Final Paper – Evaluation of Automated Theorem Provers

Anna Baglione
CSE 3521: Introduction to Artificial Intelligence
May 2015

Life may one day be classified as an improvable theorem – or, rather, a collection of many smaller theorems, some provable, some not. Mathematicians, philosophers, and great thinkers throughout history have dedicated years (and sometimes their entire lives) to proving theorems. These theorems have tested the imaginative corners – and often the brilliant limits – of the human mind: Can numbers be "countably infinite"? How many colors are required to color a graph? How can points in space be best described?

With the dawn of computing, devising methods to prove or disprove theorems became significantly easier. Hungry to solve life's pressing questions, people created massive software systems to take on artificial intelligence tasks that would be impossible (or nearly impossible) to complete by hand. These tasks have spanned a wide variety of domains, from computer processor chips to medical devices and even board games.

This paper presents three distinct case studies of the use of automated theorem proving systems. Specifically, it examines the use of: ACL2, used by AMD following the infamous Intel Pentium bug; LP, used to investigate software errors responsible for the Therac 25 medical incidents; and VICTOR, used to solve the popular board game Connect Four.

Case Study 1

Computer hardware and software designers will likely be the first to assert that accuracy in any calculation is critical. Even minute errors can have mass-scale effects, as the notoriously-expensive and amusingly inconsistent Intel Pentium "FDIV" bug of the 1990s confirms.

Professor Emeritus of Mathematics Thomas Nicely discovered the bug when he was conducting research on prime numbers. He claimed that the Pentium chip was performing integer division incorrectly, resulting in a rounding error, and this claim was later confirmed by other members of the research community (Moler 1). Cleve Moler, who was then working at Mathworks, described in his article "A Tale of Two Numbers" (3) how the error had resulted from missing lookup table entries.

While this may have seemed a "careless" error on Intel's part, the research community was perhaps more concerned with Intel's initial downplaying of the error's consequences.. Introduce imprecision and consistency in the calculations that contribute to scientific and mathematical discoveries disappears. This was unacceptable, to the research community at large. Intel's fix – replacing all defective processors – has become the standard historical of how one small hardware mistake can cost a great deal of money.

Automated theorem provers can be useful in computer hardware verification. "A Computational Logic for Applicative Common Lisp 2", also known as ACL2, is one such prover. ACL2, based off of the theorem prover Nqthm is first order logic-based and takes common LISP formulae as input. (Brock, Kauffman, and Moore 1-4). ACL2's approach, when generating a proof, is to either directly simplify a formula or "[pass] it to the next proof technique" (with induction as a last resort) (4).

In their brief introduction to ACL2, Kauffman and Moore emphasize the prover's reliance on some human assistance, asserting that it is not necessarily fully automatic and often needs to have knowledge of additional proofs prior to drawing new conclusions (4). This is in keeping with the general idea of a knowledge base. A very helpful and related aspect of ACL2 is that of books "books", which are collections of important mathematical lemmas the prover system can load into memory for reference .

Following Intel's high profile FDIV error, fellow computer chip maker and Intel rival Advanced Micro Devices (AMD) employed the use of the ACL2 prover to verify the new 5K86 chip prior to its release. In their case study, Brock, Kauffman and Moore provide details of their successful proof that the chip's floating point division was indeed correct – an endeavor that included "over 1600 definitions and lemmas" related to formal mathematical logic (14). The authors thus illustrated that, while automated theorem provers are certainly powerful, they still require an extensive amount of human effort and support to reach desired conclusions.

Case Study 2

Automated theorem provers aren't simply restricted to proofs of correctness. On the contrary, such systems can be used to identify critical flaws in software or hardware. For both the creators and patients of the Therac 25, a medical machine used in radiation treatments for cancer in the 1980s, flawed software code had devastating consequences.

Nancy Leveson's investigation into the horrific Therac 25 incidents repeatedly asserts that, unlike its predecessors, the Therac 25 machine "[relied] more on software" than on hardware (3). A lack of key hardware safety features present in earlier versions of the machine allowed major, previously-undetected software bugs to surface – specifically, bugs, which enabled machine operator to accidentally administer massive overdoses of radiation. These bugs proved fatal for six patients in the Therac 25's tumultuous lifetime on the market.

Leveson highlights several key software issues present in the machine's implementation. She discussed many flawed aspects of the code, including its lack of adequate synchronization, memory atomicity violations, and various race conditions (24). Over the course of her discussion, Levison makes clear that the Therac 25 "borrowed" and "modified" code present in its predecessor machines (22) to create a piecemeal, untested, and too-trusted Frankenstein lurking beneath the hardware.

Muffy Calder (nee Thomas) decided, in her own case study, to investigate a specific software error present in the Therac 25 that had been responsible for so many of the documented deaths – the so called "editing problem." This problem arose when the machine operator attempted to alter treatment data for a patient (specifically, data related to the magnets used in the machine to help administer the radiation) using the up arrow key on the machine's keyboard (Thomas 3).

Long after the documented cases of patient deaths, it was discovered that, within a very small window of time (8 seconds or less), editing this data using the arrow key did not in fact edit the data in the machine (3). This was due to an issue with the settings of a specific variable and its location within the code. To prove then, that the software for the Therac was *incorrect*, Calder used the LP Theorem Prover.

