

5.6 COMPOUND

This program changes pairs of words to compounds, to guarantee more uniformity in morphological and lexical analysis. It requires that the user create a file of potential compound words in a format with each compound on a separate line, as in this example.

```
night+night
Chatty+baby
oh+boy
```

Whenever the program finds “night night” in the text, whether it be written as “night+night”, “night night” or “night-night,” it will be changed to “night+night”.

5.7 COMBO

COMBO provides the user with ways of composing Boolean search strings to match patterns of letters, words, or groups of words in the data files. This program is particularly important for researchers who are interested in syntactic analysis. The search strings are specified with either the +s/-s option or in a separate file. Use of the +s switch is obligatory in COMBO. When learning to use COMBO, what is most tricky is learning how to specify the correct search strings.

5.7.1 Composing Search Strings

Boolean searching uses algebraic symbols to better define words or combinations of words to be searched for in data. COMBO uses regular expressions to define the search pattern. These six special symbols are listed in the following table:

Table 3: COMBO Strings

Meaning	Type	Symbol
immediately FOLLOWED by	Boolean	^
inclusive OR	Boolean	+
logical NOT	Boolean	!
repeated character	metacharacter	*
single character	metacharacter	_
quoting	metacharacter	\

Inserting the ^ operator between two strings causes the program to search for the first string followed by the second string. The + operator inserted between two strings causes the program to search for either of the two strings. In this case, it is not necessary for both of them to match the text to have a successful match of the whole expression. Any one match is sufficient. The ! operated inserted before a string causes the program to match a string of text that does not contain that string.

The items of the regular expression will be matched to the items in the text *only* if they directly follow one another. For example, the expression `big^cat` will match only the word *big* directly followed by the word *cat* as in *big cat*. To find the word *big* followed by the word *cat* immediately or otherwise, use the metacharacter * between the

items *big* and *cat*, as in `big^*^cat`. This expression will match, for example, *big black cat*. Notice that, in this example, `*` ends up matching not just any string of characters, but any string of words or characters up to the point where *cat* is matched. Inside a word, such as *go*`*`, the asterisk stands for any number of characters. In the form `^*^`, it stands for any number of words. The `*` alone cannot be used in conjunction with the `+g` or `+x` option.

The underscore is used to “stand in for” for any *single* character. If you want to match any single word, you can use the underscore with the asterisk as in `+s"_*."` which will match any single word followed by a period. For example, in the string *cat.*, the underscore would match *c*, the asterisk would match *at* and the period would match the period.

The backslash (`\`) is used to quote either the asterisk or the underline. When you want to search for the actual characters `*` and `_`, rather than using them as metacharacters, you insert the `\` character before them.

Using metacharacters can be quite helpful in defining search strings. Suppose you want to search for the words *weight*, *weighs*, *weighing*, *weighed*, and *weigh*. You could use the string `weigh*` to find all of the previously mentioned forms. Metacharacters may be used anywhere in the search string.

When COMBO finds a match to a search string, it prints out the entire utterance in which the search string matched, along with any previous context or following context that had been included with the `+w` or `-w` switches. This whole area printed out is what we will call the “window.”

5.7.2 Examples of Search Strings

The following command searches the `sample.cha` file and prints out the window which contains the word “want” when it is directly followed by the word “to.”

```
combo +swant^to sample.cha
```

If you are interested not just in cases where “to” immediately follows “want,” but also cases where it eventually follows, you can use the following command syntax:

```
combo +s"want^*^to" sample.cha
```

The next command searches the file and prints out any window that contains both “want” and “to” in any order:

```
combo +s"want^to" +x sample.cha
```

The next command searches `sample.cha` and `sample2.cha` for the words “wonderful” or “chalk” and prints the window that contains either word:

```
combo +s"wonderful+chalk" sample*.cha
```

The next command searches `sample.cha` for the word “neat” when it is *not* directly followed by the words “toy” or “toy-s.” Note that you need the `^` in addition to the `!` in order to clearly specify the exact nature of the search you wish to be performed.

```
combo +s"neat^!toy*" sample.cha
```

In this next example, the COMBO program will search the text for either the word “see” directly followed by the word “what” or all the words matching “toy*.”

```
combo +s"see^(what+toy*)" sample.cha
```

You can use parentheses in order to group the search strings unambiguously as in the

next example:

```
combo +s"what*(other+that*)" sample.cha
```

This command causes the program to search for words matching “what” followed by either the word “that” or the word “other.” An example of the types of strings that would be found are: “what that,” “what’s that,” and “what other.” It will not match “what is that” or “what do you want.” Parentheses are necessary in the command line because the program reads the string from left to right. Parentheses are also important in the next example.

```
combo +s"the^^!grey^^(dog+cat)" sample2.cha
```

This command causes the program to search the file `sample2.cha` for *the* followed, immediately or eventually, by any word or words except *grey*. This combination is then to be followed by either *dog* or *cat*. The intention of this search is to find strings like *the big dog* or *the boy with a cat*, and not to match strings like *the big grey cat*. Note the use of the parentheses in the example. Without parentheses around *dog+cat*, the program would match simply *cat*. In this example, the sequence `^^` is used to indicate “immediately or later.” If we had used only the symbol `^` instead of the `^^`, we would have matched only strings in which the word immediately following *the* was not *grey*.

