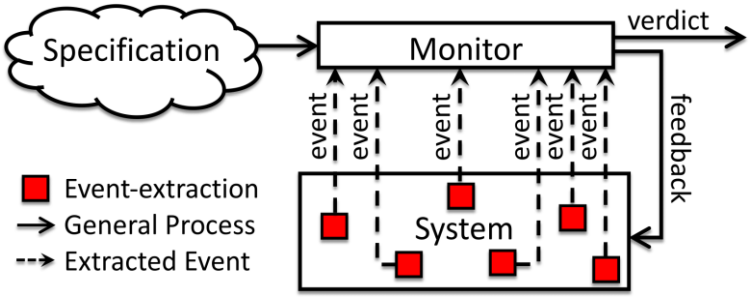
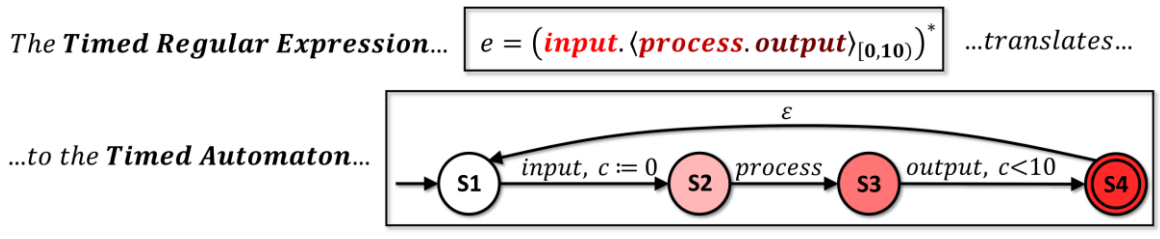
Runtime Verification of Timed Regular Expressions in Larva

Supervisors**:** Prof. Gordon J. Pace and Dr. Christian Colombo

Runtime verification is a software verification technique based on the use of monitors to verify whether a computer program behaves as expected at runtime. One of the critical factors of this technique is the overhead that the monitoring adds to the monitored system. A program's expected behaviour is defined by a specification which is typically expressed using a formal logic. This selected logic and the approach used to adapt the formalism to runtime verification has a direct influence on the degree of overhead produced by the resultant monitor, which must be minimized as much as possible while conserving the expressive power of the selected logic.



Using the Larva runtime verification tool and its automaton-based DATEs (Dynamic Automata with Timers and Events) language as a runtime verification foundation, this project investigates the potential of timed regular expressions as a specification language. Two approaches by which timed regular expressions can be adapted to runtime verification are presented and compared in terms of the overhead that they add to a monitored system. These include an approach based on timed derivatives, where the original expression changes shape in response to the occurrence of events, and an approach based on explicit state exploration, involving the conversion of timed regular expression to timed automata, which are then translated into Larva's DATEs language. A major contribution of this project is an automaton replication technique which allows certain limited forms of timed automaton non-determinism and silent (ε) transitions to be handled using deterministic runtime verification tools, such as Larva.



To evaluate the two approaches, five properties were specified for an open-source real-world FTP server use case called MinimalFTP, which was then monitored according to these properties to measure the memory and CPU overhead added to it at runtime. Through a set of experiments, it was found that both approaches produce significant monitoring overhead, but the timed derivatives approach outperforms the timed automata approach both in terms of the less monitoring overhead it produces and the overall simpler implementation of the approach.