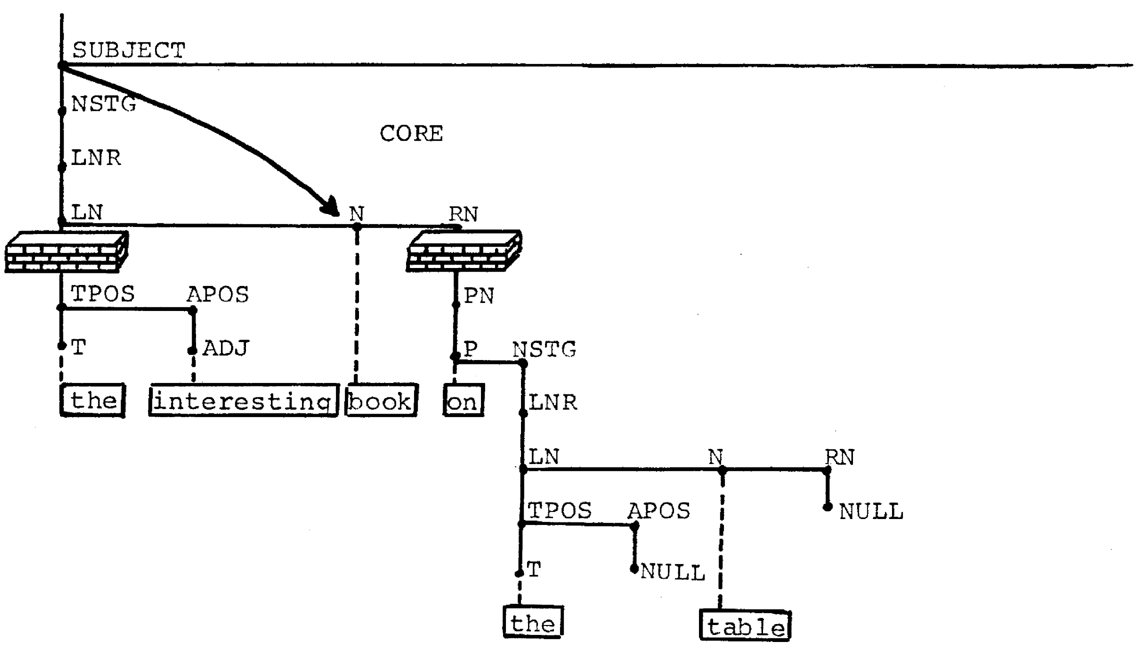
are not on the elements SUBJECT and VERB of ASSERTION but rather on some nodes labeled N and TV, several levels down in the parse tree. If we had to describe explicitly the level-by-level search required to find these nodes, the original objective of the restriction would be obscured by these details of tree motion; we would, so to speak, lose sight of the forest for the tree.

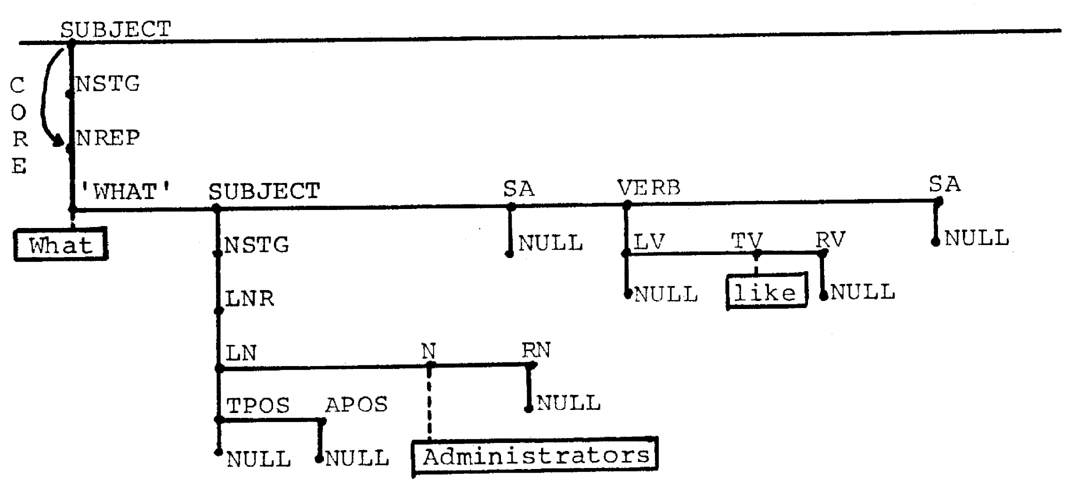
We therefore have in the restriction language an operation CORE which goes from an element of a string definition to the atomic node which corresponds to the word category in the original elementary string. Put another way, the CORE word is the one word subsumed by the element which remains after all adjunct strings have been deleted; e.g., the CORE word of "The interesting book on the table" is "book".

