1. Timestamp-based Monitoring Approach (TMA) to verify temporal constraints of Cyber-Physical Systems.

A. What is TMA?

In order to be confident about the behavior of a CPS design or implementation, its timing behavior must be tested and verified. Prior to performing testing of the temporal behavior, timing constraints must be expressed in a formal language that enables robust analysis of constraint satisfiability and consistency. The formal expression also enables the application developer to explicitly specify timing requirements in the design phase and provide the basis to automate the generation of the application and associated test code to enable a more systematic, rigorous, and iterative verification process of the System Under Test (SUT).

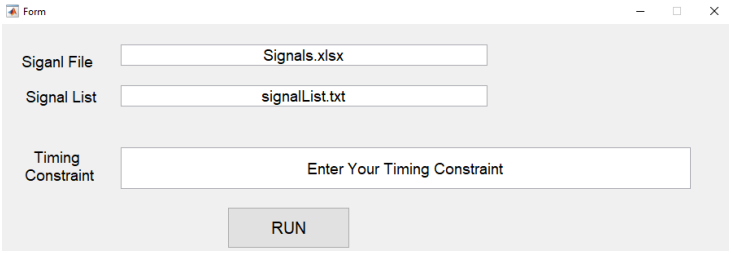
Timestamp-Based Monitoring Approach is an efficient method to have run-time monitoring of temporal specification in Cyber-Physical Systems (CPS) in either offline or online. It receives the required timing constraints for SUT and returns the time durations in which the timing constraints are met. In online monitoring, application constraints are continuously monitored during runtime. Online monitoring can be used to analyze the system behavior in the field and check for bugs in the design. In contrast, offline monitoring in real systems utilizes forensic analysis and therefore does not offer the ability for timely correction of system deviations. The “TMA tool” in MATLAB is an implementation of TMA method in offline manner used to reveal the temporal violations after running CPS. The method receives the timing constraints of CPS written in Timestamp Temporal Logic (TTL) that is a temporal logic working on the events and their timing constraints. Since it is after the operation, it needs the captured signals from the SUT and the timing constraints to be monitored. It does not need to know the system sates or even knowing the structure of the SUT. After providing the signal values and the timing constraints for the tool, it shows the time intervals in which the system timing constraints have not been met.

In general, for a constraint that is defined over a time interval of T, and must be monitored for the duration of experiment d, with a sampling frequency of f, the conventional approach requires $O(T f d)$ computation time. The requirements of both, computation time and memory, depending on the interval size and the sampling rate. In contrast, TMA has a complexity of $O(k)$, where k is the maximum number of events during time d. This is because TMA uses the last two timestamps of the events-of-interest and the last two timestamps of the result are needed. Therefore, four 32 – bit variables for each operator is needed, which is independent of interval length and sampling rate, and a small memory footprint for the state machine. We need one state

machine per operator and the size of each state machine is very small since it needs only 2 bits to store the state.

B. How it works?

Since TMA does not need the details of system implementations and just works on the logged signals, it is enough to have the signal traces and the timing constraints to test. The main file to run the testing in the MATLAB code is Form.m file.



As the figure shows, in order to run TMA, we need to fill the boxes in the form with the items below:

i. Signal file: This is the address of an excel sheet file for the logged traces of a system.

ii. Signal list: This is the list of the signals to monitor. Since a CPS may have a thousand of different signals, by having the signals' names the method just work on the listed signals here to reduce the overhead.

iii. Timing Constraint: This box is dedicated to get the target timing constraint to monitor. Since TMA uses TTL as the temporal logic and there is no prepared parsing algorithms for TTL, the user should be careful to give the right expressions with considering enough parenthesis to make everything separated. If the parenthesis are not enough or correct, the tool gives several errors. The timing constraints should be in the form of the following expressions:

$\mathcal{L}(\phi_1, \phi_2, \varepsilon) > l$	The latency between two events ϕ_1 and ϕ_2 with tolerance of ε should be greater than l
$\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l$	The latency between two events ϕ_1 and ϕ_2 with tolerance of ε should be less than l
$\mathcal{L}(\phi_1, \phi_2, \varepsilon) = l$	The latency between two events ϕ_1 and ϕ_2 with tolerance of ε should be l
$\mathcal{F}(\phi_1, \varepsilon_f) > f$	The frequency of events ϕ_1 should be greater than f with tolerance of ε_f in Hz should be greater than f.
$\mathcal{F}(\phi_1, \varepsilon_f) < f$	The frequency of events ϕ_1 should be greater than f with tolerance of ε_f in Hz should be less than f.
$\mathcal{F}(\phi_1, \varepsilon_f) = f$	The frequency of events ϕ_1 should be greater than f with tolerance of ε_f in Hz should be f.
$\mathcal{S}(\phi_1, \phi_2, \ldots, \phi_n, \varepsilon)$	The events ϕ_1 to ϕ_n should be occurred within ε
$\mathcal{C}(\phi_1, \phi_2, \ldots, \phi_n, \varepsilon)$	The events ϕ_1 to ϕ_n should be occurred in order within ε

