Formal Verification: Automated Theorem Proving

Edward Escobedo
Lewis University
CPSC-59700 Week 5 Assignment
One University Parkway, Romeoville, IL 60446
edwardescobedo@lewisu.edu

***Abstract*— Automated Theorem Proving (ATP) consists of the designing of computer programs that show a statement in a logical consequence of a set of declarations which are the axioms and hypothesis. These systems can be used in a wide variety of domains. ATP is a technology very useful in situations where a clear and concise thinking expert can interact with a powerful tool to help solve deep and exciting problems. ATP has been used successfully in many fields. This paper will address two readings assigned which are Problem-Oriented Applications of Automated Theorem Proving which details an overview of an approach to developing a coherent ATP system and the next text is the Automated Theorem Proving with SMT which discusses the power and automation offered by modern satisfiability modulo theories solvers which changed the landscape of mechanized formal theorem proving. Furthermore, this paper will address a significant problem that is the Certifying and Accreditation Web services used for reviewing and deciding risk associated with program wishing to receive an Authorization to Operate.**

Keywords— First Order of Logic, Structuring Process, Classical and Non-Classical Proving, Inductive Proof, Co-recursion Proof, Coinductive Proof, Filter Function, Property of Filter,

I. Introduction

Automated Theorem Proving (ATP) is a computer program that displays that a statement is a logical consequence of a set of statements. ATP systems can be used in a significant number of ways, such as mathematicians using the axiom of group theory to prove the conjecture that groups of orders two are cumulative; management can create axioms that describe how businesses grow and interact and demonstrate the employment between age groups and impact. Tasks like these can be performed by an ATP system, given an appropriate formulation of the issues as axioms, hypothesis, and conjecture [6]. Most importantly, ATP systems are embedded as components of much larger, complex software systems, and within this context, the ATP autonomously solves subproblems that are created by the overall system [6]. To build a useful ATP system, several considerations must be carefully addressed and independent about each other and addressed synergetic. These factors include the choice of logic used to represent the problems, the calculus used for deduction, the programming language of the ATP system, data structures the will be utilized and statements deduced by the overall scheme for controlling the deduction steps and the heuristics.

This paper will focus on the two readings assigned for this week that are Problem-Oriented Applications of Automated Theorem Proving, which provides an overview of an approach to development coherent ATP-systems, which deal with a variety of logics with different applications, tailored in various ways. The second paper discussed is the Automated Theorem Proving with SMT, which summarizes the power and automation offered by new satisfiability-modulo-theories (SMT) solvers that are changing the landscape for mechanized formal theorem proving. Lastly, this paper will discuss the effect of social engineering in the risk determination process seen from the security control assessor viewpoint. Of course, social engineering has two faces, technology, and human components. The focus will be on the technology component and discuss variables that can be used along with ATP for software and hardware engineering to ensure the technology portion is designed properly to guarantee the security enforcement.

II. Problem-Oriented Applications of AutomatedTheorem Proving

This paper begins discussing the differences between logical and mathematical reasoning and how particularly correct mathematical logic is to human problem solving because the precision it lends itself to automation easier than thinking in general. In the field of ATP, developing deductive systems, which simulate mathematical reasoning, is the goal. ATP is useful if the problem that is to be solved can be formalized in a logical equation for which a proof calculus and efficient implementation is required. However, as we, all know; efficient implementation is most likely not happening. This can occur when the logic is different from the first order of logic (FOL) which happens when applications lend themselves to other formalizations other than FOL [2] [7]. Another problem arises with ATP are when the actual proofs generated by ATP systems have a distinctive look. All this means is that before being understood, the application will need to be transformed into a readable form

A. *Structuring the Process*

The section discusses the core of each ATP system I the inference machine which a microprocessor for theorem proving. Formulas are provided and verified by a preprocessed input layer to transform it into the machine language of the inference machine, and the output of the proof is the post process to suit the original input language and for the result readable to the users. The three different tasks need to be considered with the problems of providing the mechanisms are as follows [2][3]:

- Input Transformations – The interface between the application oriented and the inference machine-oriented languages.
- Inference Machines – This is where the actual work is done in the search space consisting of small sets of inference rules, which are applied to the formula according to strategy.
- Proof Presentation - The logical interface between the calculus of the inference machine and the application. The goal is the putting the result in a readable form.

