```
WAGREE13 = IN ASSERTION:
   IF THE CORE OF THE VERB IS PLURAL
   THEN BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
        AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
```

this saves us from testing CORE OF THE VERB IS PLURAL twice. One more use of BOTH... AND... can combine WAGREE13 and WAGREE2 into a single restriction:

```
WAGREE = IN ASSERTION:
BOTH IF THE CORE OF THE VERB IS PLURAL
     THEN BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
          AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING
AND IF THE CORE OF THE VERB IS SINGULAR
     THEN THE CORE OF THE SUBJECT IS NOT PLURAL.
```

Now that we've learned how to put the whole thing together into one restriction, we shall see how we can break it up into several parts. Combining the several restrictions into one has clarified the logical relationships among the various tests, but has left us with a rather unpleasantly long single statement; further refinements will make the restriction even more unwieldy. We may therefore take any portion of the restriction which is a statement itself, assign it a name, and replace it in the body of the restriction by that name. For example, WAGREE may be rewritten as

```
WAGREE = IN ASSERTION:
   BOTH $PLURAGREE AND $SINGAGREE.
$PLURAGREE = IF THE CORE OF THE VERB IS PLURAL
   THEN BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
        AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
```

$SINGAGREE = IF THE CORE OF THE VERB IS SINGULAR
THEN THE CORE OF THE SUBJECT IS NOT PLURAL.

We call the two pieces which we have broken off the restriction *substatements* or address sentences. These substatements must follow the restriction and, like all statements in the grammar, must end in a period. The names assigned to these substatements (also called their "addresses") must consist of a dollar sign followed by one through nineteen letters, digits, or hyphens. Substatement names are local to the restriction in which they appear; in other words, the same substatement name may be used in several restrictions without any conflict.*

Our WAGREE does not yet take into account a number of more unusual cases. For instance, a number of words which are clearly SINGULAR according to our "this/these" criterion (p. 5) can appear with both singular and plural verbs:

half (∃ This half exists. ∄ These half exist(s).)
∃ Half of the ice is melted.
∃ Half of the apples are green.

group (∃ This group met. ∄ These group met.)
∃ My group is on the bus.
∃ The group near the elephant are former Democrats.

We assign such words the attribute AGGREGATE:

HALF N: (SINGULAR, AGGREGATE).
GROUP N: (SINGULAR, AGGREGATE),V:(OBJLIST:(NSTG)),
TV: (PLURAL, OBJLIST: (NSTG)).

In WAGREE, $PLURAGREE must then be modified so that it will not

*A means for sharing a substatement among several restrictions will be described later.

fail if the core of the SUBJECT has the attribute AGGREGATE. We can do this with an EITHER... OR construction

```
$PLURAGREE = IF THE CORE OF THE VERB IS PLURAL
   THEN EITHER THE CORE OF THE SUBJECT IS AGGREGATE
      OR BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
          AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
```

Since this is getting a bit long again, we can break it up:

```
$PLURAGREE = IF THE CORE OF THE VERB IS PLURAL
  THEN EITHER THE CORE OF THE SUBJECT IS AGGREGATE
            OR $NOTSINGORSTRING.
$NOTSINGORSTRING =
  BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
    AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
```

Formally, the restriction statement EITHER p OR q is executed as follows:

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

There are relatively few examples of the fourth logical form, NEITHER...NOR..., in the English grammar. It just so happens, however, that $NOTSINGORSTRING is a perfect candidate for being rewritten as a NEITHER... NOR...:

```
$NOTSINGORSTRING =
  NEITHER THE CORE OF THE SUBJECT IS SINGULAR
     NOR THE CORE OF THE SUBJECT IS OF TYPE STRING.
```

This may not be great style, but it's the content that counts. To complete our Hoyle of logical form definitions, we present the rules for executing NEITHER p NOR q

