g-HOL - A Graphical User Interface for the HOL Proof Assistant

Fahd Arshad, Hassan Mehmood, Fauzan Raza and Osman Hasan

School of Electrical Engineering and Computer Science~~s~~ (SEECS)~~,~~
National University of Sciences and Technology (NUST), Islamabad, Pakistan
{10besefarshad,11besehmehmood,10besefraza,osman.hasan} @seecs.nust.edu.pk

Abstract. Given the high expressiveness of higher-order logic, their proof assistants are being widely advocated for formally verifying cyber-physical systems these days. However, the usage of higher-order-logic proof assistants is mostly restricted to academia. One of the foremost reasons for the hesitancy of their usage in the industrial setting is their user-unfriendly interfaces. This paper presents a user-friendly graphical user interface (GUI) g-HOL for HOL, which is a widely used higher-order-logic proof assistant that has been successfully used to verify many physical, hardware and software systems and formalize most of the classical mathematical theories. g-HOL is developed in Java swing and is supported by the Windows, Linux and MAC operating systems. It tends to minimize syntax errors and the need to memorize and type commands and facilitates the searching process, which is frequently required in interactive formal reasoning. The paper describes the architecture and main features of g-HOL using an illustrative example.

Keywords: Higher-order logic, Proof assistants, Theorem Proving, HOL

1 Introduction

Theorem proving [4]. is one of the most widely used formal verification method [4]. The system that needs to be analyzed is mathematically modeled in an appropriate logic and the properties of interest are verified using computer-based formal tools called theorem provers or proof assistants.

The human interaction or the manual proof effort required for proving logical formulas in a theorem prover varies from trivial to complex depending on the underlying logic. For instance, propositional logic [4] is decidable, i.e., the logical correctness of a formula specified in propositional logic can be automatically verified using an algorithm. The main limitation of propositional logic is its limited expressiveness, as it cannot be used to represent verification problems for all sorts of systems. On the other hand, higher-order logic [4] is the most expressive form of logic that allows quantification over functions and sets. These features make it so expressive that any system, along with its continuous and unpredictable elements, can be described using higher-order logic given that its behavior can be expressed in a closed mathematical form. The added expressiveness of higher-order logic comes at the cost of manual verification where user input is required to verify all formulas expressed in higher-order-logic, due to its un-decidable nature, using a proof assistant. The user interacts with the proof assistant by providing it with the necessary tactics to prove goals. This process could be very tedious and usually takes thousands of lines of proof script and hundreds of man-hours for verifying mathematical analysis described in a page.

Due to the above-mentioned proof efforts, coupled with the rather user-unfriendly command-line interfaces of proof assistants, the higher-order-logic theorem proving is very rarely used in the industry [3], despite the great potential of this technology in verifying the correctness of complex engineering systems. In order to alleviate this problem, this paper presents a Graphical User Interface (GUI), i.e., g-HOL [2], for the widely used HOL proof assistant [1], which has been successfully usage as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

g-HOL tends to facilitate the HOL learning process and enhance the productivity, usefulness and effectiveness of HOL users. We have tried to make g-HOL equally useful for both experienced and novice HOL users. Its key features are:

- It permits viewing the already verified theorems in the HOL libraries.
- It largely removes the requirement to memorize HOL theorems and tactics.
- It may allow to finish HOL proofs by using the mouse clicks only.
- It allows archiving and loading proof scripts.
- It has its own text editor that is built into the interface.

2 g-HOL's Architecture

g-HOL [2] is developed using Java Swing [5], i.e., a framework specifically for designing and developing GUIs in Java. This choice was made due to the flexibility, platform independence and the large user community of the Java language. The g-HOL GUI follows a simple Model view controller (MVC) pattern. Figure 1 describes the architecture of g-HOL, which is composed of 5 main components:

1. HOL Theorem Prover (Back-end Software that does all the theorem proving)
2. Linker (Bridge between the HOL theorem Prover and g-HOL)
3. g-HOL (Front end Graphical Use Interface)
4. GraphicView Plugin (To create the layout using Java Swing)
5. View Controller (This consists of different type of listeners that handle interrupts, like button clicks etc.)

3 Illustrative Example

In this section, we illustrate various features of g-HOL by working through a simple proof example: $(a \wedge b = b \wedge a)$. The first step in the proof process is to load the appropriate theories required for the given proof goal. For example, the above proof goal requires the Boolean (*Bool*) theory in HOL. We can open and load theories by using the appropriate buttons in the menu pane. g-HOL allows

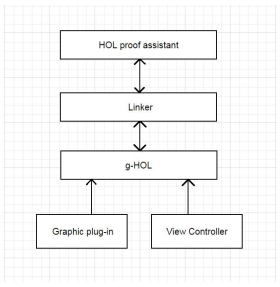


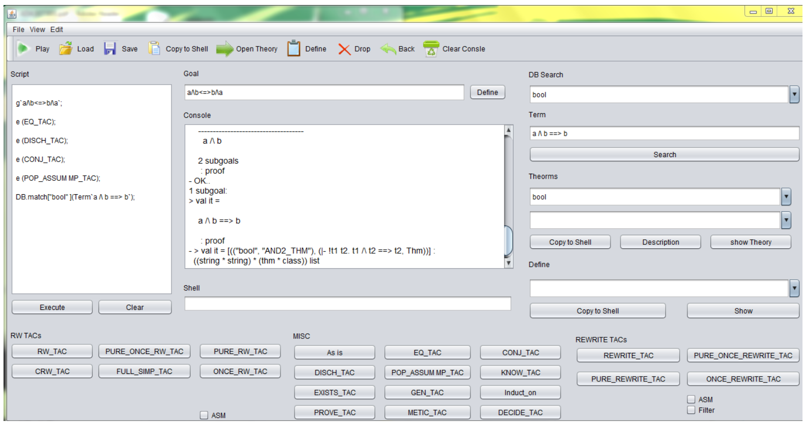
Fig. 1. Architecture Diagram g-HOL

loading multiple theories at the same time. Once the theory is loaded, it can be accessed using the *DB-Search* and *Theorem* fields.

Next, we enter the proof goal in the *Goal* field and click the *Define* button to define it as a proof goal. Considering that there is no syntax error, the new goal stack can be viewed in the *HOL console* of g-HOL. Note that the *Script* field keeps track of the running script. The script can be loaded from an existing file or by highlighting selected text from the editor and clicking the *Execute* button.

Even though this proof goal can be discharged by using one of the automatic provers in HOL, e.g., clicking the *PROVE_TAC* button in g-HOL, we would present an interactive way to verify it to illustrate the working with g-HOL. We proceed with the backward proof by splitting the equality into two implications. This can be done by apply the HOL tactic: *EQ_TAC*, which is available in the *Misc* button section. The application of this tactic breaks the goal into 2 sub-goals. The formal reasoning about the first sub-goal requires us to apply the following HOL tactics: *DISCH_TAC, CONJ_TAC* and *POP_ASSUM MP_TAC* one by one. They are all available as click-able buttons in the *Misc* section.

Now we need to apply some existing theorems to discharge our proof goal and this step requires a database search. Thus, we use *DB-Search* to find a term similar in the current subgoal by writing the term $(a \wedge b ==> b)$ in the *DB-Search* term field and then clicking *Search*. The outcome of this action is provided in Fig. 2. The theorem *AND2_THM* matches the searched term and it is clear from its formal statement, given in Fig. 2, that rewriting with this theorem would discharge our proof sub-goal. This step can be performed by using the theorem, *AND2_THM*, with *REWRITE_TAC*, which is a standard rewriting tactic in HOL. g-HOL facilitates this rewriting step by allowing the user to highlight the theorem name from the console window and clicking *Copy to Shell*. This will add the theorem to the shell, (multiple theorems can be added to the shell as well). This selected theorem, or the set of theorems, can be applied to our proof goal by clicking on the *REWRITE_TAC* in g-HOL. Repeating the above-proof steps and using the *AND1_THM* intead of *AND2_THM* would discharge the second sub-goal. Once the goal is verified, a prompt appears asking the user to save the theorem. Once saved, the theorem becomes available in the *Define* section of g-HOL and thus it can be used in future proofs.



Script:

```
g`a/\b<=>b/\a`;

e (EQ_TAC);

e (DISCH_TAC);

e (CONJ_TAC);

e (POP_ASSUM MP_TAC);

DB.match["bool" ](Term`a /\ b ==> b`);
```

Goal: a/\b<=>b/\a

Console:

```
   ------------------------------------
     a /\ b

   2 subgoals
    : proof
- OK..
1 subgoal:
> val it =

   a /\ b ==> b

    : proof
- > val it = [(("bool", "AND2_THM"), (|- !t1 t2. t1 /\ t2 ==> t2, Thm))] :
  ((string * string) * (thm * class)) list
```

DB Search: bool

Term: a /\ b ==> b

Theorms: bool

Fig. 2. g-HOL Snap Shot

4 Conclusion

The paper describes a GUI for the HOL proof assistant. The main motivation of g-HOL is to facilitate learning and working with interactive theorem proving and thus pave the path for their usage in the industry. In order to evaluate the effectiveness of g-HOL, we used it in a classroom, of 60 under-graduate students of software engineering, as an alternative to the command line interface of HOL. Half of the class students were taught interactive theorem proving using the command-line interface of HOL while g-HOL was used for the other half. The students, using the g-HOL interface were found to take significantly more interest in the subject compared to the ones using the command-line interface. Moreover, after the same amount of training time, the g-HOL users were found to be about 4-times more-effective than their counterparts. The amount of syntax errors were also predictably much less for the g-HOL users. These statistics clearly indicate the effectiveness of the proposed ideas. We are currently working on enhancing the features of g-HOL and would appreciate suggestions and comments about our interface, which is available for download at g-HOL [2].

References

1. Hol proof assistant (2015), `hol.sourceforge.net/`
2. Arshad, F., Mehmood, H., Raza, F.: g-hol - a graphical user interface for the hol proof assistant (2015), `save.seecs.nust.edu.pk/projects/g-HOL/g-HOL.html`
3. Geuvers, H.: Proof assistants: History, ideas and future. In: Academy Proceedings in Engineering Sciences. vol. 34, pp. 3–25. Springer-Verlag (2009)
4. Hasan, O., Tahar, S.: Encyclopedia of Information Science and Technology, chap. Formal Verification Methods. IGI Global Pub (2014)
5. Oracle: Java documentation (2015), `docs.oracle.com/javase/tutorial/uiswing/`