B. *Combining Classical and Non-Classical Theorem Proving*

This next section deals with the classical and non-classical proving and how ATP in non-classical logic is a combination of reasoning and traditional reasoning about the semantical properties with the logic that needs to be formalized Kripke-Semantics [3]. The idea here is that semantics is the notion of possible worlds which means they represent different interpretations of the predicate symbols and logical language.

The next section discusses classical theorem proving for constructive logs. Furthermore, explicit modal logics, intuitionistic logic does not include specific users to denote notions. Instead, it uses logical connectives that are interpreted in a theory of knowledge way [7]. A disadvantage this approach is the potential undecidability of the formula since it is formulated in full first order logic [2].

A standardized framework for Non-Normal Form Theorem proving is another method describing the additional reasoning achieved by an extension of the standard machine such as a specialized co-processor. The differences between classical routine are proving mostly consists of demonstrating techniques within a non-normal environment. This framework created a two-step algorithm and a normal transformation procedure converting the proofs into sequent style systems. The development of the algorithm is based on the uniformity of the environment [7].

III. Automating Theorem Proving with SMT

The second paper discussed in the readings is the ATP Satisfiability Modulo Theories (SMT) that refers to the issues of determining a first order formula is satisfiability with regards to logical theory. They are routinely used as the engine behind verification. However, systems are usually developed and designed at a greater level than the Boolean level, and translation to Boolean logic can be expensive. An ultimate goal of research in Satisfiability Modulo Theories (SMT) is to create verification engines that can reason natively at a greater level of abstraction, while still retaining the speed and automation of today’s Boolean engines.

Boolean logic is used by SAT solvers, and SMT solvers use first order logic. The language entails Boolean operations of Boolean logic, but instead of propositional variables, more complicated expressions involving constant, and predicate symbols are used [3].

IV. Social Engineering and ATP

If you look at most publications and websites, social engineering is considered a non-technical threat cyber attackers use that heavily rely on human interaction and often involves tricking people into violating cyber security policies and practices. Its success depends on techniques the attackers use to manipulate victims into performing actions or providing confidential information. Today, social engineering is recognized as the greatest threat facing businesses [5]. When a successful attack occurs, attackers able to gain authorized access to systems and information. Some of the most common attacks include baiting, phishing, pretexting, spear phishing, and tailgating. As a security risk assessor for the Navy, my job details reviewing hardware and software programs for its cyber security posture and determine the risk associated. While the technical aspect of this is straight forward, the human dimension of this is difficult. We have been asked to evaluate a system and include social engineering into the risk determination.

My job consists of a checklist, which if followed, will cover the essential aspects of conducting a risk assessment of a system. The problem we now face is how do we include social engineering into a checklist? How much weight do we put into a systems social engineering risk? This inlays the issue we face. I will look at both the technical and human aspects of this threat and show how if we can develop and design a system correctly and use tools such as ATP we can minimize the attack vector although there is no chance to eliminate it complete due to the human factor.

A. *Social Engineering Variables*

First, the human element needs to be a necessary part of Information Security. Historically, people of been the source of security issues in organizations. Insider threats are the first aspect that needs to be addressed. Employees are the primary issue here [5]. So of the most recent data breaches were established by employees [4]. Research confirms that the human factor is the weakest link in the IT security chain. An example, it has been demonstrated that offering a reward of US $1 is enough for a significant percentage of users to download unauthorized software, ignoring the usual security warnings [4].

Current approaches to IT security and risk management tend to underestimate and even ignore the human factor in the assessment tools, processes, models and legal structure. Since involving employees inside an evaluation is a relatively innovative approach and is considered risky, planning evaluation in a proper way assumes an important role. First, IT and security departments are not the sole actors to define the evaluation, because people are the targets. Therefore, it is important to include all the relevant stakeholders, such as human resources, legal and communications departments, to explain the threats, share the objectives, define the scope of the assessment and obtain commitment [4]. Moreover, several ethical concerns and requirements need to be considered when performing an assessment on the human factor.

From an information security governance standpoint, the end goal of including the human factor significant ability evaluation activities should be to identify proper mitigation. The most effective countermeasures against social engineering risk are awareness and training, which can help improve the security culture of employees. Unfortunately, traditional awareness and training programs are not always effective. Finding the right way to raise awareness is a key factor. The most likely attempts are related to using videos, infographics to stimulate people. Moreover, ramification is one of the most promising trends. Rewards, social engagement, and direct feedback can help, even if the right strategy depends on different factors that must be carefully explored [5].

