Diller's Small Programming Language

Diller introduces a small imperative language to provide the context for program proving. It consists of sequences of "commands" with ';' as statement seperator. The language provides assignment to (simple) variables, sequential execution, if-then-else statements, and 'for' and ' 'while' loops. The syntax is:

<cmd> ::=
    skip
    | <var> := <expr>
    | <cmd> ; <cmd>
    | **begin** <cmd> **end**
    | **while** <bool> **do** <cmd>
    | **if** <bool> **then** <cmd> **else** <cmd>
    | **for** <var> := <ll> **to** **do** <cmd>

The syntax for variables, expressions, Boolean conditions, and loop limits will follow usual conventions and is not explicitly given here. Since this language does not provide functions (or subroutines), expressions are simple and have no side-effects.

Note that this is slightly different than Diller in that we provide an explicit syntax for compound statements. This avoids an ambiguity in the programs we write. For instance, in Diller's program in Figure 14.1 (page 191), we have to use our intuition to tell whether the loop body consists of one or two statements. Using the syntax given above, we would rewrite this program as

    {true}
REM := END; QUO := 0;
while SOR ≤ REM **do**
begin REM := REM–SOR; QUO := QUO+1 **end**
    {END = SOR*QUO + REM ∧ REM<SOR}