5.7.3 Referring to Files in Search Strings

Inside the `+s` switch, one can include reference to one, two, or even more groups of words that are listed in separate files. For example, you can look for combinations of prepositions with articles by using this switch:

```
+s@preps^@arts
```

To use this form, you first need to create a file of prepositions called “preps” with one preposition on each line and a file of articles called “arts” with one article on each line. By maintaining files of words for different parts of speech or different semantic fields, you can use COMBO to achieve a wide variety of syntactic and semantic analyses. Some suggestions for words to be grouped into files are given in the chapter of the CHAT manual on word lists. Some particularly easy lists to create would be those including all the modal verbs, all the articles, or all the prepositions. When building these lists, remember the possible existence of dialect and spelling variations such as *dat* for *that*.

5.7.4 Cluster Pairs in COMBO

Most computer search programs work on a single line at a time. If these programs find a match on the line, they print it out and then move on. Because of the structure of CHAT and the relation between the main line and the dependent tiers, it is more useful to have the CLAN programs work on “clusters” instead of lines. The notion of a cluster is particularly important for search programs, such as COMBO and KWAL. A cluster can be defined as a single utterance by a single speaker, along with all of its dependent tiers. By default, CLAN programs work on a single cluster at a time. For COMBO, one can extend this search scope to a pair of contiguous clusters by using the `+b` switch. However, this switch should only be used when cross-cluster matches are important, because addition of the switch tends to slow down the running of the program.

5.7.5 Searching for Clausemates

When conducting analyses on the %syn tier, researchers often want to make sure that the matches they locate are confined to “clausemate” constituents. Consider the following

two %syn tiers:

```
%syn: ( S V L ( O V ) )
%syn: ( S V ( S V O ) )
```

If we want to search for all subjects (S) followed by objects (O), we want to make sure that we match only patterns of the type found in the embedded clause in the second example. If we use a simple search pattern such as `+sS^*^O`, we will match the first example as well as both clauses in the second example. In order to prevent this, we need to add parentheses checking to our search string. The string then becomes:

```
+s"S^*^(!\(+!\))^*^O
```

This will find only subjects that are followed by objects without intervening parentheses. In order to guarantee the correct detection of parentheses, they must be surrounded by spaces on the %syn line.

5.7.6 Tracking Final Words

In order to find the final words of utterances, you need to use the complete delimiter set in your COMBO search string. You can do this with this syntax `(\!+?+.)` where the parentheses enclose a set of alternative delimiters. In order to specify the single word that appears before these delimiters, you can use the asterisk wildcard preceded by an underline. Note that this use of the asterisk treats it as referring to any number of letters, rather than any number of words. By itself, the asterisk in COMBO search strings usually means any number of words, but when preceded by the underline, it means any number of characters. Here is the full command:

```
combo +s"_*^(\!+?+.)" sample.cha
```

This can then be piped to `FREQ` if the `+d3` switch is used:

```
combo +s"_*^(\!+?+.)" +d3 sample.cha | freq
```

5.7.7 Tracking Initial Words

Because there is no special character that marks the beginnings of files, it is difficult to compose search strings to track items at utterance initial position. To solve this problem, you can run use `CHSTRING` to insert sentence initial markers. A good marker to use is the `++` symbol, which is only rarely used for other purposes. You can use this command:

```
chstring +c -t@ -t% +t* *.cha
```

You also need to have a file called `changes.cut` that has this one line:

```
":           "      ":           ++"
```

In this one-line file, there are two quoted strings. The first has a colon followed by a tab; the second has a colon followed by a tab and then a double plus.

5.7.8 Adding Excluded Characters

COMBO strings have no facility for excluding a particular set of words. However, you can achieve this same effect by (1) matching a pattern, (2) outputting the matches in `CHAT` format, (3) altering unwanted matches so they will not rematch, and (4) then rematching with the original search string. Here is an example:

```
combo +s"*ing*" +d input.cha | chstring +c +d -f | combo +s"*ing*"
```

The goal of this analysis is to match only words ending in participial *ing*. First, COMBO matches all words ending in *ing*. Then `CHSTRING` takes a list of unwanted words that end in *ing* like *during* and *thing* and changes the *ing* in these words to *iing*, for example. Then COMBO runs again and matches only the desired participial forms.