The set of adjunct string definitions is given by the TYPE ADJSET list (see § 2). The CORE of a given node is the one atomic node in the subtree below the starting node which is not also below a node in the subtree which is on the ADJSET list.



In effect, the ADJSET nodes block the CORE operation from looking further down the tree. (To make the definition of CORE more precise, we should note that the CORE operation performs a "breadth-first" search; that is, it first searches all the nodes on one level from left to right for an atomic, then goes down to the next level, being sure not to go below any ADJSET nodes. This means that if there were several atomic nodes not in adjuncts, CORE would take the one highest in the tree, and among several at the same level it would take the leftmost. However, we will always use the CORE operation starting from an element in a string definition, so in our grammar there will be only one node fitting the criteria given above).

As we mentioned in § 1, it is possible in string theory to replace a category in one string by certain other strings. For instance, the first N in the center string N TV N can be replaced by 'WHAT' N TV to produce a sentence like "What administrators like are lengthy forms". This particular replacement is



provided for by the NREP option of NSTG in our small grammar. In this case the CORE OF SUBJECT is not one word, but the string "What administrators like". To take this into account, the CORE operation searches down the tree for either an atomic or a string node (i.e., one on the TYPE STRING list). Thus, as indicated at right, the CORE OF SUBJECT will be the node NREP.

With all this in mind, we can consider the effect of the restriction

WSSING = IN ASSERTION: CORE OF SUBJECT IS SINGULAR.

If the SUBJECT were an LNR, as in the example on page 48 , the CORE operation would descend to the N node and the restriction predicate would then test if the corresponding word had the attribute SINGULAR. Similarly, if the SUBJECT were an LPROR, the CORE would descend to the PRO and the attribute of the pronoun would be tested. If the SUBJECT were an NREP, however, as in the example on this page, the CORE operation would leave us at the NREP; the restriction predicate, which looks for an attribute, would then fail immediately, since we are not at an atomic node. Thus WSSING allows as sentence subjects only SINGULAR nouns and pronouns (with their adjuncts), and no replacement strings. In contrast,

WSNOTPL = IN ASSERTION: CORE OF SUBJECT IS NOT PLURAL.

would allow both nouns and pronouns which are not PLURAL and replacement strings.

Armed with our new restriction subject, CORE OF node, we are ready to write our first linguistically meaningful restriction. As was noted in passing in §1, there is a severe restriction on the sorts of verbs which can appear in an NREP string:

∃ "What I write is tripe." but ∄ "What I smile is tripe." In terms of our grammar, this restriction can be formulated as: a verb can appear in an NREP string only if it can appear in an ASSERTION string with an NSTG object. (∃ "I write tomes." but ∄ "I smile tomes.") Now we mentioned in §4 that the options of OBJECT with which a particular verb may appear are given in the word dictionary as a list beneath the attribute OBJLIST. Our restriction on NREP is therefore

WWH = IN NREP: CORE OF VERB HAS ATTRIBUTE OBJLIST: NSTG.

Another restriction which we can write now concerns pronouns appearing as the subject of a sentence. ACCUSATIVE pronouns are not allowed in this position (∃ "I write tomes." but ∄ "Me write tomes."). In restriction language, this can be stated as:

WPRO = IN SUBJECT: THE CORE OF THE SUBJECT IS NOT PRO: ACCUSATIVE.

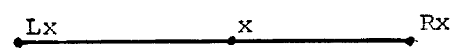
You may note the rather cavalier fashion in which we insert or omit the articles A, AN, and THE in writing restrictions. These words are purely window-dressing, which we add to make the restriction read better; they are entirely ignored by the restriction language compiler (except in the special case where they appear in quotes: THE VALUE OF TPOS IS 'THE'). We could therefore have written WPRO just as well as

WPRO = IN SUBJECT: CORE OF SUBJECT IS NOT PRO: ACCUSATIVE.