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

The last three logical operations (all except IF... THEN...) can be naturally extended to connect three or more statements. The forms used to connect three or more statements, however, are quite different from those described above:

i) BOTH...AND... generalizes to

   ALL OF $A, $B, $C ARE TRUE

   which succeeds only if all of the listed substatements succeed

ii) EITHER... OR generalizes to

   ONE OF $A, $B, $C IS TRUE

   which succeeds if any one of the listed substatements succeeds

iii) NEITHER... NOR... generalizes to

   NONE OF $A, $B, $C IS TRUE

which succeeds if none of the listed substatements succeeds. In the above $A, $B, $C may be replaced by any sequence of substatement names separated by commas. In contrast to the binary operations described earlier, however, statements may not appear directly in these forms, only substatement names.

9. THE RESTRICTIONS: REGISTERS

In the previous chapter we developed a rather substantial (though by no means complete) restriction for checking number agreement between subject and verb. Collecting the pieces, we have

```
WAGREE = IN ASSERTION:
   BOTH $PLURAGREE AND $SINGAGREE.
$PLURAGREE =
   IF THE CORE OF THE VERB IS PLURAL
   THEN EITHER THE CORE OF THE SUBJECT IS AGGREGATE
         OR $NOTSINGORSTRING.
$NOTSINGORSTRING =
   BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
   AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
$SINGAGREE =
   IF THE CORE OF THE VERB IS SINGULAR
   THEN THE CORE OF THE SUBJECT IS NOT PLURAL.
```

One striking fact is that CORE OF THE VERB appears twice and CORE OF THE SUBJECT four times in this restriction. This means that both COREs will usually be computed several times during each execution of the restriction. If we want to make this restriction more efficient (i.e., take less time to execute), we should record the places in the tree corresponding to CORE OF SUBJECT and CORE OF VERB the first time they are computed, rather than computing them anew each time. The restriction language provides such place holders in the form of registers, which may point to any node in the parse or sentence tree.

These registers mimic a function you would naturally perform when going through the restriction by hand. If you were trying to see whether WAGREE was true in a particular tree, you probably wouldn't try to locate the CORE OF THE SUBJECT over and over again, starting from ASSERTION. Instead, the first time you found it you would put a finger on the node (or a pebble, or whatever); when you had to refer to the CORE OF THE SUBJECT again, you would simply look at the node your finger was pointing to, without repeating the search. Registers perform the same function for the machine as your finger does for you. In fact, we refer implicitly to the analogy when we say that the register "points to" the node.

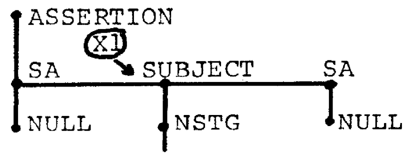
Before we examine the rules for registers in detail, let us consider one simple example to give us the flavor of the mechanism. According to the description given in the previous chapter, the following restriction should always succeed:

```
WALWAYS = IN ASSERTION:
IF THE CORE OF THE SUBJECT IS AGGREGATE
THEN THE CORE OF THE SUBJECT IS SINGULAR.
```

Since this restriction will always succeed, we want to waste as little time as possible executing it. We can save the time required to search the level below ASSERTION a second time for SUBJECT by saving it the first time in a register:

IF THE CORE OF THE SUBJECT X1 IS AGGREGATE
THEN THE CORE OF X1 IS SINGULAR.

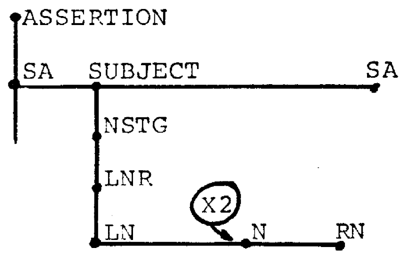
We have associated the register name, "X1", with the node SUBJECT by writing it immediately after the node name in the first restriction sentence; in CORE OF X1 the operation CORE will therefore start at the SUBJECT.