5.7.9 Limiting with COMBO

Often researchers want to limit their analysis to some particular group of utterances. CLAN provides the user with a series of switches within each program for doing the simplest types of limiting. For example, the +t/-t switch allows the user to include or exclude whole tiers. However, sometimes these simple mechanisms are not sufficient and the user will have to use COMBO or KWAL for more detailed control of limiting. COMBO is the most powerful program for limiting, because it has the most versatile methods for string search using the +s switch. Here is an illustration. Suppose that, in *sample.cha*, you want to find the frequency count of all the speech act codes associated with the speaker *MOT when this speaker used the phrase “want to” in an utterance. To accomplish this analysis, use this command:

```
combo +t*MOT +t%spa sample.cha +s"want^to" +d | freq
```

The +t*MOT switch (Unix users should add double quotes for +t"*MOT") tells the program to select only the main lines associated with the speaker *MOT. The +t%spa tells the program to add the %spa tier to the *MOT main speaker tiers. By default, the dependent tiers are excluded from the analysis. Then follows the file name, which can appear anywhere after the program name. The +s"want^to" then tells the program to select only the *MOT clusters that contain the phrase *want to*. The +d option tells the program to output the matching clusters from *sample.cha* without any non-CHAT identification information. Then the results are sent through a “pipe” indicated by the | symbol to the *FREQ* program, which conducts an analysis on the main line. The results could also be piped on to other programs such as *MLU* or *KEYMAP* or they can be stored in files.

Sometimes researchers want to maintain a copy of their data that is stripped of the various coding tiers. This can be done by this command:

```
combo +s* +o@ -t% +f *.cha
```

The +o switch controls the addition of the header material that would otherwise be excluded from the output and the -t switch controls the deletion of the dependent tiers. It is also possible to include or exclude individual speakers or dependent tiers by providing additional +t or -t switches. The best way to understand the use of limiting for controlling data display is to try the various options on a small sample file.

5.7.10 Adding Codes with COMBO

Often researchers leave a mark in a transcript indicating that a certain sentence has matched some search pattern. For example, imagine that you want to locate all sentences with a preposition followed immediately by the word “the” and then tag these sentences in some way. You can use the COMBO +d4 switch to do this. First, you would create a file with all the prepositions (one on each line) and call it something like *prep.cut*. Then you would create a second support file with this line:

```
"@prep.cut^the" "$Pthe" "%cod:"
```

The first string in this line gives the term used by the standard +s search switch. The second string says that the code produced will be \$Pthe. The third string says that this code should be placed on a %cod line under the utterance that is matched. If there is no %cod line there yet, one will be created. The COMBO command that uses this information would then be:

```
combo +s"@combo.cut" +d4 filename.cha
```

The resulting file will have this line added:

```
%cod: $Pthe
```

You can include as many lines as you wish in the `combo.cut` file to control the addition of additional codes and additional coding lines. Once you are done with this, you can use these new codes to control better inclusion and exclusion of utterances and other types of searches.

5.7.11 Unique Options

+b COMBO usually works on only one cluster at a time. However, when you want to look at a contiguous pair of clusters, you can use this switch.

+d Normally, COMBO outputs the location of the tier where the match occurs. When the `+d` switch is turned on you can output only each matched sentence in a simple legal CHAT format. The `+d1` switch outputs legal CHAT format along with line numbers and file names. The `+d2` switch outputs files names once per file only. The `+d3` switch outputs legal CHAT format, but with only the actual words matched by the search string, along with `@Comment` headers that are ignored by other programs. Try these commands:

```
combo +s"want^to" sample.cha
combo +s"want^to" +d sample.cha
combo +s"want^to" +d1 sample.cha | freq
combo +d2 +s"_*^." sample.cha | freq
```

This final command provides a useful way of searching for utterance final words and tabulating their frequency. The use of the `+d4` switch was described in the previous section.

+g COMBO can operate in either string-oriented or word-oriented mode. The default mode is word-oriented. COMBO can be converted to a string-oriented program by using the `+g` option. Word-oriented search assumes that the string of characters requested in the search string is surrounded by spaces or other word delimiting characters. The string-oriented search does not make this assumption. It sees a string of characters simply as a string of characters. In most cases, there is no need to use this switch, because the default word-oriented mode is usually more useful.

The interpretation of metacharacters varies depending on the search mode. In word-oriented mode, an expression with the asterisk metacharacter, such as `air*^plane`, will match *air plane* as well as *airline plane* or *airy plane*. It will not match *airplane* because, in word-oriented mode, the program expects to find two words. It will not match *air in the plane* because the text is broken into words by assuming that all adjacent nonspace characters are part of the same word, and a space marks the end of that word. You can think of the search string `air` as a signal for the computer to search for the expressions: `_air_`, `_air.`, `air?`, `air!`, and so forth, where the underline indicates a space.

