Assignment 2: Runtime Verification of Properties specified in Mission-Time LTL

Design and develop a tool that constructs an RV monitor that verifies a given property specification, and then engages the monitor to verify a given signal trace. You may extend your Assignment 1 submission.

Please refer to the TACAS'14 paper for the definition of the Mission-Time LTL.

Formats of Input and Output

Input

1. Property file
   Format of property file: a text file with a single line of text that contains the formula to be verified. May contain any of the following operators (in order of precedence): ! (not), G(□), H(▣), F(◇), E(◈), X (next), U (until), ^ (and), v (or), and implies (>). For □, ◇, and U, the interval is specified using square brackets and commas. E.g., U[10,15]. For H (▣) and E (◈), a single number is used to denote the upper limit of the interval. E.g. E5 . A valid name for an atomic proposition is any sequence of English alphabets (capital or small). You may assume that characters that denote operators will not be used in the names of atomic propositions.
2. Instrumented signal file
   Same as Assignment 1

Output

Same as Assignment 1, but for one subtle difference. The verdict file contains only verdicts up until the point in time where verdicts may be conclusively determined. For example, consider the `simple_next` property (`A>XB`) in the given `testcases` folder. `system1.input` contains the input signal trace for the time range [0,7]. However, the verdict file contains verdicts only for the time range [0,6]. This is because there is not enough information to give the verdict for time 7.

Working

Same as Assignment 1

Submission

Upload a single zip file `<roll_number>_assignment2.zip` that contains `construct_monitor.sh`, `perform_RV.sh`, and any other source files that you use. The submission will be tested on a standard linux machine that has support for bash, C, C++, Java, Python. Do not use any non-standard libraries.