In short, the restriction language compiler, like most New York restaurants, is not concerned about missing articles.

A principal relation in the string grammar is between an element of a string and its left and right adjuncts. Corresponding to this relation in the string grammar are three operations in the restriction language: LEFT-ADJUNCT*, RIGHT-ADJUNCT, and HOST (which goes from either adjunct to the element it adjoins). With one exception, these operations offer core-to-core service; that is, they expect to start at the core of the host and bring you to the core of the adjunct, or vice versa.

These operations take advantage of the uniform structure of the LXR (host-adjunct) definitions in the grammar. In each case (LNR, LPROR and LTVR)



the definition has three elements: the left adjunct, the atom, and the right adjunct. LEFT-ADJUNCT, starting at x, goes left to Lx (more precisely, to a node on the TYPE LADJSET list), and then descends to the core. Similarly, RIGHT-ADJUNCT, starting at x, goes right to Rx (a node of TYPE RADJSET), and then descends to the core. HOST goes up the tree until it finds an Lx or Rx, and then goes right or left respectively, to the atomic node.

Suppose we felt like excluding "hardly" as a right adjunct of the verb. By now we are able to write quite a few different restrictions to accomplish this. The simplest would be

WHARDLY1 = IN RV: VALUE OF RV IS NOT 'HARDLY'.

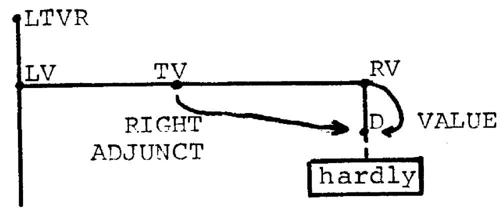
(note that we could not have written D IS NOT 'HARDLY' since this would fail if the value of RV were PN). If we wanted to slow the parser down, we could house the restriction one level higher, in LTVR:

WHARDLY2 = IN LTVR: VALUE OF RV IS NOT 'HARDLY'.

---

*No blanks are allowed around the hyphen in LEFT-ADJUNCT and RIGHT-ADJUNCT.

Starting from LTVR, there is another way to get to D: go to TV and then use the RIGHT-ADJUNCT operation:



WHARDLY3 = IN LTVR: RIGHT-ADJUNCT OF TV IS NOT ↓HARDLY↓.
If the right adjunct is a PN, the RIGHT-ADJUNCT operation will bring you to the PN node (since the CORE operation stops at a TYPE STRING node).

The one exception to the core-to-core rule arises in going to the LEFT-ADJUNCT of N. Because LN has several elements, it isn't clear which one to go to -- the core of TPOS or the core of APOS. Therefore, in this one case, the LEFT-ADJUNCT operation stops at the LN node.

Thus if, by some strange whimsey, we felt that the only good OBJECT is an NSTG containing an adjective, we could write a restriction which, starting at OBJECT,

1. descended to the CORE
2. went left to the LEFT-ADJUNCT
3. went down to APOS
4. checked that it (i.e., APOS) was not empty

Such a restriction would rule out all options of OBJECT except NSTG and all options of NSTG except LNR. We can combine steps 1 and 2 in the restriction language construct LEFT-ADJUNCT OF CORE OF OBJECT (in general, operations can be nested this way, using several OF's). We have no operation in our repertoire at the moment, however, for step 3. We know, though, that if the restriction <u>began</u> at LN we could accomplish step 3 simply by making APOS the subject of our restriction language statement, e.g.,

APOS IS NOT EMPTY

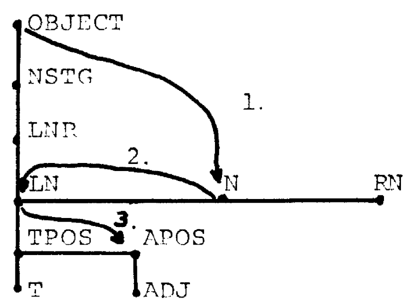
To put the whole thing together, we place steps 1 and 2 in an <u>introducer</u>:

WOA = IN ASSERTION:

IN LEFT-ADJUNCT OF CORE OF OBJECT,

APOS IS NOT EMPTY.

