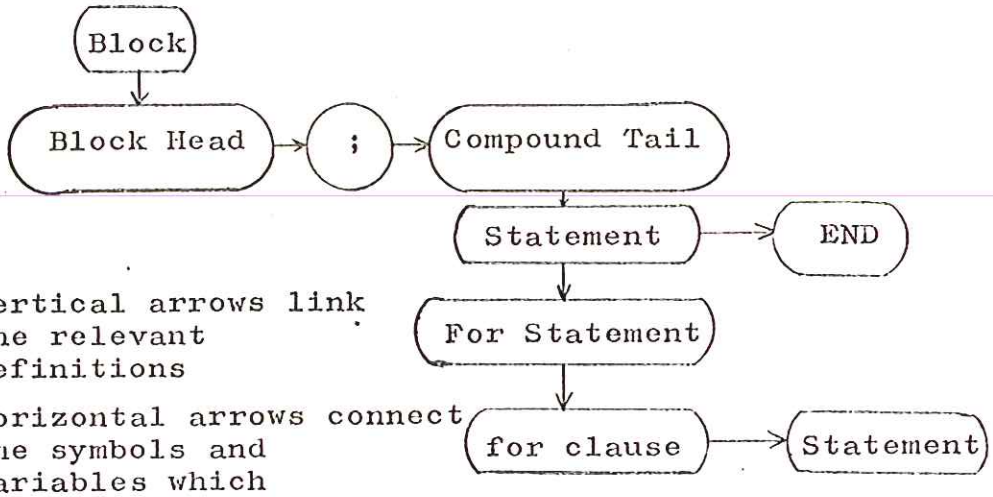
end

DECTYP is checked to make sure that this delimiter does not complete a declaration (e.g. begin real a end), and the subroutines ENDSTA and UNSTAK are used to complete the processing of the preceding statement. The top of the stack is then inspected; if end shows that we are ending a for statement (TS = for begin) we must cycle round (having compiled something into the object program using FSEND) and inspect it again, because ALGOL's definition of a for statement is recursive (see diagram) and therefore one end may terminate lots of for statements.



Note:

Vertical arrows link the relevant definitions

Horizontal arrows connect the symbols and variables which together form a definition.

Having unstacked any lurking for begins, the top of the stack should now be a form of begin. If it is not it is a failure. e.g. for end.

Where the top of the stack is begin, this is the end of a compound statement. begin TR signifies that this is the end of a block containing no array (i.e. no block entry compiled), and begin ALL that this is the end of a block containing an array (and therefore there is an unconditional jump to be updated). For the relevant entries that are unstacked here, see DEC.

In the case of a block, the name list is collapsed back to the first entry for this block, as all the variables declared local to this block are now no longer valid.

A test is then made for the end of the program (is stack empty) and if so ENDPRO is accessed. Otherwise BCR is called to extract any comment following end

e.g. end of this routine;

A warning message is emitted if there is a delimiter in this section, which will catch

```
end
x := 1;
```

Final exit is to OUT 2, as BCR will already have recognised

one of the delimiters which terminate an end comment, namely end, else or ;.

ERRORS

FAIL 40 ; Top of stack not a begin