The same expression `air*^plane` in the string-oriented search will match *airline plane*, *airy plane*, *air in the plane* or *airplane*. They will all be found because the search string, in this case, specifies the string consisting of the letters "a," "i," and "r", followed by any number of characters, followed by

the string “p,” “l,” “a,” “n,” and “e.” In string-oriented search, the expression (**air^plane**) will match *airplane* but not *air plane* because no space character was specified in the search string. In general, the string-oriented mode is not as useful as the word-oriented mode. One of the few cases when this mode is useful is when you want to find all but some given forms. For example if you are looking for all the forms of the verb *kick* except the *ing* form, you can use the expression “kick*^! ^!ing” and the +g switch.

+o The +t switch is used to control the addition or deletion of particular tiers or lines from the input and the output to COMBO. In some cases, you may want to include a tier in the output that is not being included in the input. This typically happens when you want to match a string in only one dependent tier, such as the %mor tier, but you want all tiers to be included in the output. In order to do this you would use a command of the following shape:

```
combo +t%mor +s"*ALL" +o% sample2.cha
```

+s This option is obligatory for COMBO. It is used to specify a regular expression to search for in a given data line(s). This option should be immediately followed by the regular expression itself. The rules for forming a regular expression are discussed in detail earlier in this section.

+t Particular dependent tiers can be included or excluded by using the +t option immediately followed by the tier code. By default, COMBO excludes the header and dependent code tiers from the search and output. However, when the dependent code tiers are included by using the +t option, they are combined with their speaker tiers into clusters. For example, if the search expression is **the^^kitten**, the match would be found even if *the* is on the speaker tier and *kitten* is on one of the speaker’s associated dependent tiers. This feature is useful if one wants to select for analyses only speaker tiers that contain specific word(s) on the main tier and some specific codes on the dependent code tier. For example, if one wants to produce a frequency count of the words *want* and *to* when either one of them is coded as an imitation on the %spa line, or *neat* when it is a continuation on the %spa line, the following two commands could be used:

```
combo +s(want^to^^^%spa:^^$INI*)+(neat^^^%spa:^^$CON*)
+t%spa +f +d sample.cha
freq +swant +sto +sneat sample.cmb
```

In this example, the +s option specifies that the words *want*, *to*, and *\$INI* may occur in any order on the selected tiers. The +t%spa option must be added in order to allow the program to look at the %spa tier when searching for a match. The +d option is used to specify that the information produced by the program, such as file name, line number and exact position of words on the tier, should be excluded from the output. This way the output is in a legal CHAT format and can be used as an input to another CLAN program, **FREQ** in this case. The same effect could also be obtained by using the piping feature, which is discussed in the section on **FREQ** on page [76](#).

+x COMBO searches are sequential. If you specify the expression **dog^cat**,

the program will match only the word “dog” directly followed by the word “cat”. If you want to find clusters that contain both of these words, in any order, you need to use the +x option. This option allows the program to find the expressions in both the original order and in reverse order. Thus, to find a combination of “want” and “to” anywhere and in any order, you use this command:

```
combo +swant^to +x sample.cha
```

COMBO also uses several options that are shared with other commands. For a complete list of options for a command, type the name of the command followed by a carriage return in the Commands window. Information regarding the additional options shared across commands can be found on page [132](#).

5.8 COOCCUR

The COOCCUR program tabulates co-occurrences of words. This is helpful for analyzing syntactic clusters. By default, the cluster length is two words, but you can reset this value just by inserting any integer up to 20 immediately after the +n option. The second word of the initial cluster will become the first word of the following cluster, and so on.

```
cooccur +t*MOT +n3 sample.cha +f
```

The +t*MOT switch tells the program to select only the *MOT main speaker tiers. The header and dependent code tiers are excluded by default. The +n3 option tells the program to combine three words into a word cluster. The program will then go through all of *MOT main speaker tiers in the sample.cha file, three words at a time. When COOCCUR reaches the end of an utterance, it marks the end of a cluster, so that no clusters are broken across speakers or across utterances. Co-occurrences of codes on the %mor line can be searched using commands such as this example:

```
cooccur +t%mor -t* +s*def sample2.cha
```

5.8.1 Unique Options

+d Strip the numbers from the output data that indicate how often a particular cluster occurred.

+n Set cluster length to a particular number. For example, +n3 will set cluster length to 3.

+s Select either a word or a file of words with @filename to search for.

COOCCUR also uses several options that are shared with other commands. For a complete list of options for a command, type the name of the command followed by a carriage return in the Commands window. Information regarding the additional options shared across commands can be found in chapter 6: Options on page [132](#).

5.9 DATES

The DATES program takes two time values and computes the third. It can take the child’s age and the current date and compute the child’s date of birth. It can take the date of birth and the current date to compute the child’s age. Or it can take the child’s age and the date of birth to compute the current date. For example, if you type:

6: Options

This chapter describes the various options or switches that are shared across CLAN commands. To see a list of options for a given program such as KWAL, type *kwal* followed by a carriage return in the Commands window. You will see a list of available options in the CLAN Output window.

Each option begins with a + or a -. There is always a space before the + or -. Multiple options can be used and they can occur in any order. For example, the command:

```
kwal +f +t*MOT sample.cha
```

runs a KWAL analysis on *sample.cha*. The selection of the +f option sends the output from this analysis into a new file called *sample.kwa.cex*. The +t*MOT option confines the analysis to only the lines spoken by the mother. The +f and +t switches can be placed in either order.

6.1 +F Option

This option allows you to send output to a file rather than to the screen. By default, nearly all of the programs send the results of the analyses directly to the screen. You can, however, request that your results be inserted into a file. This is accomplished by inserting the +f option into the command line. The advantage of sending the program's results to a file is that you can go over the analysis more carefully, because you have a file to which you can later refer.

The -f switch is used for sending output to the screen. For most programs, -f is the default and you do not need to enter it. You only need to use the -f switch when you want the output to go to the screen for CHSTRING, FLO, and SALTIN. The advantage of sending the analysis to the screen (also called standard output) is that the results are immediate and your directory is less cluttered with nonessential files. This is ideal for quick temporary analysis.

The string specified with the +f option is used to replace the default file name extension assigned to the output file name by each program. For example, the command

```
freq +f sample.cha
```

would create an output file *sample.frq.cex*. If you want to control the shape of the extension name on the file, you can place up to three letters after the +f switch, as in the command

```
freq +fmot sample.cha
```

which would create an output file *sample.mot.cex*. If the string argument is longer than three characters, it will be truncated. For example, the command

```
freq +fmother sample.cha
```

would also create an output file *sample.mot.cex*.

On the Macintosh, you can use the third option under the File menu to set the directory for your output files. On Windows you can achieve the same effect by using the +f switch with an argument, as in:

+fc:	This will send the output files to your working directory on c:.
+f".res"	This sets the extension for your output files.

+f'c:.res"

This sends the output files to c: and assigns the extension .res.

When you are running a command on several files and use the +f switch, the output will go into several files – one for each of the input files. If what you want is a combined analysis that treats all the input files as one large file, then you should use the +u switch. If you want all of the output to go into a single file for which you provide the name, then use the > character at the end of the command along with an additional file name. The > option can not be combined with +f.

6.2 +K Option

This option controls case-sensitivity. A case-sensitive program is one that makes a distinction between uppercase and lowercase letters. The CLAN programs, except for CHSTRING, are not case-sensitive by default. Use of the +k option in all of the other programs overrides the default state and allows them to become case-sensitive as well. For instance, suppose you are searching for the auxiliary verb “may” in a text. If you searched for the word “may” in a case-sensitive program, you would obtain all the occurrences of the word “may” in lower case only. You would not obtain any occurrences of “MAY” or “May.” Searches performed for the word “may” using the +k option produce the words “may,” “MAY,” and “May” as output.

6.3 +P Option

This option allows you to define a custom punctuation set. Because most of the programs in the CLAN system are word-oriented, the beginning and ending boundaries of words must be defined. This is done by defining a punctuation set. The default punctuation set for CLAN includes the space and these characters:

, . ; ? ! [] < >

This punctuation set applies to the main lines and all coding lines with the exception of the %pho and %mod lines which use the UNIBET and PHONASCII systems. Because those systems make use of punctuation markers for special characters, only the space can be used as a delimiter on the %pho and %mod lines.

All of the word-oriented programs have the +p option. This option allows the user to redefine the default punctuation set. This is useful because the CHAT coding conventions use special characters that at times are used as delimiters and other times as parts of words. For example, sometimes the - character is used as a morpheme boundary marker and, therefore, should not be considered part of the word. This is also quite useful when you are working on a language that uses diacritics. To change the punctuation set, you must create a small file that lists all the punctuation marks present in the file. You do this by simply typing out all the punctuation marks on a single line with no spaces between them. This line will change the punctuation set of the main speaker tiers and the code tiers. The name of your new punctuation file should immediately follow the +p in the command line. Here is an example situation. Suppose you wish to change both the main speaker tier and the code tier punctuation sets from the default to the set in newpunct.cut. The contents of the newpunct.cut file are as follows:

§*&^!

This line indicates the desired punctuation set for the main line and coding tier. You can now issue commands such as the following:

```
freq +pnewpunct.cut sample.cha
```

If you use the +p switch with no file name, the programs look for a file called punct.cut in the current working directory. If you do not use the +p switch at all, the programs look for a punctuation file called punct.cut. If the punct.cut file is not found, the program will then use the default built-in punctuation set. It is advisable to create a punct.cut file when the punctuation characters of the language being analyzed are different from the default punctuation characters. The punct.cut file should contain the new punctuation set and should be located in the current working directory. Because the punct.cut file is referred to automatically, this feature allows you to change the punctuation set once for use with all the CLAN programs. If you do not want CLAN to ever change the default punctuation set, make sure you do not have a punct.cut file in your current working directory and make sure you do not use the +p switch.

6.4 +R Option

This option deals with the treatment of material in parentheses.

+r1 Removing Parentheses. Omitted parts of words can be marked by parentheses, as in “(be)cause” with the first syllable omitted. The +r1 option removes the parentheses and leaves the rest of the word as is.

+r2 Leaving Parentheses. This option leaves the word with parentheses.

+r3 Removing Material in Parentheses. This option removes all of the omitted part.

Here is an example of the use of the first three +r options and their resulting outputs, if the input word is “get(s)”:

Option	Output
"no option"	gets
"+r1"	gets
"+r2"	get(s)
"+r3"	get

+r4 Removing Prosodic Symbols in Words. By default, symbols such as #, /, and : are ignored when they occur inside words. Use this switch if you want to include them in your searches. If you do not use this switch, the strings cat and ca:t will be seen as the same. If you use this switch, they will be seen as different. The use of these prosodic marker symbols is discussed in the CHAT manual.

+r5 Text Replacement. By default, material in the form [: text] replaces the material preceding it in the string search programs. If you do not want this replacement, use this switch.

+r6 Retraced Material. By default, material in retracings is included in

searches and counts. However, this material can be excluded by using the +r6 switch. In the MLU and MODREP programs, retracings are excluded by default. For these programs, the +r6 switch can be used to include material in retracings.

6.5 +S Option

This option allows you to search for a particular string. The +s option allows you to specify the keyword you desire to find. You do this by putting the word in quotes directly after the +s switch, as in +s"dog" to search for the word "dog." You can also use the +s switch to specify a file containing words to be searched. You do this by putting the file name after the +s preceded by the @ sign, as in +s@adverbs, which will search for the words in a file called adverbs.cut. If you want to look for the literal character @, you need to precede it with a backslash as in +s"@".

By default, the programs will only search for this string on the main line. Also by default, this switch treats material in square brackets as if it were a single "opaque" form. In effect, unless you include the square brackets in your search string, the search will ignore any material that is enclosed in square brackets. The COMBO program is the only one that allows you to specify regular expressions with this option. The only programs that allow you to include delimiters in the search string are COMBO, FREQ, and KWAL.

It is possible to specify as many +s options on the command line as you like. Use of the +s option will override the default list. For example, the command

```
freq +s"word" data.cut
```

will search through the file data.cut looking for "word."

The +s/-s switch is usually used to include or exclude certain words. However, it can actually be used with five types of material: (1) words, (2) codes or postcodes in square brackets, (3) text in angle brackets associated with particular codes within square brackets, (4) whole utterances associated with particular postcodes, and (5) particular postcodes themselves. The effect of the switch for the five different types is as follows.

Table 4: Search Strings for Five Types of Material

Level	+s	-s	+s+
Word	+s"dog"	-s"dog"	+s+xxx
	only the word "dog"	all words except "dog"	all words plus "dog"
[code]	+s"[//]"	by default, all codes are excluded	+s+"[//]"
	only this code		all text plus this code
<text>[x]	+s"<///>"	-s"<///>"	+s+"<///>"
	only text marked by this code	all text except material marked by this code	all text plus material marked by this code
Utterance	+s"<+imi>"	-s"<+imi>"	by default, all utterances are included

	utterances marked with this postcode	utterances not marked with this postcode	
Postcode	+s"[+imi]"	by default, all postcodes are excluded	+s"[+imi]"
	only this postcode		all text plus postcode

Multiple +s strings are matched as exclusive or's. If a string matches one +s string, it cannot match the other. The most specific matches are processed first. For example, if your command is

```
freq +s$gf% +s$gf:a
```

and your text has these codes

```
$gf $gf:a $gf:b $gf:c
```

your output will be

```
$gf%          3
$gf           1
```

Because \$gf:a matches specifically to the +s\$gf:a, it is excluded from matching +s\$gf%.

One can also use the +s switch to remove certain strings from automatic exclusion. For example, the MLU program automatically excludes xxx, 0, uh, and words beginning with & from the MLU count. This can be changed by using this command:

```
mlu +s+uh +s+xxx +s+0* +s+&* file.cha
```

6.6 +T Option

This option allows you to include or exclude particular tiers. In CHAT formatted files, there exist three tier code types: main speaker tiers (denoted by *), speaker-dependent tiers (denoted by %), and header tiers (denoted by @). The speaker-dependent tiers are attached to speaker tiers. If, for example, you request to analyze the speaker *MOT and all the %cod dependent tiers, the programs will analyze all of the *MOT main tiers and only the %cod dependent tiers associated with that speaker.

The +t option allows you to specify which main speaker tiers, their dependent tiers, and header tiers should be included in the analysis. All other tiers, found in the given file, will be ignored by the program. For example, the command

```
freq +t*CHI +t%spa +t%mor +t"@Group of Mot" sample.cha
```

tells FREQ to look at only the *CHI main speaker tiers, their %spa and %mor dependent tiers, and @Situation header tiers. When tiers are included, the analysis will be done on only those specified tiers.

The -t option allows you to specify which main speaker tiers, their dependent tiers, and header tiers should be excluded from the analysis. All other tiers found in the given file should be included in the analysis, unless specified otherwise by default. The command

```
freq -t*CHI -t%spa -t%mor -t"@Group of Mot" sample.cha
```

tells FREQ to exclude all the *CHI main speaker tiers together with all their dependent tiers, the %spa and %mor dependent tiers on all other speakers, and all @Situation header tiers from the analysis. All remaining tiers will be included in the analysis.

When the transcriber has decided to use complex combinations of codes for speaker IDs such as *CHI-MOT for “child addressing mother,” it is possible to use the +t switch with the # symbol as a wildcard, as in these commands:

```
freq +t*CHI-MOT sample.cha
freq +t*#-MOT sample.cha
freq +t*CHI-# sample.cha
```

When tiers are included, the analysis will be done on only those specified tiers. When tiers are excluded, however, the analysis is done on tiers other than those specified. Failure to exclude all unnecessary tiers will cause the programs to produce distorted results. Therefore, it is safer to include tiers in analyses than to exclude them, because it is often difficult to be aware of all the tiers present in any given data file.

If only a tier-type symbol (*, %, @) is specified following the +t/-t options, the programs will include all tiers of that particular symbol type in the analysis. Using the option +t@ is important when using KWAL for limiting (see the description of the KWAL program), because it makes sure that the header information is not lost.

The programs search sequentially, starting from the left of the tier code descriptor, for exactly what the user has specified. This means that a match can occur wherever what has been specified has been found. If you specify *M on the command line after the option, the program will successfully match all speaker tiers that start with *M, such as *MAR, *MIK, *MOT, and so forth. For full clarity, it is best to specify the full tier name after the +t/-t options, including the : character. For example, to ensure that only the *MOT speaker tiers are included in the analysis, use the +t*MOT: notation.

As an alternative to specifying speaker names through letter codes, you can use the form:

```
+t@id=idcode
```

In this form, the “idcode” is any character string that matches the type of string that has been declared at the top of each file using the @ID header tier. The basic form of this code is language.corpus.file.age=XXX where XXX is the participant code.

All of the programs include the main speaker tiers by default and exclude all of the dependent tiers, unless a +t% switch is used.

6.7 +U Option

This option merges specified files together. By default, when the user has specified a series of files on the command line, the analysis is performed on each individual file. The program then provides separate output for each data file. If the command line uses the +u option, the program combines the data found in all the specified files into one set and outputs the result for that set as a whole. If too many files are selected, CLAN may eventually be unable to complete this merger.

6.8 +V Option

This switch gives you the date when the current version of CLAN was compiled.

6.9 +W Option

This option controls the printing of additional sentences before and after a matched sentence. This option can be used with either KWAL or COMBO. These programs are used to display tiers that contain keywords or regular expressions as chosen by the user. By default, KWAL and COMBO combine the user-chosen main and dependent tiers into “clusters.” Each cluster includes the main tier and its dependent tiers. (See the +u option for further information on clusters.)

The -w option followed by a positive integer causes the program to display that number of clusters before each cluster of interest. The +w option followed by a positive integer causes the program to display that number of clusters after each cluster of interest. For example, if you wanted the KWAL program to produce a context larger than a single cluster, you could include the -w3 and +w2 options in the command line. The program would then output three clusters above and two clusters below each cluster of interest.

6.10 +Y Option

This option allows you to work on non-CHAT files. Most of the programs are designed to work best on CHAT formatted data files. However, the +y option allows the user to use these programs on non-CHAT files. The program considers each line of a non-CHAT file to be one tier. There are two values of the +y switch. The +y value works on lines and the +y1 value works on utterances as delimited by periods, question marks, and exclamation marks. Some programs do not allow the use of the +y option at all. Workers interested in using CLAN with nonconversational data may wish to first convert these files to CHAT format using the TEXTIN program described on page [126](#) in order to avoid having to avoid the problematic use of the +y option.

6.11 +Z Option

This option allows the user to select any range of words, utterances, or speaker turns to be analyzed. The range specifications should immediately follow the option. For example:

+z10w	analyze the first ten words only.
+z10u	analyze the first ten utterances only.
+z10t	analyze the first ten speaker turns only.
+z10w-20w	analyze 11 words starting with the 10th word.
+z10u-20u	analyze 11 utterances starting with the 10th utterance.
+z10t-20t	analyze 11 speaker turns starting with the 10th turn.
+z10w-	analyze from the tenth word to the end of file.
+z10u-	analyze from the tenth utterance to the end of file.
+z10t-	analyze from the tenth speaker turn to the end of file.

If the +z option is used together with the +t option to select utterances from a particular speaker, then the counting will be based only on the utterances of that speaker. For example, this command

```
mlu +z50u +t*CHI 0611.cha
```

will compute the MLU for the first 50 utterances produced by the child. If the +z option

is used together with the +s option, the counting will be dependent on the working of the +s option and the results will seldom be as expected. To avoid this problem, you should use piping, as in this example:

```
kwal +d +z1-3u +t*CHI sample.cha | kwal +sMommy
```

If the user has specified more items than exist in the file, the program will analyze only the existing items. If the turn or utterance happens to be empty, because it consists of special symbols or words that have been selected to be excluded, then this utterance or turn is not counted.

The usual reason for selecting a fixed number of utterances is to derive samples that are comparable across sessions or across children. Often researchers have found that samples of 50 utterances provide almost as much information as samples of 100 utterances. Reducing the number of utterances being transcribed is important for clinicians who have been assigned a heavy case load.

You can use the +z switch with KWAL and pipe the results to a second program, rather than using it directly with FREQ or MLU. For example, in order to specifically exclude unintelligible utterances in an MLU analysis of the first 150 utterances from the Target Child, you could use this form:

```
kwal +z150u +d +t*CHI 0042.cha -syyy -sxxx | mlu
```

You can also use postcodes to further control the process of inclusion or exclusion.

6.12 Metacharacters for Searching

Metacharacters are special characters used to describe other characters or groups of characters. Certain metacharacters may be used to modify search strings used by the +s/-s switch. However, in order to use metacharacters in the CHSTRING program a special switch must be set. The CLAN metacharacters are:

*	Any number of characters matched
%	Any number of characters matched and removed
%%	As above plus remove previous character
_	Any single character matched
\	Quote character

Suppose you would like to be able to find all occurrences of the word “cat” in a file. This includes the plural form “cats,” the possessives “cat-'s,” “cat-'s” and the contractions “cat-'is” and “cat-'has.” Using a metacharacter (in this case, the asterisk) would help you to find all of these without having to go through and individually specify each one. By inserting the string cat* into the include file or specifying it with +s option, all these forms would be found. Metacharacters can be placed anywhere in the word.

The * character is a wildcard character; it will find any character or group of con-

tinuous characters that correspond to its placement in the word. For example, if b*s were specified, the program would match words like “beads,” “bats,” “bat-'s,” “balls,” “beds,” “bed-s,” “breaks,” and so forth.

The % character allows the program to match characters in the same way that the * symbol does. Unlike the * symbol, however, all the characters matched by the % will be ignored in terms of the way of which the output is generated. In other words, the output will treat “beat” and “bat” as two occurrences of the same string, if the search string is b%t. Unless the % symbol is used with programs that produce a list of words matched by given keywords, the effect of the % symbol will be the same as the effect of the * symbol.

When the percentage symbol is immediately followed by a second percentage symbol, the effect of the metacharacter changes slightly. The result of such a search would be that the % symbol will be removed along with any one character preceding the matched string. Without adding the additional % character, a punctuation symbol preceding the wildcard string will not be matched and will be ignored. Adding the second % sign can be particularly useful when searching for roots of words only. For example, to produce a word frequency count of the stem “cat,” specify this command:

```
freq +s"cat-%%" file.cha.
```

The first % sign matches the suffixes and the second one matches the dash mark. Thus, the search string specified by the +s option will match words like: “cat,” “cat-s,” “cat-'s,” and “cat-s” and **FREQ** will count all of these words as one word “cat.” If the data file `file.cha` had consisted of only those four words, the output of the **FREQ** program would have been: 4 cat. The limitation of this search is that it will not match words like “cats” or “cat's,” because the second percentage symbol is used to match the punctuation mark. The second percentage symbol is also useful for matching hierarchical codes such as \$NIA:RP:IN.

The underline character `_` is similar to the * character except that it is used to specify any single character in a word. For example, the string `b_d` will match words like “bad,” “bed,” “bud,” “bid,” and so forth. For detailed examples of the use of the percentage, underline, and asterisk symbols, see the section on **FREQ** on page [76](#).

The quote character `\` is used to indicate the quotation of one of the characters being used as metacharacters. Suppose that you wanted to search for the actual symbol (*) in a text. Because the (*) symbol is used to represent any character, it must be quoted by inserting the `\` symbol before the (*) symbol in the search string to represent the actual (*) character, as in “string*string.” To search for the actual character `\`, it must be quoted also. For example, “string\\string” will match “string” followed by “\” and then followed by a second “string.”