Definition 5. *Execution of $\mathcal{F}ocal$ statements and expressions is defined in figure 2.2, and has the following shape:*

$$\rightsquigarrow \quad : \quad Module \times state \times Stmts \longrightarrow state$$
$$\rightsquigarrow \quad : \quad Module \times state \times Rhs \longrightarrow heap \times val$$

Note that execution is undefined – will be stuck – if we try to access fields or call methods which are not part of the object, and also, if we execute a conditional where the condition is not a boolean (neither **true** nor **false**). Such situations are possible, since $\mathcal{F}ocal$ is untyped. Wrt to security the question arises whether the language then guards against denial of service attacks - pass the wrong kind of object, and get stuck. The answer is: yes in theory. Even though we could guard against these by throwing exceptions, we cannot guard against denial of service attacks when passed an object on which "our" code will execute a method which will loop forever. All this does not affect the validity of our approach, since we are only claiming partial correctness.[4].