Even more time can be saved by recording the CORE in a register:

IF THE CORE X2 OF THE SUBJECT IS AGGREGATE
THEN X2 IS SINGULAR.

Writing the register name, "X2", immediately after CORE associates the register with the node reached by the CORE operation; the second restriction sentence thus immediately starts out at the core node and need only test its attribute.



Now for the rules: the name of a register must consist of the letter X followed by 1 through 19 digits, such as X1, X2, or X987265492817632845. A register can be assigned a value (i.e., associated with a point in the parse or sentence tree) by placing its name

1) after the name of a node appearing as the sole or last item in a restriction subject:

SUBJECT X1 HAS VALUE NSTG

CORE OF VERB X4 IS PLURAL

the value of the register is the node whose name it follows

2) after any operation (VALUE, CORE, HOST,...) appearing in the subject of a restriction sentence; if the specification of the operation includes the name of a node (ELEMENT OBJECT, COELEMENT VERB), the ~~[illegible]~~ register name must follow the node name:

CORE X2 OF SUBJECT IS SINGULAR

COELEMENT VERB X5 OF OBJECT HAS VALUE LTVR

the value assigned to the register is the node reached by the operation

3) after any positive predicate (one not containing the word NOT):

THE CORE OF THE SUBJECT IS N X1

THE VALUE OF THE OBJECT IS NSTG X2

the value assigned to the register is the point in the tree which the restriction is looking at when the predicate is finished.

The assignment to the register will take place only if all the operations in the restriction sentence which precede the assignment have been successfully performed. For example, if the CORE OF THE SUBJECT is N, then both CORE X4 OF THE SUBJECT IS N and CORE OF THE SUBJECT IS N X4 will assign the node N beneath SUBJECT to X4. If, however, the CORE OF THE SUBJECT were PRO, the sentence CORE X4 OF THE SUBJECT IS N would assign the core (the node PRO) to X4, whereas CORE OF THE SUBJECT IS N X4 would not assign any value to X4, since the assignment follows the test "IS N", which would fail. In the last case we say that the value of the register remains undefined.

All registers are initially undefined when a restriction begins execution; it is not possible to pass information from one restriction to another through a register. A register may be assigned more than one value in the course of a ~~register~~ restriction; the value of a register is then the last value assigned to it.

Once a register has been assigned a value, it may be used in place of the name of a node at several possible points in a restriction sentence, including:

1) as the last or sole item in a restriction subject

   X1 IS SINGULAR

   CORE OF X1 IS SINGULAR

   the restriction subject looks at (or begins by looking at) the node associated with the register

2) in the predicate, IS register

   CORE OF SUBJECT IS X1

   this predicate tests whether the current node has the same name as the node associated with the register

3) in the predicate HAS ATTRIBUTE register

   or HAS ATTRIBUTE attribute: register

   looks for an attribute whose name is the name of the node associated with the register

   CORE OF OBJECT HAS ATTRIBUTE X7

   CORE OF VERB HAS ATTRIBUTE OBJLIST: X1

If the register is currently undefined, any reference to (use of) the register will cause the restriction sentence to fail.

Now that we have gotten the rules under our belts (or wherever you chose to put them), let us return to the WAGREE restriction with which we began this chapter. To speed up this restriction, we shall first save the CORE OF THE VERB in X1:

```
     WAGREE = IN ASSERTION:  BOTH $PLURAGREE AND $SINGAGREE.
     $PLURAGREE = IF THE CORE X1 OF THE VERB IS PLURAL
                THEN EITHER THE CORE OF THE SUBJECT IS AGGREGATE
                   OR $NOTSINGORSTRING.
     $NOTSINGORSTRING = BOTH THE CORE OF THE SUBJECT IS NOT SINGULAR
                AND THE CORE OF THE SUBJECT IS NOT OF TYPE STRING.
     $SINGAGREE = IF X1 IS SINGULAR
                   THEN THE CORE OF THE SUBJECT IS NOT PLURAL.
```

