ECE 522/622 - Spring 2025
Modeling and Verification of Embedded Systems
Problem Set #3: Hybrid Systems Verification

The purpose of this problem set is to introduce you to the PHAVer language and PHAVerLite tool that you will use in lab 3. Specifically, you will use PHAVerLite to check reachability of a timed automaton, and check your results against UPPAAL, which you have learned in lab 2. Background on the PHAVer tool and its algorithms are explained in the research paper listed below, which is available on Canvas.

Since you are allowed to work with a partner in lab 3, you can also work with that partner in this problem set. If you work with a partner, each person should submit the same files under their own account on Canvas.

The software you will use is PHAVerLite, which is an updated version of PHAVer. PHAVerLite analyzes hybrid systems models that are written in PHAVer's syntax. Try installing PHAVerLite soon and use Piazza or office hours for help if you have trouble. As explained in the next section, there are two ways of installing the PHAVerLite software on your computer. You can use either method.

Regardless of whether you use the statically linked executable or build PHAVerLite from source, you should also install the plotutils software package. This software package includes the "graph" program that will plot your results, as we demonstrated in class.

Sources:

- G. Frehse, "PHAVer: Algorithmic Verification of Hybrid Systems past HyTech". *International Journal on Software Tools for Technology Transfer*, 263–279 (2008).
- PHAVer: www-verimag.imag.fr/~frehse/phaver_web/
- PHAVer Manual: www-verimag.imag.fr/~frehse/phaver_web/phaver_lang.pdf
- PHAVerLite webpage: www.cs.unipr.it/~zaffanella/PPLite/PHAVerLite
- Plotutils software (which includes graph program for plotting): www.gnu.org/software/plotutils/
- PHAVerLite (v0.7) executable: github.com/ezaffanella/PHAVerLite/blob/main/releases/phaverlite-0.7_static.gz
- PHAVerLite (v0.7) source: github.com/ezaffanella/PHAVerLite/blob/main/releases/phaverlite-0.7.tar.gz
- flint (v3.1.0): Fast Library for Number Theory: flintlib.org/download/flint-3.1.0.tar.gz
- gmp (v6.3.0): GNU multiple precision arithmetic library
- mpfr (v4.2.2): multiple-precision floating-point computations
- PPLite (v0.12): convex polyhedra library: github.com/ezaffanella/PPLite/blob/main/releases/pplite-0.12.tar.gz

Installing PHAVerLite software:

You can get the PHAVerLite software either by downloading the statically linked executable, or by building it from source, as described below.

Method 1: Downloading statically linked executable: If you are using an x86_64 processor (i.e. not an M series mac, which use AArch64), you have the option to use the statically linked executable from the PHAVerLite Github page (see link above to static executable) instead of building it from source.

Method 2: Build software from source: To build the PHAVerLite from source, use the steps below. You may need to adjust some details according to specifics of your computer. The code can be built on x86_64 or AArm64 computers; the steps are written under the assumption that most people building the software from source will be mac users. If you get stuck, post details on Piazza or come to office hours so we can help.

- Step 1: Install **gmp** and **mpfr.** For example, with homebrew on mac: `brew install gmp mpfr`

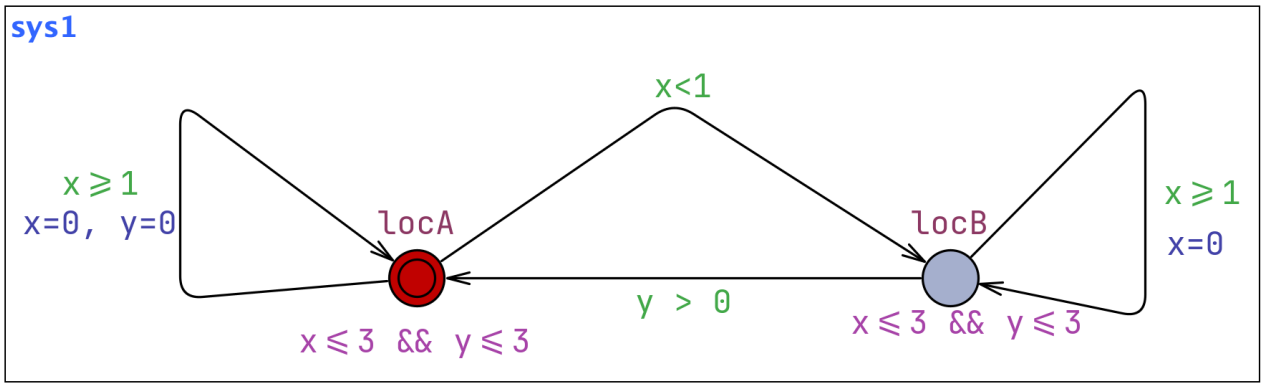
- Step 2: Install **flint** (version 3.1.0). The latest version of flint (version 3.2) does not work with pplite, and for that reason you should not install flint using homebrew. Instead, use the following procedure to install it:
    1. Download flint version 3.1.0 from github (see link above)
    2. Unzip source and change directory in terminal to the flint 3.1.0 directory.
    3. `./configure`
       If you get errors, you can resolve them by passing additional arguments. Use `./configure --help` for options. In my case, the errors were related to not finding gmp and mpfr. The solution was to explicitly provide their directories using the arguments below. Make sure to use the right path for your own setup:
       `--with-gmp=/opt/homebrew/Cellar/gmp/6.3.0`
       `--with-mpfr=/opt/homebrew/Cellar/mpfr/4.2.2`
    4. `make`
    5. `sudo make install`