Ok so now that the human element has been addressed, we can now look at how the technology portion can help mitigate the risk of social engineering. In beginning an assessment of a system, how do we look at the risk determination from a technology standpoint? Of course, designing and developing systems with security in mind instead of an add-on is a start.

Software verification is a requirement that is slowly being realized using ATP technology. Three examples are given here, while much more are indexed using formal methods. The Karlsruhe Interactive Verifier (KIV) which was designed as an experimental program for interactive program verification and has since been used to verify a range of software applications successfully. The starting point is a commercial CASE tool, which is augmented by capabilities for formal specification and verification. The goal is to make the verification process transparent for the user on the simple object-oriented model. Hardware verification is the largest application of ATP. IBM, Intel, and Motorola are among the companies that employ ATP technology for verification. Formal methods are good examples of the use of ATP systems for hardware verification.

B. *Other Variables*

Now that the typical software and hardware design questions are addressed and how we can use ATP to develop a secure system, I would like to address some variables that a difficult to address in a risk determination are questions I ask myself every day on my job

- Security Polices in place (username and passwords, is dual factor authentication used, PKI)
- The number of administrators and ordinary users.
- Complexity of the system (Is it an entire enclave or an application)
- Biometrics
- Physical security where the system or application is located.

Addressing these considerations and determining how to include it in a risk determination is the toughest part. With the rise with this treat, first discussing the design and development is the beginning them comes the human aspect and then how to address the many other variables associated with trying to determination risk and how social engineering play in that decision.

V. Conclusion

Practical application of an automated theorem prover poses entirely different requirements on the prover than in traditional mathematical problems or benchmarks. The prover should be able of solving inductive problems, handling logic in an efficient way. The ATP must be able to process large formulas that require simplification and selection of axioms. When a proof is found, the automated system should return a proof (human readable) or answer substitutions. This paper summarized the two readings assigned for this week. It discussed problem-oriented applications of APT and Satisfiability Modulo Theories. We then addressed an area of cyber security in my job that no standard information exists on conducting risk assessments with social engineering. System design and development with security in mind is the start. They are serval different APT tools that can be used for this process. Nevertheless, what about the some of the other variables list above? How is the human aspect factored in? These are decisions that further research and understanding of the systems involved are needed.

References

[1] B.I. Dahn, J. Gehne, T. Honigmann, A. WolfIntegration of automated and interactive theorem provingW.W. McCune (Ed.), Proc. 14th International Conference on Automated Deduction, Townsville, Australia, Lecture Notes in Artificial Intelligence, 1249, Springer, Berlin (1997), pp. 57-60

[2] Bonichon, R. (2006, Autumn). An Extensible Automated Theorem Prover Producing Checkable Proofs. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-75560-9_13

[3] Koblitz, N. (2008, February 5). Another look at automated theorem-proving. Retrieved from https://www.degruyter.com/view/j/jmc.2007.1.issue-4/jmc.2007.020/jmc.2007.020.xml

[4] Puricelli, A. (2015, Summer). The Underestimated Social Engineering Threat in IT Security Governance and Management. Retrieved from http://www.citationmachine.net/apa/cite-a-website/manual

[5] Valentin, J. (n.d.). How to Plan a Social Engineering Assessment. Retrieved March 1, 207, from http://resources.infosecinstitute.com/plan-social-engineering-assessment/#gref

[6] Sutcliffe, Cliff. "Automated Theorem Proving." N.p., Apr. 2010. Web. 2 May 2017.

[7] Schumann, J. M. (2001, May). Automated Theorem Proving. Retrieved fromhttps://books.google.com/books?id=GQdJCAAAQBAJ&pg=PT249&lpg=PT249&dq=Automated theorem proving conclusions&source=bl&ots=gGh7Of-ahi&sig=lRQfMSQ-CrpB11C8ld_HJ5iH07k&hl=en&sa=X&ved=0ahUKEwiX66eS6tLTAhVK7GMKHQH7DSkQ6AEITDAH#v=onepage&q=Automated%20theorem%20proving%20conclusions&f=false