Next, we would like to save the CORE OF THE SUBJECT in X2:

```
     WAGREE = IN ASSERTION:  BOTH $PLURAGREE AND $SINGAGREE.
     $PLURAGREE = IF THE CORE X1 OF THE VERB IS PLURAL
         THEN EITHER THE CORE X2 OF THE SUBJECT IS AGGREGATE
              OR $NOTSINGORSTRING.
     $NOTSINGORSTRING = BOTH X2 IS NOT SINGULAR
                         AND X2 IS NOT OF TYPE STRING.
     $SINGAGREE = IF X1 IS SINGULAR THEN X2 IS NOT PLURAL.
```

The restriction certainly is a lot shorter now. Only one problem -- it doesn't work any more (as you will discover if you try to run it). The trouble arises when the CORE OF THE VERB is SINGULAR; the portion of $PLURAGREE following the "THEN" is then skipped, according to the rules for IF... THEN... . Consequently, when $SINGAGREE is executed, X2 is undefined; therefore both $SINGAGREE and the entire restriction will fail.

If we only want to compute the CORE OF THE SUBJECT once, we must place the assignment to X2 where it will always be executed. We can do this by creating a new restriction statement whose sole function is to put the CORE OF THE SUBJECT in X2, and making this the first statement executed in the restriction:

```
    WAGREE = IN ASSERTION:  ALL OF $FINDSUBJECT,
        $PLURAGREE, $SINGAGREE ARE TRUE.
    $FINDSUBJECT = CORE X2 OF THE SUBJECT EXISTS.
    $PLURAGREE = IF THE CORE X1 OF THE VERB IS PLURAL
        THEN EITHER X2 IS AGGREGATE OR $NOTSINGORSTRING.
    $NOTSINGORSTRING = BOTH X2 IS NOT SINGULAR
                       AND X2 IS NOT OF TYPE STRING.
    $SINGAGREE = IF X1 IS SINGULAR THEN X2 IS NOT PLURAL.
```

The predicate "EXISTS" actually doesn't do anything -- the restriction statement so-and-so EXISTS will succeed if the restriction subject, so-and-so, succeeds and fail if the restriction subject fails. In this case, we expect that we will always find a CORE OF THE SUBJECT, so $FINDSUBJECT will always succeed; its only job is to put that node in X2.

We will now turn our attention to a restriction which could not have been written without registers. You may recall that we assigned to each verb in the word dictionary an attribute OBJLIST, followed by a list of the options of OBJECT which may appear with this verb; for instance,

```
    STRETCHES TV:  (SINGULAR, OBJLIST:(NSTG,NULLOBJ)).
```

The restriction which uses this information will have to

1. find the name of the node below OBJECT
2. find the core of the VERB
3. check that the name found in 1. appears below the attribute OBJLIST of the node found in 2.

Because the attribute to be checked for is variable (the name of the node below OBJECT), we are forced to use a register to save the node in step 1. Assuming that the restriction begins at OBJECT, and that we save the node in X1, step 1. becomes

```
        VALUE X1 OF OBJECT EXISTS
```

Since VERB is on the same level as OBJECT, step 2. is taken care of by the restriction subject

```
        CORE OF COELEMENT VERB
```

For step 3. we must first go to the OBJLIST attribute of the core (TV) and then check if the OBJLIST has as attribute the name of the node in X1:

```
    CORE OF COELEMENT VERB HAS ATTRIBUTE
                                      OBJLIST:X1.
```

The first sentence (VALUE... EXISTS) should always succeed, and the success of the restriction should depend on the success of the second sentence, so we may reasonably join these two sentences with BOTH... AND... or IF... THEN... . Choosing the latter form and adding the trimmings, we have

```
    WVERBOBJ = IN OBJECT:
        IF VALUE X1 OF OBJECT EXISTS
      THEN CORE OF COELEMENT VERB
                              HAS ATTRIBUTE OBJLIST:  X1.
```