- Step 3: Install **PPLite** version 0.12:
    1. Download the PPLite source from github (see link above)
    2. Unzip source and change directory in terminal to the pplite-0.12 directory
    3. `./configure`
       If you get errors, you can resolve them by passing additional arguments. Use `./configure -help` for options. In my case, the errors were related to not finding gmp and mpfr. The solution was to explicitly provide their directories using the arguments below. Make sure to use the right path for your own setup:
       `--with-gmp=/opt/homebrew/Cellar/gmp/6.3.0`
       `--with-mpfr=/opt/homebrew/Cellar/mpfr/4.2.2`
    4. `make`
    5. `sudo make install`

- Step 4: Install **PHAVerLite** version 0.7
    1. Download the source from github (see link above)
    2. Unzip source, and change directory in terminal to the phaverlite-0.7 directory
    3. `./configure`
       You may need to pass additional arguments to resolve any errors. Use `./configure -help` for options. In my case, the error was related to not finding gmp. The solution was to explicitly provide their directories by adding this argument:
       `--with-gmp=/opt/homebrew/Cellar/gmp/6.3.0`
    4. `make`
    5. `sudo make install`
    6. Now PHAVerLite should be installed, and you can run it from terminal. For example, if you download the model ps3.pha from Canvas, you can run it using:
       `phaverlite ps3.pha`

Questions to answer:

Timed automata are a restricted case of general hybrid automata (hybrid systems), and therefore timed automata can be modeled in PHAVer. In the "examples" section on Canvas, we have provided the files `ps3.pha` (for PHAVerLite) and `ps3.xml` (for UPPAAL). Both files are models for the same timed automaton, which is shown below, but their included reachability queries are checking different conditions. Take some time to understand the PHAVer model and read carefully the comments included in that file. These comments explain the syntax of PHAVer models that you will need to know.



Q1: Run PHAVerLite to find the reachable states of the PHAVer model using command `phaverlite ps3.pha`. When the tool runs it will tell you how many locations are reached while searching for the reachable parts of the state space. These locations are partitions of the state space.

a. After running PHAVerLite, plot the reachable parts of the state space from the output of the tool using the following command but replace "name1_name2" with your own last names.
`graph -T png -C -B -L name1_name2 -q 0.1 out_inv -C -q 0.5 out_reachable > ps3.png`

b. The plot of reachable states will show you which combinations of x and y are reachable but will not distinguish which parts are reachable in locA and which are reachable in locB. Determine for yourself which parts of the state space are reachable in locB, and then annotate the plot to show this.

Q2: Now check that the PHAVer and UPPAAL models are equivalent.

c. Modify the reachability queries in the PHAVer model to add the condition checked by the query in the UPPAAL model.

d. Modify the UPPAAL model to add 2 new queries that are equivalent to the reachability checks in the PHAVer model.

e. Now each model should be checking reachability of 3 conditions. Run each model using the appropriate tool (PHAVerLite or UPPAAL) and observe the results. Make sure they agree about reachability of all 3 conditions.

Submission Instructions: As your solution, upload to Canvas the graph from Q1, and the PHAVer and UPPAAL models from Q2, with all 3 queries included in each model. Aside from these 3 files, there is no writeup required for this problem set.