**Figure 12.1** Comparaisons between the mechanisms of macro, function and process.

| | **MACRO** | **FUNCTION** | **PROCESS** |
|---|---|---|---|
| "*lives in*" | text | expression | action |
| *parameterized by* (kind of argument) | arbitrary text | values<br>(through expression) | values<br>(through expression) |
| *body is* | an arbitrary text | an extended expression | a group of actions |
| *returns* | none<br>text expanded *in place* | a value<br>the result of the evaluation | a value<br>the process *exec* |
| *lifespan of the execution* | not applicable<br>the expanded text may have one | 0<br>evaluation is instantaneous | *d*<br>running actions may takes time |
| *where a call may appears* | anywhere | where an expression is expected | where an action is expected |
| *when the arguments are computed* | when the score is loaded, for each occurrence of the argument in the macro body | when the function call is reached by the evaluation flow, one time per argument | when the process call is reached by the execution flow, one time per argument |
| *definition can be recursive* | no | yes | yes |
| *partial application* | no | yes<br>(the result is a function) | no |
| *is a denotable entity* | no | yes<br>a function is a value referred by its name | yes<br>a process is a value referred by its name, as well as the result of the evaluation referred by its exec |
| *possibility to create local variable* | no (and yes)<br>the macro may expand into a group which contains new local variable, but the possibility is not linked to the macro mechanism itself. | yes<br>but the variable exists only during the evaluation (which takes zero time), and they cannot be referred from outside the function body | yes<br>and the local variables of a process can be referred from outside, used in a whenever, etc. |
| *may launch actions* | yes | only messages and assignation<br>(but you can use the `EXPR` construct) | yes |