

Stability Verification of Optical and Laser Resonators in HOL Light

Umair Siddique and Sofiène Tahar

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada
{muh_sidd,tahar}@ece.concordia.ca

Technical Report

November, 2014

Abstract

Optical and laser resonators are widely used in optical communication, bio-sensors and aerospace systems. Ray optics provides an efficient formalism to analyze the stability properties of such resonators. In this report, we describe the use of our formalization of ray optics in HOL Light theorem prover for the broad range of readers from Computer Science, Physics and Optical Engineering. In particular, we outline the complete steps to execute our source code including the installation for Linux/Windows operating systems. Finally, we give a tutorial like demonstration of the formal analysis of a real-world Fabry Pérot resonator.

1 Introduction

Generally, optical systems are composed of different components (e.g., mirrors and lenses) which process light to achieve desired functionalities such as light amplification, ultrashort pulse generation and astronomical imaging. In order to model and analyze the behavior of such systems, light can be characterized at three levels of abstraction, i.e., ray, electromagnetic and quantum [8]. Geometrical optics (also known as ray optics) describes light as a collection of straight lines which linearly propagates through optical systems. On the other hand, electromagnetic and quantum optics characterize light as a coupled vector field and a stream of photons, respectively. The analysis of engineering optical systems (e.g., refractometry of cancer cells and optical networks) using geometrical optics is an integral part of their design life-cycle. Traditional optical system analysis techniques like paper-and-pencil based proofs and numerical algorithms have some known limitations of human-error proneness and incompleteness, respectively, which impeded their usage in the designing of critical optical systems which may result in the loss of human lives (e.g., laser surgeries) or heavy financial loss (e.g., Hubble Telescope failure [1]). We therefore propose theorem proving based formal methods for the accurate and scalable analysis of optical systems.

The stability analysis of optical resonators identifies geometric constraints of the optical components which ensure that light remains inside the resonator. Both stable and unstable resonators have diverse applications, e.g., stable resonators are used in the measurement of refractive index of cancer cells [13], whereas unstable resonators are used in the laser oscillators for high energy applications [12]. In this report, we describe the use of our formalization of ray optics [11] and optical resonators [10, 9]. Note that only the usability aspects are presented in tutorial like fashion and technical details are omitted which can be found in [11, 10, 9]. In Section 2, we present the installation detail of HOL Light on Linux and Windows operating systems along with the steps to load our optical resonator stability analysis files which we refer to as **Formal-Stability**. We present a case study about the formal stability analysis of a real-world Fabry P erot resonator in Section 3. Finally, we conclude our report in Section 4 and point to further readings.

2 Installation

In this section, we describe the procedure to install HOL Light and load our stability verification code (**Formal-Stability**) on two most widely used operating systems, i.e., Linux and Windows.

2.1 Linux

Generally, there are several methods to install HOL Light on different Linux based systems. Here, we describe the use of a little script that will download, compile and install HOL Light, including compatible versions of OCaml and Camlp5. This is a very quick and easy way to get things up and running, even if the distribution does not provide suitable packages for OCaml. All installation is local to a given directory (say, under your home directory). So, no root permissions are required in case you are using your university's or organizations' server. This script has been tested on Linux (both 32 and 64 bit). It might also work on other Unix-like systems, possibly with modifications.

Open the terminal and execute the following commands:

Step 1: We need to install the subversion (svn) package by the following command:

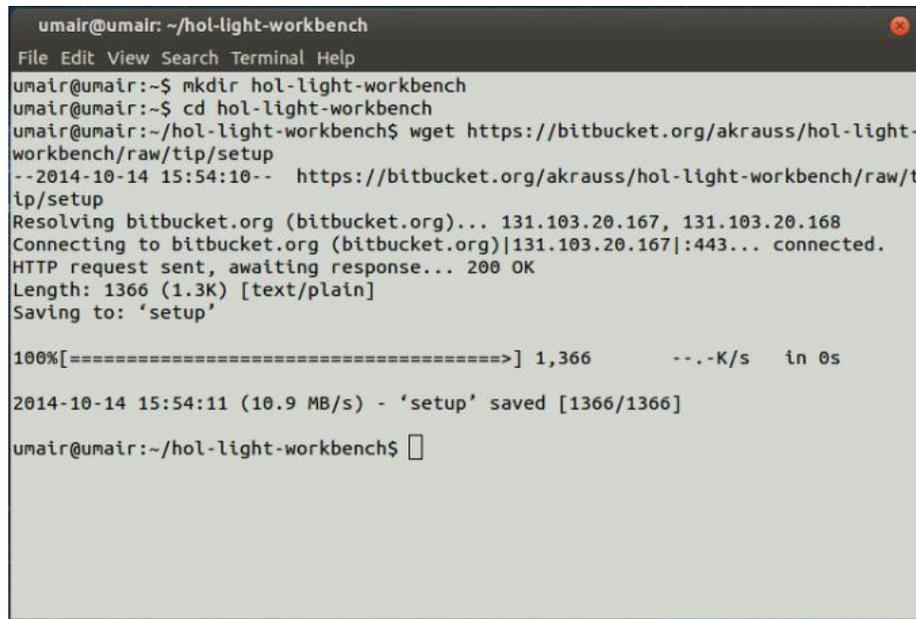
```
sudo apt-get install subversion
```

Step 1: `mkdir hol-light-workbench`

Step 2: `cd hol-light-workbench`

Step 3: `wget https://bitbucket.org/akrauss/hol-light-workbench/raw/tip/setup`

The corresponding output is shown in Figure 1.



```
umair@umair: ~/hol-light-workbench
File Edit View Search Terminal Help
umair@umair:~$ mkdir hol-light-workbench
umair@umair:~$ cd hol-light-workbench
umair@umair:~/hol-light-workbench$ wget https://bitbucket.org/akrauss/hol-light-
workbench/raw/tip/setup
--2014-10-14 15:54:10-- https://bitbucket.org/akrauss/hol-light-workbench/raw/t
ip/setup
Resolving bitbucket.org (bitbucket.org)... 131.103.20.167, 131.103.20.168
Connecting to bitbucket.org (bitbucket.org)|131.103.20.167|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1366 (1.3K) [text/plain]
Saving to: 'setup'

100%[=====] 1,366      --.-K/s   in 0s

2014-10-14 15:54:11 (10.9 MB/s) - 'setup' saved [1366/1366]

umair@umair:~/hol-light-workbench$
```

Fig. 1: HOL Light Installation on Linux

Step 4: `chmod 755 setup`

Step 5: `./setup`

The corresponding output is shown in Figure 2.

Step 6: `. setpaths`

The installer creates a little script with shell environment settings (PATH etc.). Import it into the current shell as follows to adjust the paths.

Finally, we can verify the installation by loading HOL Light (see Figure 2) as follows:

```
cd hol-light
```

```
ocaml
```

```
#use "hol.ml";;
```

```

umair@umair: ~/hol-light-workbench
File Edit View Search Terminal Help
" -o `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.' | cut -c1-6` = "6.06" ; \
    then cp pa_j_3.1x_6.02.2.ml pa_j.ml; \
    else if test `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.'
| cut -c1-6` = "6.06" -o `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.' | cu
t -c1-6` = "6.07" -o `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.' | cut -c1
-6` = "6.08" -o `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.' | cut -c1-6` =
"6.09" -o `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.' | cut -c1-6` = "6.1
0" -o `camlp5 -v 2>&1 | cut -f3 -d' ' | cut -f1-3 -d'.' | cut -c1-6` = "6.11" ;
\
    then cp pa_j_3.1x_6.11.ml pa_j.ml; \
    else cp pa_j_3.1x_`camlp5 -v 2>&1 | cut -f3 -d' ' | cut -
c1`.xx.ml pa_j.ml; \
        fi \
        fi \
        fi \
        fi
if test `ocamlc -version | cut -c1-3` = "3.0" ; \
    then ocamlc -c -pp "camlp4r pa_extend.cmo q_MLast.cmo" -I `ca
mlp4 -where` pa_j.ml; \
    else ocamlc -c -pp "camlp5r pa_lexer.cmo pa_extend.cmo q_MLas
t.cmo" -I `camlp5 -where` pa_j.ml; \
        fi
Done. Run ". setpaths", to set adjust PATH
umair@umair:~/hol-light-workbench$

```

Fig. 2: HOL Light Installation on Linux

```

umair@umair: ~/hol-light-workbench/hol-light
File Edit View Search Terminal Help
Warning: inventing type variables
0..0..0..solved at 3
val instantiate_casewise_recursion : term -> thm = <fun>
val pure_prove_recursive_function_exists : term -> thm = <fun>
val prove_general_recursive_function_exists : term -> thm = <fun>
val define : term -> thm = <fun>
- : unit = ()
File "define.ml" already loaded
- : unit = ()
val help_path : string list ref = {contents = ["$/Help"]}
val help : string -> unit = <fun>
val theorems : (string * thm) list ref = {contents = []}
val omit : term -> term = <fun>
val exactly : term -> term = <fun>
val name : string -> term = <fun>
val search : term list -> (string * thm) list = <fun>
- : unit = ()
File "help.ml" already loaded
- : unit = ()
- : unit = ()
- : unit = ()
Camlp5 Parsing version 5.15
#

```

Fig. 3: Loading HOL Light

2.2 Windows

We can install HOL Light on Windows operating system (32 or 64 bit) without the Cygwin. Following are the detailed steps for installation:

Step 1: We need to install [Objective CAML 3.09.3](http://caml.inria.fr/pub/distrib/ocaml-3.09/) which can be downloaded (.zip format) here:

<http://caml.inria.fr/pub/distrib/ocaml-3.09/>

Do not install the latest version 3.10 or later.

Step 2: Download HOL Light from <http://www.cl.cam.ac.uk/~jrh13/hol-light/>. Best way is to download by subrevision which can be downloaded as an executable file from <https://sliksvn.com/en/download/>. Once subrevision is installed, we can download HOL Light by using the command prompt:

```
svn checkout http://hol-light.googlecode.com/svn/trunk/ hol_light
```

Step 3: Change the current directory with the DOS command `cd` to make `hol_light` the current directory.

Step 4: For HOL-Light to run for the first time, you also need to create files called `pa_j.cmo` and `pa_j.cmi` in your `hol_light` directory. If you know somebody else who has created these files using the same versions of [Objective CAML](#) and HOL Light, you can copy their files. Otherwise, you need to create them yourself as follows. Rename the file `pa_j-3.09.ml` to `pa_j.ml`. (It can be done manually).

Step 5: Run the following command:

```
ocamlc -c -pp "camlp4r pa_extend.cmo q_MLast.cmo" -I +camlp4 pa_j.ml
```

This should create the files `pa_j.cmo` and `pa_j.cmi`. It can be verified in the HOL Light folder manually.

As described in the previous section, we can verify the installation by loading HOL Light (see Figure 2) as follows:

```
cd hol-light
```

```
ocaml
```

```
#use "hol.ml";;
```

On most computers it takes a few minutes to start up. Note that `#` is necessary even though you might see it on screen it will look like this `# #use "hol.ml";;`)

2.3 Loading Formal-Stability

The latest version of the code can be downloaded from <http://hvg.ece.concordia.ca/projects/optics/rayoptics.htm>. Table 1 outlines the included files and their corresponding description.

We need to unzip `ray.zip` and move (in `ocaml`) to the directory containing our files:

```
#cd "path/to/directory";;
```

Finally, load the top file:

```
needs "main.ml";;
```

This will also load all the intermediate files¹. If the last theorem showing on window is `STABLE_FP`, this means all the files have been loaded correctly.

¹This development needs "Multivariate" analysis libraries of HOL Light. It takes 3 hours on my laptop with Intel Core i3 processor and 4GB of RAM. On my lab server it takes one hour or so. In any case it takes a while to load depending upon the machine.

Table 1: Structure of Formal-Stability Code

File Name	Description
main.ml	Top file which can load all the required libraries
ray_optics.ml	Core formalization of ray optics. It includes type definitions, modeling of the physical behavior of ray and ray transfer matrices.
component_library	Formalization of some basic optical components such as thin lens, thick lens and parallel plate. These components can further be used to formalize more complex optical components
resonator.ml	Main formalization of optical resonators, their validity, some fundamental operations (e.g., unfolding and round trip) and generalized stability theorem for symmetric and non-symmetric resonators
fp_fiber_rod_lens	Modeling and stability analysis of Fabry P�erot resonator with fiber-rod lens
z_resonator	Modeling and stability analysis of Z-shaped resonator
fp_resonator	Modeling and stability analysis of two mirror Fabry P�erot resonator

Last Loaded Theorem

```

val (STABLE_FP) : thm =
  |- !R d n.
    ~ (R = &0) /\ &0 < d / R /\ d / R < &2 /\ &0 < n /\ &0 < d
    ==> is_stable_resonator (fp_cavity R d n)

```

3 Case Study: Fabry-P erot Resonator with Fiber-rod-lens

In optics literature, Fabry-P erot resonator has numerous variants in terms of its structure due to its wide scope of applications (e.g., wavelength division multiplexing [7] measurement of the refractive index of cancer cells [13] and optical bio-sensing devices [2]). Recently, a state-of-the-art FP core architecture has been proposed which overcomes the some known limitations of existing FP resonators [6, 4]. In the new design, cylindrical mirrors are combined with a fiber rod lens (FRL) inside the cavity, to focus the beam of light in both transverse planes as shown in Figure 4 (a). The fiber rod lens is used as light pipe which allows the transmission of light from one end to the other with relatively small leakage. Building a stable FP resonator requires to determine the geometric constraints in terms of the radius of curvature of mirrors R and the free space propagation distance (d_{free_space}) using the stability analysis. It is important to note that the design shown in Figure 4 (a), has a 3-dimensional structure. We can still apply the ray-transfer-matrix approach to analyze the stability by dividing the given architecture into two planes, i.e., XZ and YZ planes. In this case, the stability problem becomes a couple of planar problems which are still valid since the ray focusing behaviours in both directions (XZ and YZ) are decoupled. This can be seen in Figure 4 (b) and (c), where the resonator is divided into two cross-sections. In the following, we focus only on the analysis of the YZ plane, since the analysis in the XZ plane is fairly similar (the complete analysis can be found in [10] along with the source code at [14]).

In the YZ cross-section (Figure 4 (c)), the curved mirrors becomes a straight mirror and the fiber rod lens acts as a converging lens. In this case, a ray that makes a round-trip in the cavity undergoes (from left to right) first reflection from the straight mirror, propagation

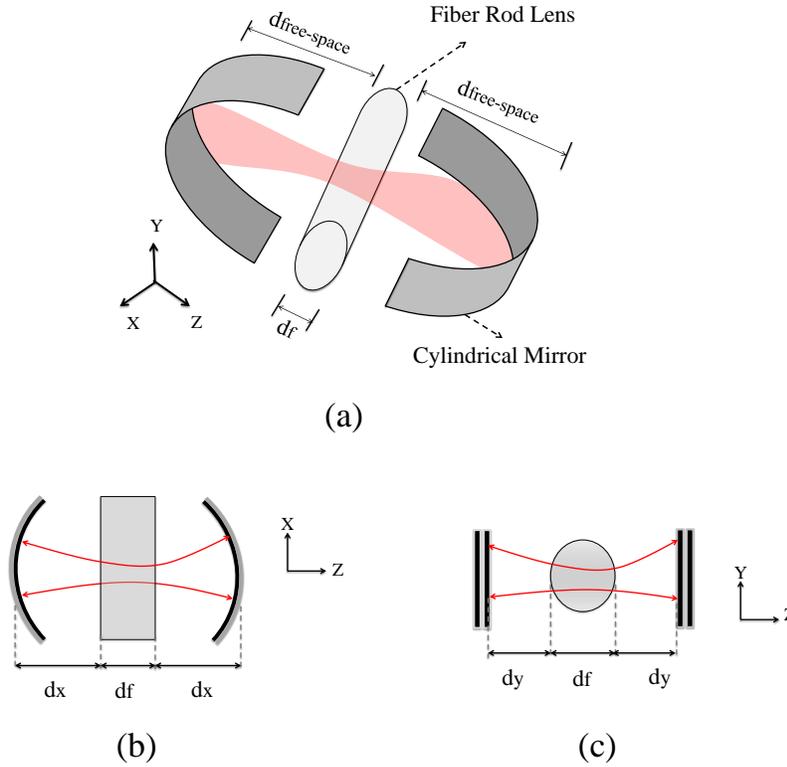


Fig. 4: Fabry PÉrot (FP) Resonator with fiber rod lens (a) 3-Dimensional Resonator Design (b) Cross-Section view in the XZ Plane (c) Cross-Section view in the YZ Plane

through free space of length d_y and refractive index 1, refraction through the curved interface with radius of curvature $\frac{d_f}{2}$ (due to concavity), propagation within the fiber rod lens of length d_f and refractive index n_f , refraction through the curved interface with radius of curvature $-\frac{d_f}{2}$ (due to convexity) and again the propagation through free space of length d_y . We formally model this system description as follows:

HOL Script: FP Resonator Model in YZ Plane

```
let fp_frl_cav_ym = new_definition
  ' fp_frl_cav_ym dy df nf : resonator =
    plane, [(&1,dy), (spherical (df/ &2)), transmitted;
            (nf,df), (spherical (--df/ &2)), transmitted], (&1,dy), plane';;
```

Output: FP Resonator Model in YZ Plane

```
|- !nf df dy.
    fp_frl_cav_ym dy df nf =
    plane, [(&1,dy), spherical (df / &2), transmitted;
            (nf,df), spherical (--df / &2), transmitted],
            (&1,dy), plane
```

where function `fp_frl_cav_ym` takes the parameters free space of length (`dy`), length of fiber

rod lens (**df**) and refractive index (**nf**) and it returns an optical resonator.

Now, we analyze this model in three steps: 1) Model Validity; 2) Verification of ray-transfer-matrix and 3) the stability.

Step 1: For an optical resonator structure is considered to be valid, it should satisfy some constraints such as distances cannot be negative, and refractive index needs to be non-negative etc. We can check the validity of the model using the tactic `VALID_RESONATOR`. The general format of the tactic is as follows: `VALID_RESONATOR def goal`, it takes two parameters, i.e., `def` which indeed represents the definition of the resonator mode (in our case `fp_frl_cav_yz`) and `goal` which is a desired goal that needs to be proved. So we can verify resonator validity as follows:

HOL Script: FP Resonator Model Validity

```
VALID_RESONATOR fp_frl_cav_yz
  ' !dy df nf. &0 < dy /\ &0 < df /\ &0 < nf
    ==> is_valid_resonator (fp_frl_cav_yz dy df nf) ';;
```

Output: FP Resonator Model Validity

```
1 basis elements and 0 critical pairs
2 basis elements and 0 critical pairs
2 basis elements and 0 critical pairs
3 basis elements and 0 critical pairs
3 basis elements and 0 critical pairs
1 basis elements and 0 critical pairs
val it : thm =
|- !dy df nf.
  &0 < dy /\ &0 < df /\ &0 < nf
  ==> is_valid_resonator (fp_frl_cav_yz dy df nf)
```

Step 2: It is clear from the Figure 3 (c) that the resonator configuration in YZ plane is symmetric which means that order of components in unfolded resonator is equal to the concatenation of two half round trips. Mathematically, for any resonator, the system composition is equal to square of a matrix (which essentially represents the half round trip). We define the matrix of half round trip as follows:

HOL Script: FP Resonator Half Roundtrip Matrix

```
let fp_frl_yz_mat_u = new_definition
  ' fp_frl_yz_mat_u dy df nf =
    mat2x2 ( --(df * ( -- &2 + nf) + &4 * dy * ( -- &1 + nf)) / (df * nf))
            (((df + &2 * dy) * (df - &2 * dy * ( -- &1 + nf))) / (df * nf))
            ((&4 - &4 * nf) / (df * nf))
            ( --(df * ( -- &2 + nf) + &4 * dy * ( -- &1 + nf)) / (df * nf)) ';;
```

Then we verify the symmetry in HOL Light as follows (here, we give the whole proof which involves mainly rewriting and quite obvious):

Complete Proof Script: FP Resonator Matrix (in Symmetric Form)

```
let FP_FRL_YZ_MATRIX = prove ('
  !dy df nf dy. !dy df nf. &0 < dy /\ &0 < df /\ &0 < nf ==>
  system_composition (unfold_resonator_s (fp_frl_cav_yz dy df nf) 1) =
  fp_frl_yz_mat_u dy df nf pow 2',

(*-----PROOF STEPS-----*)

REWRITE_TAC[fp_frl_yz_mat_u;MAT_POW2] THEN
REWRITE_TAC common_defs THEN
REPEAT(POP_ASSUM MP_TAC) THEN REWRITE_TAC[GSYM ONE] THEN
SIMP_TAC[GSYM MATRIX_MUL_ASSOC] THEN
SIMP_TAC[MAT2X2_MUL;REAL_MUL_RZERO;REAL_MUL_LZERO;REAL_ADD_LID;
  REAL_ADD_RID;REAL_MUL_LID;REAL_MUL_RID] THEN
SIMP_TAC[MAT2X2_EQ ] THEN
SIMP_TAC[lemma1] THEN
CONV_TAC REAL_FIELD);;
```

Step 3: Finally, we formally verify the stability of the FP resonator in YZ-plane as follows:

Complete Proof Script: Stability Verification

```
let STABLE_FP_FRL_YZ = prove('
  !dy df nf. &0 < dy /\ &0 < df /\ &0 < nf /\
  &0 < (&1 - &2/nf) + (&4*dy/df)*(&1 - &1/nf) /\
  (&1 - &2/nf) + (&4*dy/df)*(&1 - &1/nf) < &1 ==>
  is_stable_resonator(fp_frl_cav_yz dy df nf)',

(*-----PROOF STEPS-----*)

REPEAT STRIP_TAC THEN MP_REWRITE_TAC STABILITY_THEOREM_SYM THEN
EXISTS_TAC('fp_frl_yz_mat_u dy df nf :real^2^2') THEN
CONJ_TAC THENL[ASM_SIMP_TAC[VALID_FP_FRL_YZ_CAVITY]; ALL_TAC] THEN
CONJ_TAC THENL[ASM_SIMP_TAC[FP_FRL_YZ_MATRIX]; ALL_TAC] THEN
CONJ_TAC THENL[REWRITE_TAC[DET_MAT2X2;fp_frl_yz_mat_u] THEN
REPEAT(POP_ASSUM MP_TAC) THEN
SIMP_TAC[lemma2] THEN
asm_real_prove ; ALL_TAC] THEN
REWRITE_TAC[fp_frl_yz_mat_u ;VECTOR_2;mat2x2 ] THEN
IMP_REWRITE_TAC[LEMMA_BASIC] THEN
REPEAT(POP_ASSUM MP_TAC) THEN
SIMP_TAC[lemma3] THEN
asm_real_prove);;
```

The verification of this theorem is a direct consequence of generalized stability theorem [14] and Steps 1 and 2.

It is important to note that for the case of FP resonator with fiber rod lens, we have obtained two sets of stability constraints, i.e., in the XZ plane and in the YZ plane. In fact, the resonator can be stable in one plane and unstable in the other. Therefore, stability constraints in both planes have to be satisfied. In real-world scenarios, the most fundamental step is to find the allowable values of the parameters associated with resonators such as radius of convergence and the width of free space. The verification of above theorems have been done in a generic form, i.e., we derive the stability constraints for arbitrary values of R , d_x , d_f and n_f . This is one of the main advantages of theorem proving based stability analysis of optical resonators. We further demonstrate the strength of our approach by the verification of stability constraints used as the guidelines for the fabrication of FP resonators, reported in [5]. In the design it is considered that $d_x = d_y = d$ and the values of n_f and d_f are fixed and equal to 1.47 and $125\mu m$, respectively. The main goal is to find the ranges of d where the resonator is stable in both planes. We developed a tactic (`STABILITY_PROVE_TAC`) which can automatically verify that the resonator is stable under the given range of parameters. For example, first we define the resonator model in XZ and YZ planes as follows:

HOL Light Script: FP Resonator Real-world Model

```
let FP_XZ_RES = define '
    FP_XZ_RES d = fp_frl_xz_cavity
    (#140 * #0.000001) d (#125.0 * #0.000001) (#1.47)';;

let FP_YZ_RES = define '
    FP_YZ_RES d = fp_frl_cav_yz
    d (#125.0 * #0.000001) (#1.47)';;
```

Based on the general format (`STABILITY_PROVE_TAC`) needs two parameters, i.e., the goal which needs to be proved and list of theorems or definitions. One particular case is given as follows:

Automatic Verification of Resonator Stability

```
let RANGE_1_YZ = STABILITY_PROVE
    '(d IN real_interval (#97.5 * #0.000001, #132.9 * #0.000001))
    ==> is_stable_resonator (FP_YZ_RES d)' [FP_YZ_RES];;

(*-----VERIFIED THEOREM-----*)

val ( RANGE_1_YZ ) : thm =
  |- d IN real_interval (#97.5 * #0.000001, #132.9 * #0.000001)
    ==> is_stable_resonator (FP_YZ_RES d)
```

It is interesting to see the time required to verify the resonator stability. We have implemented a variant of above mentioned tactic namely `STABILITY_PROVE_TIME` which also provides the cpu time along with the verified theorem. We have verified important stability ranges in XZ and YZ domain inspired from the paper [5]. The corresponding theorems along with timing information are given as follows:

Different Stability Ranges in XZ and YZ Domain

```
(*-----RANGE 1-----*)

CPU time (user): 3.381
val ( RANGE_1_XZ_TIME ) : thm =
  |- d IN real_interval (#27.5 * #0.000001,#35 * #0.000001)
    ==> is_stable_resonator (FP_XZ_RES d)

(*-----RANGE 2-----*)

CPU time (user): 3.121
val ( RANGE_2_XZ_TIME ) : thm =
  |- d IN real_interval (#38 * #0.000001,#97 * #0.000001)
    ==> is_stable_resonator (FP_XZ_RES d)

(*-----RANGE 3-----*)

CPU time (user): 3.456
val ( RANGE_1_YZ_TIME ) : thm =
  |- d IN real_interval (#97.5 * #0.000001,#132.9 * #0.000001)
    ==> is_stable_resonator (FP_YZ_RES d)

(*-----RANGE 4-----*)

CPU time (user): 3.42
val ( RANGE_2_YZ_TIME ) : thm =
  |- d IN real_interval (#38 * #0.000001,#97 * #0.000001)
    ==> is_stable_resonator (FP_YZ_RES d)
```

4 Conclusions

In this report, we described the use of our resonator stability code (Formal-Stability). In particular, we describe the detailed steps to install HOL Light on Linux and Windows based systems followed by the loading steps for our source code. Finally, we presented the case study to verify the stability of state-of-the-art Fabry P erot resonator. The main intention was to give a tutorial like introduction to non formal methods people (e.g., Physicists and optical engineers) and formal methods people which are not user of HOL Light. This work is a part of an ongoing project to develop a formal reasoning support for the analysis of geometrical optics. More details can be found at <http://hvg.ece.concordia.ca/projects/optics/rayoptics.htm> and interested readers can find more details in [3, 9, 10, 11].

References

- [1] The Hubble Space Telescope Optical System Failure Report. Technical report, NASA, 1990.

- [2] M. Baaske and F. Vollmer. Optical Resonator Biosensors: Molecular Diagnostic and Nanoparticle Detection on an Integrated Platform. *ChemPhysChem*, 13(2):427–436, 2012.
- [3] S. Khan-Afshar, U. Siddique, M. Y. Mahmoud, V. Aravantinos, O. Seddiki, O. Hasan, and S. Tahar. Formal Analysis of Optical Systems. *Mathematics in Computer Science*, 8(1):39–70, 2014.
- [4] M. Malak, F. Marty, N. Pavy, Y.-A. Peter, Ai-Qun Liu, and T. Bourouina. Cylindrical Surfaces Enable Wavelength-Selective Extinction and Sub-0.2 nm Linewidth in 250 μm -Gap Silicon Fabry-Perot Cavities. *IEEE Journal of Microelectromechanical Systems*, 21(1):171–180, Feb. 2012.
- [5] M. Malak, N. Pavy, F. Marty, Y. Peter, A.Q. Liu, and T. Bourouina. Stable, High-Q Fabry-Perot Resonators with Long Cavity Based on Curved, All-Silicon, High Reflectance Mirrors. In *IEEE 24th International Conference on Micro Electro Mechanical Systems (MEMS)*, pages 720–723, 2011.
- [6] M. Malak, N. Pavy, F. Marty, E. Richalot, A. Q. Liu, and T. Bourouina. Design, Modeling and Characterization of Stable, High Q-factor Curved Fabry Perot cavities. *Microsyst. Technol.*, 17(4):543–552, 2011.
- [7] B. Saadany, M. Malak, M. Kubota, F. M. Marty, Y. Mita, D. Khalil, and T. Bourouina. Free-Space Tunable and Drop Optical Filters Using Vertical Bragg Mirrors on Silicon. *IEEE Journal of Selected Topics in Quantum Electronics*, 12(6):1480–1488, 2006.
- [8] B. E. A. Saleh and M. C. Teich. *Fundamentals of Photonics*. Wiley, 2007.
- [9] U. Siddique, V. Aravantinos, and S. Tahar. A New Approach for the Verification of Optical Systems. In *Optical System Alignment, Tolerancing, and Verification VII*, volume 8844 of *SPIE*, pages 88440G–88440G–14, 2013.
- [10] U. Siddique, V. Aravantinos, and S. Tahar. Formal Stability Analysis of Optical Resonators. In *NASA Formal Methods*, Lecture Notes in Computer Science (LNCS), pages 368–382, 2013.
- [11] U. Siddique, V. Aravantinos, and S. Tahar. On the Formal Analysis of Geometrical Optics in HOL. In *Automated Deduction in Geometry*, volume 7993 of *Lecture Notes in Computer Science*, pages 161–180, 2013.
- [12] A. E. Siegman. *Lasers*. University Science Books, 1st edition, 1986.
- [13] W. Z. Song, X. M. Zhang, A. Q. Liu, C. S. Lim, P. H. Yap, and Habib Mir M. Hosseini. Refractive Index Measurement of Single Living Cells Using On-Chip Fabry-Perot Cavity. *Applied Physics Letters*, 89(20):203901, 2006.
- [14] U. Siddique and S. Tahar. Formal Reasoning about Geometrical Optics, Hardware Verification Group, Concordia University, QC, Canada. <http://hvg.ece.concordia.ca/projects/optics/rayoptics.htm>, 2014.