An introducer is simply the word IN, followed by some construct which could also appear as a restriction language subject, followed by a comma. The restriction first goes to the point specified by the introducer and then performs the test indicated by the subject and predicate. If the introducer is unable to find the specified point (e.g., in this case, if the value of OBJECT were THATS, which has no left adjunct) the restriction fails.



Two other operations, ELEMENT and COELEMENT, bear mention at this point. ELEMENT is an extension of the operation performed when the subject of a restriction sentence is simply the name of a node. In general, ELEMENT a OF b causes the restriction to go to b and then to look on the level below b for a node named a. In particular, OF b can be omitted, in which case the operation begins at the starting node of the restriction (this is true of all the operations discussed so far). Many of the restrictions we have written thus far with the name of a node as subject could therefore be rewritten using ELEMENT; for example,

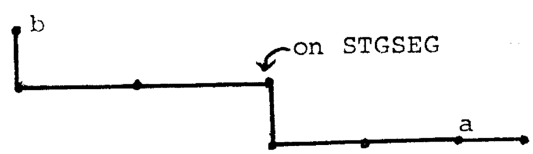
WT1 = IN ASSERTION: ELEMENT OBJECT IS EMPTY.

WOA = IN ASSERTION:

IN LEFT ADJUNCT OF CORE OF OBJECT,

ELEMENT APOS IS NOT EMPTY.

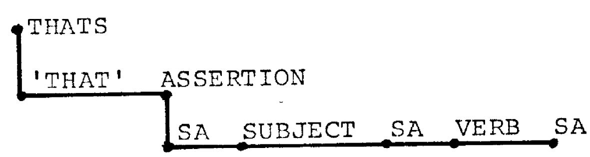
This, however, is pointless. The important additional fact about ELEMENT a OF b is that, if a node on the level below b is on the STGSEG (string segment) list, the restriction searches the level below that node as well for a node called a.



For instance, the restriction

WSTGSEG = IN THATS: ELEMENT VERB IS NOT EMPTY.

will succeed because ELEMENT will "look through" ASSERTION (on the STGSEG list) to find VERB. This is in accord with

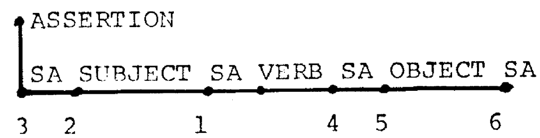


the information we intended to convey by placing ASSERTION on the STGSEG list, namely, that ASSERTION is used in certain contexts in place of writing out the sequence of elements <SA> <SUBJECT><SA><VERB><SA><OBJECT><SA>. In other words, linguistically we would like to consider THAT, the core of SUBJECT, the core of VERB, and the core of OBJECT as being elements of a single noun-replacement string THATS.

In string theory, two elements of a single string are said to be coelements of each other. In the restriction language, the operation COELEMENT a searches the current level of the tree for a node named a. Thus, if we are in the ASSERTION string at the node VERB, we could refer to the COELEMENT SUBJECT or the COELEMENT OBJECT; for example, WT1 could be written

WT1 = IN ASSERTION: COELEMENT OBJECT OF VERB IS EMPTY.

COELEMENT searches first to the left from the starting node and then to the right (the order of search, starting at VERB, is shown in the figure), so COELEMENT SA OF VERB would leave you at the SA immediately to the left of the VERB. COELEMENT, like ELEMENT, will "look through" a node on the STGSEG list. Starting at 'THAT', the first element of THATS, we may therefore proceed to COELEMENT SUBJECT or COELEMENT VERB.



To recap the last two sections, we present a BNF grammar of the restriction language we have learned so far:

<statement> ::= <introducer><subject><predicate>

<subject> ::= <*node>|
CORE [OF<subject>]|
LEFT-ADJUNCT [OF<subject>]|
RIGHT-ADJUNCT [OF <subject>]|
HOST [OF <subject>]|
ELEMENT <*node> [OF <subject>]|
COELEMENT <*node> [OF <subject>]|
VALUE [OF <subject>].

```
<predicate>  ::= IS <bepred>|IS NOT <bepred>|
                 HAS <havepred>|DOES NOT HAVE <havepred>.
<bepred>     ::= EMPTY|
                 <*node>|
                 <*text>|
                 <attribute list>.
<havepred>   ::= VALUE <*node>|
                 ATTRIBUTE <attribute list>.
<attribute list> ::= <*attribute>|<*attribute>':'<attribute list>.
<introducer>::= IN<subject>','|<*null>.
```

The square brackets used in the definition of <subject> indicate that OF <subject> is optional after any of the operations; if it is not present, the operation begins at the starting node of the restriction. <*node> is the name of any BNF definition or atomic symbol in the English grammar, and <*attribute> is any name appearing on the ATTRIBUTE list. <*text> is any sequence of characters appearing between quote marks ('...').