$\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) > p$	The phase of two repetitive events on two separate signals ϕ_1 and ϕ_2 should be greater than p with tolerance of ε_p (Their frequencies should be equal with tolerance of ε_f)
$\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$	The phase of two repetitive events on two separate signals ϕ_1 and ϕ_2 should be less than p with tolerance of ε_p (Their frequencies should be equal with tolerance of ε_f)
$\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p$	The phase of two repetitive events on two separate signals ϕ_1 and ϕ_2 should be equal to p with tolerance of ε_p (Their frequencies should be equal with tolerance of ε_f)

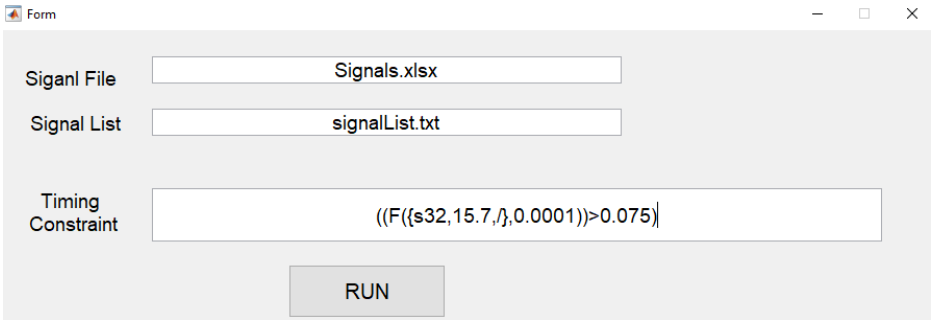
C. What does TMA need to run?

In order to have a demo for the tool, there are some examples embedded in the tool. The "*Signals.xlsx*" contains a trace for an assumed CPS, "*signalList.txt*" is a text file containing a list of the signals to monitor, and "*ConstraintsExamples.txt*" has some examples for timing constraints.

D. An example to run TMA

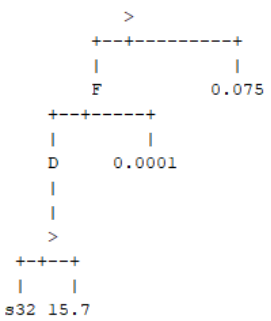
The example below shows the way to give the inputs to the tool and see the evaluated timing constraint.

The timing constraint: ((F({s32,15.7,/},0.0001))>0.075)

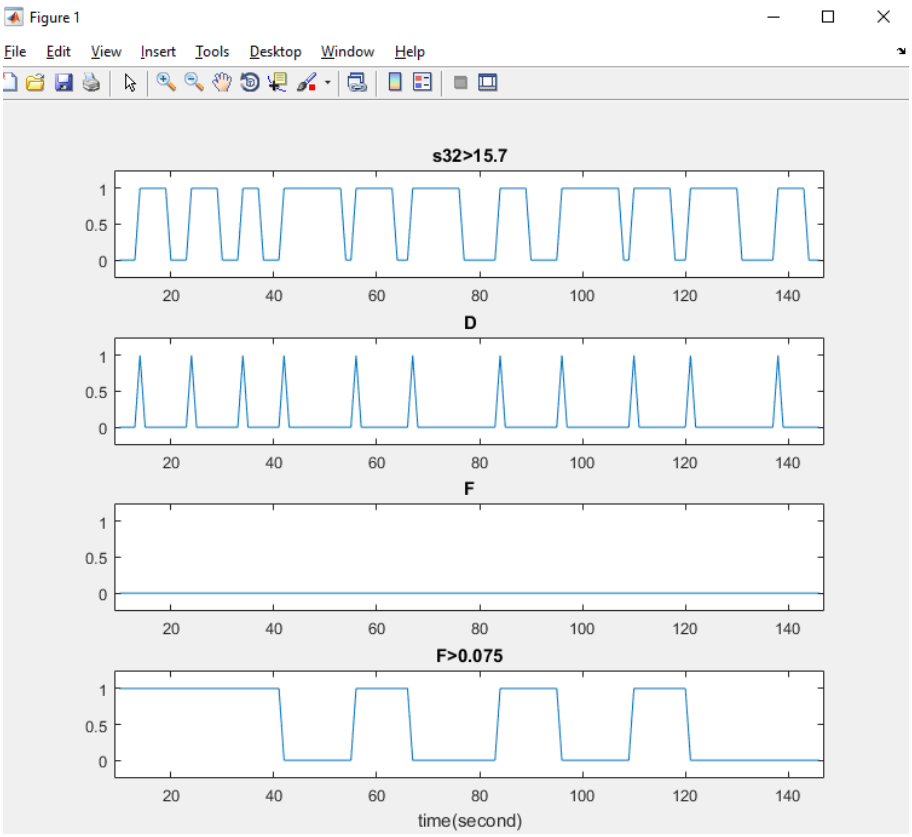


After hitting the Run button, the produced result is:

The parsing tree:



The figures after running:



The figure provides the result for each part of the expression.

D: This part demonstrates the events on signal "*s32*"

F: This part shows the value of frequency (which is a real value) during the time.

F>0.075: This part shows the timing durations in which the frequency is greater than 0.075.