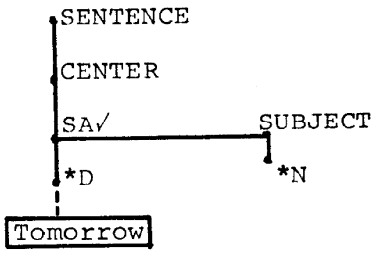
matched not against the category list of the current word but against the word itself. Since the current word is indeed '.', the parser can thus complete SENTENCE, yielding a parse tree [11].

[11]



After this parse tree has been completed, the parser must determine whether any alternate analyses can be obtained for the sentence. To do this, it uses the same back-up mechanism it employed earlier when it got stuck while building the tree.

[12]



It detaches '.' and TV; VERB has no alternate options so it detaches VERB and PRO. SUBJECT does have another option, so this is tried [12]. The N, however, does not match any category of the current word ("we"), so the retreat must be resumed. The parser detaches SUBJECT and D; both options of SA have already been tried, so the parser detaches SA. CENTER and SENTENCE have no alternate options, so they are also detached. When the root node is detached, the parsing process is complete--all combinations of options have been tried and no alternate analyses have been obtained.

4. THE WORD DICTIONARY

In order to parse a text, the parsing program requires three inputs: a grammar, a word dictionary containing entries for all of the words in the text, and the text itself. In a previous section we developed a small restrictionless string grammar in a form which is suitable for input to the LSP parser. We should now like to describe the form which the word dictionary should have in order to be an input to the parser.

First it should be noted that every lexically distinct item in the input text must have an entry in the word dictionary. This includes literals, such as punctuation marks, even though they have no category assignment. The entry for period, for example, is:

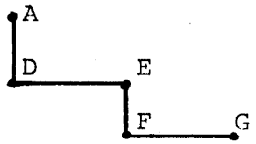
```
'.' .
```

Notice that the word defined in the above entry (the period) is enclosed in single quotation marks (').
It is important to know that all items containing special characters must be enclosed in ''. The more precise statement is that every item which contains a character other than a letter, digit or hyphen, or does not begin with a letter must be enclosed in '' (for example, 'CA++', 'DR.'). In addition, certain symbols which are handled in a special way by the parser must be enclosed in quotes ('). These include the articles A, AN, and THE; the ordinals FIRST, SECOND, and THIRD; and symbols used in the grammar, such as SENTENCE, SUBJECT, and STRING. When in doubt, quote--quotes may be used around any word in the dictionary.

Each dictionary entry is a definition of the word in question with respect to its grammatical properties. The elements of the definition in a word dictionary entry are the major category assignments of the word, and these are separated by commas. In addition, within each of its major category assignments the word may be a member of one or more subcategories of the major category, and, in turn, of one or more subcategories of each of these subcategories, and so forth. To avoid any confusion as to which subcategories belong to which parent category, each set of subcategories which belong to the same parent is enclosed in parentheses and preceded by a colon. For example, if a word has the major category assignments A, B, C, and as a member of A is in the subcategories D and E, and as a member of E is further subcategorized into F and G, the dictionary entry will have the form:

WORD A: (D, E:(F, G)), B, C.

The subcategory (or attribute structure within each major category assignment of a word can be represented as a tree. Thus, WORD, above, as an A, has the attribute structure:



In parsing a sentence, when a major category of a sentence word is found to match a category on the parse tree, the program attaches the subtree of that category assignment of the word to the matching category in the parse tree. It then becomes possible to test whether the words corresponding to atomic nodes of the parse tree have particular attributes, as is often done

in restriction tests.

If we parse using the BNF grammar alone, without restrictions, only the major categories of words will be tested. However, to lay the groundwork for a restriction component of the grammar we will develop some word dictionary entries which include both the major word categories of the small BNF grammar of section 3 and the subcategories which are named on the list ATTRIBUTE following the BNF definitions. These attributes are SINGULAR, PLURAL, NCOUNT, NHUMAN, OBJLIST, NOMINATIVE, ACCUSATIVE.

As noted earlier, the attributes SINGULAR and PLURAL are defined in the footnote to page 5,and NCOUNT is described on p. 10. NHUMAN is the class of "human" nouns; more precisely, a noun is NHUMAN if it can be adjoined by a right adjunct string beginning with "who" or "whom", e.g. "a man whom I know." The attribute OBJLIST applies to verbs, and is a list of those object strings in the grammar (options of OBJECT) which can occur as the object of the given verb. Thus, the TV and V categories of "stretch" each have the attribute OBJLIST: (NSTG, NULLOBJ) because of the existence of sentences like "Users stretch tapes" (NSTG object) and "Tapes stretch" (NULLOBJ). The full dictionary entry for "stretch" in terms of the categories of the small grammar is:

```
STRETCH   N:  (SINGULAR, NCOUNT), TV(PLURAL,
          OBJLIST:(NSTG, NULLOBJ)), V:(OBJLIST:
          (NSTG, NULLOBJ)).
```

A small word dictionary, which can be used to test the small grammar, is shown in Fig. 2.

