Automated Theorem Proving

Processes and Applications

Brian Dutremble
CPSC 59700
Lewis University
Romeoville, IL, USA
Brian.dutremble@lewisu.edu

Abstract— **Automated theorem proving attempts to simulate mathematical reasoning via deductive systems. ATP has many applications including everything from synthesizing computer programs from a given specification to automating control of intelligent agents within a given environment. SMT provers are a subset of automated theorem provers that feature an especially high level of automation and that have become much more powerful recently. Given the benefits of ATP, I believe that such a tool could certainly be utilized to improve the behavior of the database tools at my place of work.**

Keywords—ATP; automated theorem proving; practical application;

1 Applications of Automated Theorem Proving

1.1 Introduction

Mathematical reasoning lends itself to automation because it is the most precise form of human problem solving. Automated Theorem Proving (ATP) attempts to simulate mathematical reasoning via deductive systems. ATP has many applications including synthesizing computer programs from a given specification and automating the control of intelligent agents within a given environment using a resource-sensitive logic. ATP is useful whenever a problem can be formalized in a logical language.

There are some issues with utilizing ATP. For example, if the logic of the problem is different from first order logic, there may not be an available implementation. Additionally, the proofs generated by ATP systems may have a very technical look that cannot be understood by users and will often need to be transformed before being able to be read. The Intellectics department of the Technical University in Darmstadt has come up with solutions for these problems, having created systems that are better able to deal with both classical and non-classical logics while producing proofs that are application-oriented.

The Intellectics approach is divided into the three areas of application: mathematics, programming, and planning. Three well-known systems are provided, one for each area (Mathematica, NuPRL, and ISABELLE).

For each of the areas, a logic is selected (FOL, intuitionistic logic, modal logic). Classical logic covers programs that utilize mathematical theorem inputs, intuitionistic logic refers to applications that require constructive proofs, and modal logics are applications that require concepts like time and belief. The Intellectics approach aims to create a system that deals with all of these logics in a combined ATP system.

1.2 Structuring the Process of Theorem Proving

Each ATP system has an inference machine at its core. An input must first be transformed from the language of the application to one that the inference machine can understand. Once able to grasp the input, the inference machine does the actual exploration of the search space spanned by the input's underlying logic. The inference machine then generates a machine-oriented result that is transformed into a more readable form. This structure of proof process allows for a great deal of flexibility in the design of problem-specific ATP systems.

1.3 Combining Classical and Non-Classical Theorem Proving

Seen as a combination between classical reasoning and reasoning about the semantic meta-knowledge for a given logic, ATP in non-classical logics utilizes formalizations organized in terms of Kripke-Semantics, which deal with the notion of possible worlds. Reasoning in non-classical logics is, basically, classical reasoning around the first order fragment of the non-classical language within individual possible worlds combined with reasoning around modal operators and the whole set of possible worlds. The reasoning in non-classical logics is usually processed within the inference machine while the second might take place elsewhere. There are two main approaches to this additional process. The first one takes place at the time of processing, formalizing the Kripke-Semantics notions in terms of classical logic. The second approach checks each classical inference step to see if it is understandable in the modal context.

2 Automated Theorem Proving with SMT

Mechanized proof assistants are becoming popular tools for formalizing and proving theorems about mathematics, computer programs, and programming-language semantics. These proof assistants often use satisfiability-modulo-theories (SMT) solvers as subroutines because SMT solvers provide a high level of automation and have become more powerful recently. General proof assistants can be used to verify the correctness of computer programs, but there are also some verification tools specifically designed for this purpose. These verification tools use an SMT solver as a reasoning engine and move the user's interaction from the formula level to the program level. This sort of verifier is referred to as 'auto-active' because it is a blend of automation and user interaction at the program level.

Dafny, a popular auto-active program verifier, has features that have been previously only available in interactive proof assistants. These features include inductive and co-inductive definitions, proofs by induction and by co-induction, as well as human-readable proofs. Dafny and other tools like it are able to minimize the amount of effort required from the user by utilizing automation and most of their responses are near-instant. The higher degree of automation that will no doubt be possible in the years to come will enable tools like Dafny to have much richer sets of features and will reduce the currently large trusted computing base for SMT-based verifiers.

3 Practical Application of ATP

Automated theorem proving would be very useful in testing a system that would accelerate the rate at which the tool we use to inspect our data can be refreshed. It currently takes over twenty-four hours for changes to our database to be registered by CALM, our main data visualization tool. CALM is an extremely useful tool but often its information is too outdated to be useful in situations where a quick turnaround is needed. Its refresh rate is so slow because changes to the database are gathered for a twenty-four-hour period then added to the database all at once. It is done this way to prevent errors that might result from many concurrent attempts to edit a portion of the database. Data fidelity is prioritized over all else, for good reason, but the length of time required for the tool's view to update means that some of our quotes cannot be verified prior to deployment. For this reason it would be preferable to speed up the refresh process.

Accelerating the refresh process would require that we take into account the following variables: total size of database, number of changes made, areas affected by the changes, order of changes, and types of changes (among others).

An auto-active program verifier could be utilized to verify whichever algorithms could be designed to minimize the refresh time of the CALM tool. The CALM environment is large but its parameters are very clearly defined and the characteristics of the data it holds are limited. There are also very few outside factors that can affect the performance of the system. The very simple logic of the relationships between assets in the database lends itself to algorithms for its cataloguing and processing. The team responsible for designing those algorithms and managing the database could certainly benefit from a tool like Dafny that would automate much of the program checking process and enable many non-ideal steps to be eliminated from the refresh process. Fidelity being such a high priority means that eliminating errors in every part of the system is paramount, and that verification is an essential step in the process of ensuring that the system remains error-free. Automated theorem provers are particularly attractive in this case because it allows the company to minimize the time and resources involved in testing refresh algorithms and ultimately minimize the time needed to refresh the system.

[1] K. Rustan, M. Leino, "Automating Theorem Proving with SMT," Microsoft Research, Manuscript KRML 234, 19 May 2013, pp.1-15.

[2] W. Bibel, D. Korn, C. Kreitz, S. Schmitt, "Problem-Oriented Applications of Automated Theorem Proving," Fachgebiet Intellektik, Fachbereich Informatik Technische Hochschule Darmstadt, pp.1-21.