8. THE RESTRICTIONS: IFs, ANDs, and ORs

We have obviously been hard pressed until now in trying to create meaningful restrictions. We have been able to test only one condition in each of our restrictions, whereas most "real" restrictions can only be stated as a combination of two, three, or more conditions. We shall overcome this limitation in this section by introducing the logical connectives.

Given any two restriction language statements p and q, such as those we wrote above, we may combine them to form the new statements

IF p THEN q

BOTH p AND q

EITHER p OR q

NEITHER p NOR q

we shall consider each of these logical forms in turn.

A restriction of the form

IF p THEN q

will be executed as follows:

1. execute p
2. if p fails, the restriction succeeds (q is not executed)
3. return to the point the restriction was looking at before it executed p
4. execute q
5. if q succeeds, the restriction succeeds
   if q fails, the restriction fails

This construction is therefore used to test one condition if another condition is true. For example, a large class of nouns in English must be preceded by a determiner. (e.g., "box": $\exists$"The box is red." and $\exists$"My box is red." but $\nexists$"Box is red."). Such nouns have the attribute NCOUNT in our grammar. We should therefore include a restriction to the effect that, if a noun has the attribute NCOUNT, the TPOS position of its left adjunct is filled:

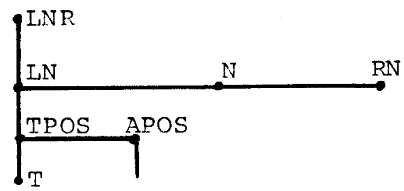
WCOUNT = IN LNR:

IF THE CORE IS N: NCOUNT

THEN IN THE LEFT-ADJUNCT OF THE CORE,

TPOS IS NOT EMPTY.

Keep in mind that, if the core is NCOUNT, the restriction goes back to LNR before starting on



the second half (IN THE LEFT-ADJUNCT OF THE CORE...).

One other constraint on our grammar which we can now attack is the "warhorse" of restrictions, subject-verb agreement in number. In our grammar the constraint must be that, if the verb is PLURAL the subject is not SINGULAR, and, if the verb is SINGULAR the subject is not PLURAL. We require this negative formulation for our grammar because we have chosen to assign neither the attribute SINGULAR nor PLURAL to nouns which can appear with either a SINGULAR or PLURAL verb (e.g., the noun "fish"). This constraint can evidently be written as two restrictions, each of them an IF... THEN... construction:

WAGREE1 = IN ASSERTION:

IF THE CORE OF THE VERB IS PLURAL

THEN THE CORE OF THE SUBJECT IS NOT SINGULAR.

WAGREE2 = IN ASSERTION:

IF THE CORE OF THE VERB IS SINGULAR

THEN THE CORE OF THE SUBJECT IS NOT PLURAL.

This is not quite enough: we must also check that a noun-replacement string subject is not used with a plural verb

(∃"What I say appears confusing." ∄"What I say appear confusing."). This calls for a third restriction:

```
WAGREE3 = IN ASSERTION:
    IF THE CORE OF THE VERB IS PLURAL
    THEN THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
```

For this restriction we have used a new predicate, IS OF TYPE *type*, where *type* is the name of one of the TYPE lists in the grammar. This predicate tests whether the current node appears in the TYPE list specified.

Things seem to be getting a bit out of hand, what with three restrictions required already for one linguistic constraint. We can consolidate these restrictions by introducing the BOTH... AND... construction. The restriction statement BOTH p AND q is executed by

1. executing p
2. if p fails, the entire statement fails (q is not executed)
3. returning to the point the restriction was looking at before executing p
4. executing q
5. if q succeeds, the entire restriction statement succeeds; if q fails, the statement fails

These logical constructions can be nested; that is, p and q can be not only simple (subject-predicate) sentences, but also logical forms themselves. We can use one dose of BOTH... AND... to combine WAGREE1 and WAGREE3: