Automated Theorem Proving

Customized Versions and Real World Application

Joan Yoon
College of Arts and Sciences
Lewis University
Romeoville, IL 60446
Email: joanhyoon@lewisu.edu

***Abstract*—This paper reviews two pieces of literature that offer a more customized type of automated theorem proving (ATP) to serve the priority needs of verification. I summarize the views of each in relation to simple ATP. According this to knowledge, I then provide a real world case by applying it to an instance at my current company. I look at the variables involved and quantify them, and evaluate how ATP could improve upon the measures assumed.**

I. Introduction

Theorem proving is considered a strong method of formal verification, but is less commonly endorsed for a project due to its accordingly higher complexities in use. With this regard, it is interesting to note the essential concepts of the papers reviewed, as they propose that ATP can be further justified - for the matter of checking and also to make sure it is more easily understandable and usable. I try to imagine ATP application in my workplace, and even consider the setups offered by Bibel et al and Leino to determine the average expected benefit of the method.

II. Literature Reviews

The following papers reviewed gave a view into ATP and the proposals that the researchers provide that would allegedly augment the standard ATP. Bibel et al combines classical and non-classical logics to adapt the ATP to the specific problem in question. Leino augments his ATP with satisfiability-modulo-theories (SMT), and uses Dafny as an example to argue how this would allow a stronger check on the mathematical reasoning portion and open up creativity to the user.

A. Bibel, et al: "Problem-Oriented Applications of Automated Theorem Proving"

Within formal verification methods, theorem proving is a deductive approach that usually is known to have more involvement of the user due to its level of understanding needed on the details of the system functions. However, the mathematical reasoning portion is able to be automated reasonably well, as it usually consists of precise forms. Automated Theorem Proving (ATP) assists with this by developing systems that simulate mathematical reasoning.

Bibel and his associates acknowledged that there were certain issues to be addressed in their approach to create an ideal situation for ATP to apply to. The first regards to the reason for high user involvement. There was typically difficulty in general understanding for the systems at onset because of its technicalities. Furthermore, of course ATP is best for formalized cases, but especially if the logic here is not in the classical first order logic (FOL), such an efficient implementation may not be available.

The Intellectics department of the Technical University in Darmstadt came up with the solutions of either a combined system that uses both classical and non-classical logics, or presenting the generated proofs in an application-oriented manner. Bibel et al embraced these solutions to develop an all-encompassing ATP system. Their main three areas of application were mathematics, programming, and planning. From these, a particular logic of classical, intuitionistic, or modal was selected to cover certain types of applications, and ultimately have the system cover all available types. The selection depended on the proof tasks to be solved that the user had extracted, and the use of generated proofs within each of the applications. In this proposed approach, Mathematica, NuPRL, and ISABELLE were possible uses.

The basic structure of the proof process of ATP-systems was reviewed "to describe the need for and the location and details of the extensions which enable the handling of different logics and of application-oriented proof presentations" [1]. Some input layer preprocesses a formula to be proved into "machine language" (i.e. input transformation step), and once proved (by inference machines), post-processing occurs to return the original input language as well as make it readable for the user (proof presentation). This process is flexible and is also not completely set on the level of advancement; it has both advantages and disadvantages to having a sophisticated input transformation and a simple inference machine, or the opposite case.

Finally, the report was detailed with how to treat semantical meta-knowledge originating from non-classical logics. They looked at formalizing in terms of Kripke-Semantics, or by using a coprocessor to check for each classical inference. A uniform framework was developed from these approaches for non-normal form theorem proving, consisting "of a two step algorithm, i.e. a uniform procedure finding the proofs and a uniform transformation procedure converting these proofs into sequent-style systems" [1].

Expanding on this research, Bibel and associates have many projects underway. The main purpose of the future work is for generalization and efficiency for similar technologies. They want to focus on the core aspects of a program and its design, and have the mechanisms built in to ensure accuracy and correctness properly. Their experiments and their according examples from applications will provide additional insight into whether non-normal form proof techniques are up to par or even have a greater use over conventional clause-form theorem provers. I can only imagine the actual power that such an ATP system that orients to a specific problem would produce, especially in the well-rounded manner that Bibel et al describes.

B. Leino: "Automating Theorem Proving with SMT"

Leino seemed to take on a somewhat narrower perspective on ATP by tying it with SMT, especially Dafny. SMT is viewed as auto active, or "a mix of automatic decision procedures and user interaction at the program" [2], moving the user from a formula level to a program level. Supposedly tools based on SMT focus on mathematics than on programs, although their primary purpose is for verifying programs. This means, however, that many formal aspects can be processed automatically to relieve the user of some of this load and instead be more engaged in the creative functions of the proofs.

The paper presented their ideal of having all ATP systems to be loaded with SMT. Dafny is the main representation in question, chosen for its features like inductive, co-inductive, and declarative proofs, which are traditionally found only in interactive proof assistants. Within Dafny, a type and a function was defined, and then a lemma was stated and proved by induction. This was turned infinite, so that the "filter" function could be formed with the combination of inductive and co-inductive features; this combination ultimately applies to prove a theory about filters. With these expressions, Leino presented his ideal of a future that has ATP systems loaded with SMT solving.

The outlook that Leino produces is something that I can connect with, as I can only see a growing interest in ATP. From here, it would seem that many researchers would want to find further ways to improve upon the method, of which SMT would align. This boosters the performance of ATP and also allows for ease of the user to control other factors of the program, such as in design structuring.

III. Application: Legal Tech

A. Overview

I recently entered the legal tech industry. Somehow, these two industries were brought together even though they seem to be opposite in working perspectives. Legal workers are used to traditions, for the purpose of assert their case statements and in their general functions as well. Reviewing and producing their case files consisting of hundreds of thousands of documents was a manual process, requiring multiple people and long extensions of time to go over. On the other hand, the technology industry is used to being very innovative and moving at a faster pace, always seeking to be quicker and more efficient. What our company offers is to improve productivity for people in the legal industry with a program that automates much of their processing. Although the adoption process may meet some initial skepticism, legal tech is growing.

We offer a more streamlined workflow with a highly user friendly interface. This is a major selling point for the skeptics, the lawyers, paralegals, etcetera out there. Smart algorithms and formulas remove trivialities and essentially complete the crux of the labor. This means that more than half of their work is cut. What would usually take weeks and months to complete, can be accomplished with simple steps and rather take only a few days or even mere hours.

B. Uploading

This is obviously a very powerful tool. However, once these tools are placed in the customers hands, they can become "greedy" and ask for more features. A faster, stronger, more customized program - their requests are endless. One of the main demands is regarding increasing the speed for specific tasks, such as for uploads or search results. Due to the fact that the latter is most dependent on internet connection, lets consider upload speed.

The upload process is broken up into the following parts:

- Transfer: The part of the process where the data is copying over from the original file source to our secure and encrypted storage. Since the data is not yet in our system at this point, we do not have much control over its speed here.
- Processing: This is where the magic takes place. During processing, languages are detected, the transferred data gets scanned for viruses, text is extracted or OCR is applied, and these documents are rendered to web-viewable formats. This list mentioned previously is not all-inclusive.
- Post-processing: The program ties up ends by applying any applicable QC tags, analyzing duplicates, indexing text into the search engine, and syncing with previous uploads.

It is clear to see that processing (processing and post-processing) takes the longest amount of time. Two sets 1 GB files can have very different types and amounts of data that could significantly alter the time it takes to process them. As a result, it is difficult to determine an average time; our rule-of-thumb is 2 GB per hour per project, or about 5,000 documents per hour, depending on how compact the container is. During this, to ensure optimal transfer conditions, we make sure to recommend a wired connection, a local drive being the transfer source (as opposed to a network or shared drive), and sleep mode set to off to prevent interruption.

C. Variables

We are currently looking at decreasing the upload time, especially the processing portion. A variable involved for this would be the documents. Specifically, this points at the number of documents that relate to the overall size of the upload, and the types of documents. The documents can be excel spreadsheets, word documents, email messages, images, etc.). As mentioned previously, this would impact the level of work needed to be done on each document so that the filtering and review process is easy for the reviewer.

Another variable would be the number of users. Please note that this program is cloud-based on AWS to scale, but the technical reality is that there may still be a bottleneck with the limits of virtual workers. At that point, it would then be cost in terms of expanding the hosting service to increase the resource. We will disregard that last possibility at this time so that we can measure the traffic of users in correlation to the program's upload speed.

The last variable is the internet connection. This is one of the more negligible aspects. We do have the aforementioned recommendations, but of course this may be ignored so we have less control over this factor. We will see it as being a considerably small contributor in actually affecting the actual speed in this scenario.

D. ATP Application

The main benefit of applying ATP would be applying it at all as a system verifier. Although we do make use of testing and other manual bug checking, the scope is not as wide or detailed as would a method like ATP. The previous methods could still be vulnerable to human errors and consumes much resources (i.e. manpower, time). With the base of the program already set and being that it has already been established with various features, the most difficult work has been done. The next most difficult hurdle is defining the proper characters and functions to allow ATP to prove them by layers, induction and co-induction. However, the mathematical reasoning and automated portions of the proofs would be helpful in the long-run in saving some of those resources and ensure accuracy - especially as further advancements are rolled out.

Due to the fact that the processing stage has the most activity with the various tasks of OCR, virus scanning, and such, ATP would benefit in this area. I can assume that ATP can apply to verifying the theorems maintaining the structure and smooth flow of uploading. ATP would confirm the mathematics of the background algorithms dictating the tasks. We can also format the structure to alert on any abnormal behaviors of the tasks and the speed. With a linked system report, the ATP here can then verify that the program is running correctly and smoothly, on an individual activity basis as well as overall. I am assuming that in this case, it would follow inferences from supported file types, tagging them appropriately in addition to any other marking and labeling, based on the formulas established. If there are no issues run into here, that would mean the checks were successful.

With such efficient tools to check on the variables that contribute to upload speed, I can envision the real potential of the ATP systems that Bibel et al and Leino argue for. For instance, an ATP loaded with SMT would seemingly filter and prove out any possibilities for error in our program. An ATP that further caters to a specific problem and uses a combination of approaches to tackle them is very valuable to all programmers and users since it would target the core of the issues directly.

IV. Conclusion

Theorem proving is usually known for being more complex and thus requiring more user involvement. However, Bibel et al and Leino contend with customized versions of ATP that would supposedly make it easier on the user to initialize, and also more efficient to implement. This gives evidence that ATP is a current hot topic that will continue to be researched for greater advancement potential. Incorporating these automated methods in our regular maintenance schedules would definitely improve the quality of our technological performance, and be a necessary smart update.

References

[1] Bibel, W., Korn, D., Kreitz, C., Schmitt, S. *Problem-Oriented Applications of Automated Theorem Proving*. Darmstadt, Germany: Technical University in Darmstadt.

[2] Leino, K. (2013, May). *Automating Theorem Proving with SMT*. Redmond, WA: Microsoft Research.