Fig. 2

Small Word Dictionary for the Small Grammar

```
*WD
  'A'  T.
  'THE'  T.
 '.' .
MY T.
WHAT .
CAT N:(SINGULAR, NCOUNT).
CATS N:(PLURAL).
EGG N:(SINGULAR, NCOUNT).
EGGS N:(PLURAL).
FISH N, TV:(PLURAL, OBJLIST(NULLOBJ)), V:(OBJLIST:(NULLOBJ)).
MEANS N:(PLURAL), TV:(SINGULAR, OBJLIST:(THATS, NSTG, TOVO)),
   V:(OBJLIST:( THATS, NSTG, TOVO)).
MEAN N:(SINGULAR, NCOUNT), TV:(PLURAL, OBJLIST:(THATS, NSTG,
   TOVO)), V:(OBJLIST:(THATS, NSTG, TOVO)), ADJ.
IS TV:(SINGULAR, OBJLIST:(NSTG, ADJ)).
EATS TV:(SINGULAR, OBJLIST:(NSTG, NULLOBJ)).
EAT TV:(PLURAL, OBJLIST:(NSTG, NULLOBJ)), V:(OBJLIST:(NSTG, NULLOBJ)).
SLEEPS TV:(SINGULAR, OBJLIST:(NULLOBJ)).
SLEEP TV:(PLURAL, OBJLIST:(NULLOBJ)), V:(OBJLIST:(NULLOBJ)).
HAPPY ADJ.
GENERALLY D.
WITH P.
TO P.
HE PRO:(SINGULAR, NOMINATIVE, NHUMAN).
HIM PRO:(SINGULAR, ACCUSATIVE, NHUMAN).
SHE PRO:(SINGULAR, NOMINATIVE, NHUMAN).
HER PRO:(SINGULAR, ACCUSATIVE, NHUMAN).
THEY PRO:(PLURAL, NOMINATIVE).
THEM PRO:(PLURAL, ACCUSATIVE).
THAT.
EVEN  D, ADJ.
ONLY  D.
```

Fig. 2a

MAJOR CATEGORY DEFINITIONS

A word is classified as a noun if it can occur in a possessive form, e.g.:

John's
Washington's
blood's
man's
men's

A word is classified as an adjective if it can occur with the suffixes -er (or more) and -est (or most) in the environment

is [more / the most] ———— .

and/or is capable of forming an adverb with the addition of -ly, e.g.:

Mary is prettier./ the prettiest.
This explanation is more fundamental./the most fundamental.
He merely looked up.

A word is classified as a verb if it occurs in at least three of the five inflectional forms

Stem	Present 3rd sing.	Present Participle	Past Tense	Past Participle
rise	rises	rising	rose	risen
go	goes	going	went	gone
leave	leaves	leaving	left	left

Despite the simple appearance of word dictionary entries, it is not always a simple matter to assign words to the grammatical categories represented in the word dictionary. While the criteria for assigning words to the various categories and subcategories are sharp (the definition of a category or subcategory is how it is used in the grammar), the decision as to whether a given word meets the criteria is not always easy to make. Even in the trivial word dictionary for our trivial grammar, there are even such questions; for example sleeps, and sleep: Should they be classified also as nouns (I had a good sleep,(?) I haven't had many good sleeps lately). A question of regularity is involved since generally if a word is classified as a noun in the singular one would expect it to have a plural form as well. (For our purposes, here, we have simply ignored the noun classification of sleep). These problems are many times greater using 25 major classes and 150 subclasses, as is done in preparing the LSP word dictionary at present. The decisions with regard to individual words rest in large measure on the type of textual material which is to be analyzed. Scientific English, for example, is very different from colloquial English, and even within the scientific area word usage varies from discipline to discipline. Once one has decided upon the general type of textual material, however, one has to be careful not to become too specific in characterizing a word's usage, so that real grammatical alternatives are not excluded.

Our equipment for parsing sentences now includes: a small BNF grammar covering a small arbitrary subset of English sentences,

a top-down context-free parser, and a small word dictionary in which words are assigned to the major categories appearing as the terminal symbols of the BNF grammar. We have also included in the word dictionary some subcategories which will be used in developing a restriction component for the small grammar. A few experiences with context-free parsing will point very quickly to the need for restrictions in the grammar, so we now turn our attention to this matter.