To represent the control flow graph, the specification DFA and their product, three different structs where used. They have the following definitions:

(struct cfg-prod-struct (non-terminals terminals productions start) #:mutable #:transparent)
(struct dfa-struct (alphabet states transitions start end) #:mutable #:transparent)
(struct cfg-struct (nodes methods entry-nodes return-nodes transfer-edges call-edges) #:mutable #:transparent)

Calls to functions without definition are handled as silent calls and are treated like transfer edges. This choice is the same as removing the calls from the graph altogether. The alternative would be to treat the calls by some default manner which would introduce an arbitrary behaviour in the program.

If the specification DFA is underspecified in relation to the control flow graph, the product algorithm will create unreachable terminals from the start symbol. This might mean that the DFA/CFG product will be unable to generate any words and therefore display emptiness allthough specification does not match the system.

The counter example search used the data structure holding the result of the emptiness test. Each time the counter is reduced due to a found generating term, the counted down variable in the production body is replaced with the terminal that lead to the determination of the generating term. When a counter reaches zero, the corresponding body where the counted down variable resides is then completely replaced with terminals. When the head of this variable is then put in queue to count down the next round, the body with terminals is saved besides it and used to replace non terminals with terminals. At the end of the process, when the starting symbol is counted down to zero, a counter example is attached next to it, coming from prior non terminal count downs.

The empty symbols in the counter example represent either silent actions, non actions or actions by external libraries not specified by the DFA.

Performance seem to be approximately the same regardless of the test used.

One way to optimize the program is to use functions instead of macros for the DSL implementations.