LP, better known as "Larch Prover", is a Theorem Prover for First Order Logic, developed by Stephen J. Garland and John V. Guttag. By its very design, it accounts for human error: "Its design is based on the assumption that initial attempts to state conjectures correctly, and then to prove them, usually fail." (Garland and Guttag 1) LP not meant to act as standalone system, but rather as a kind of assistive debugger (1). It is known to be good for regression testing ("LP: The Larch Prover..."). Mostly, however, LP is known for simplifying the proof process; it does not use heuristics or allow for user-defined "proof tactics" but rather "provides a set of standard tactics and simple mechanisms for controlling the application of these tactics." (2)

LP's assistive and simplified nature made it suitable for Calder's analysis of the Therac 25 software. In the case study, Calder first walks through her methods for restricting her analysis to just the necessary information regarding the Therac's editing problem. She details pseudocode, illustrating the idea that, if an edit occurred, the code was to return to a specified point of execution. She then proceeds to use LP to prove the software's incorrectness by showing how dependence on a single variable causes other edits to be discarded or ignored.

Calder's study highlights LP's usefulness by showing how LP can be used to work backwards to find the exact source of error. What is perhaps more impactful than the actual proof, however, is her subsequent proposal of corrections to the original Therac 25 software routine the first caused the error. Thus, she shows that automated theorem provers like LP can be especially useful when used for analysis and debugging purposes.

Case Study 3

Despite widespread use in more serious (perhaps what some might deem comparatively "boring" applications), automated theorem provers have managed to find their way into the world of much-loved classic board games - thus "proving" that research *can* sometimes be all fun and games.

Consider Connect Four, known by many other names throughout history but finally popularized in plastic form by Hasbro (then Milton Bradley) in the 1970s. The game, designed

for two players, consists of a 7x6 vertical board with circular slots. Players take turns dropping colored tokens (red or yellow checkers, depending on the player) into the slots. The goal is evident in the game's name: to "connect" four pieces across a row, a column, or a diagonal (Connect Four instructions, Hasbro.com).

Whether named to boldly convey the spirit of triumph or to secretly pay homage to its author, the VICTOR program confirmed what many seasoned Connect Four players might claim, after years of experience: that one can win the game by simply going first. Masters Candidate Victor Allis of Vrije Universiteit published his thesis in 1988 on solving the popular game of Connect Four by using a system called VICTOR. Allis's wanted to explore how master chess players might go about winning a game of chess. However, due to the complexity of chess and the lack of systems for evaluating chess, at the time of his writing, Allis sought to take a game which is less complex than chess, but more complex than can be dealt with using brute force only" (Allis 5) and try to solve it instead. Thus, he took on the challenge of Connect Four.

VICTOR is an automated theorem prover written in C. Allis's basic approach using VICTOR was to build up a knowledge base of "move rules", try to solve smaller boards, and, finally solve the full-sized (7x6) board. Borrowing ideas from well-known problems like the 8-queens puzzle and from lesser-known problems like as the "wumpus world" introduced by Stuart Russell and Peter Norvig Allis used the ideas of "threats" and "tactics" to show how VICTOR accounts for many different environment variables.

VICTOR uses nine main "strategic rules" to analyze a given position. Essentially, VICTOR's process is this: Given a current board description, find and apply the rules "in all possible instances" (58) relating to this current description, find which of these instantiations will lead to a state of four connected tokens for the given player, and record these instantiations.

What made VICTOR both interesting and sophisticated, at its inception, is its ability to analyze across problem solutions. For example, it stores multiple solutions for the same problem, and uses adjacency matrices to see how those solutions can interact with each other (59). Allis wanted to use these features to try to find "a set of nodes which can be used to solve all problems" (59).

Key to Allis's analysis using VICTOR was the idea of control of "Zugzwang", a term best described as being in a position of power so as to force the other player to make a bad move (25). Allis was able to use this idea, along with the nine rules he developed, to show that having a position of power from the very beginning can be critical to winning the game. He ultimately concluded in his study that, for the standard-size (7x6) board, a given player (white) "can always win" if that player starts the game from a given position (59).

Besides confirming that the first move in a game can indeed determine the final outcome, Allis came to an interesting conclusion regarding the time complexity of VICTOR's algorithm: "The recursive search for the independent set is potentially NP-complete. Given the fact that the number of solutions on the 7 x 6 board can be as much as several hundreds, one could expect that the recursive search would be the bottleneck of VICTOR during execution. This is not the case." (59-60). Allison speculates that this unexpected happy outcome for time complexity results from

certain positions that, in reality, have few very few solution options ("only one or two"). While this may be considered a natural feature of the game itself, the fact that VICTOR is able to identify positions with both few and many solution options shows its sophistication for its time.

Conclusions

Each of the three case studies presented in this paper highlights a different area of artificial intelligence that can benefit from the use of automated theorem provers.

The first case, a study on the use of ACL2, shows that hardware verification can be made significantly easier, provided human theorem testers are willing to put in the tedious upfront work to prepare the theorem prover to derive a proof from existing lemmas. More importantly, the case illustrated how hardware testing prior to release can help companies avoid expensive implementation mistakes similar to Intel's division error.

The second case, a study on the use of LP, highlighted the potentially deadly implications of using untested software on hardware that does not employ safety mechanisms. Those involved in the analysis of the software errors showed how race conditions, atomicity violations, and inadequate synchronization can all contribute to undetected errors. Muffy Calder took her analysis a step further by showing how theorem provers like LP can be useful for showing incorrectness in software.

The third and final case, a study on the use of VICTOR, showed how theorem provers can be used to enumerate solution possibilities for complex games, based on a player's current position. The results of the study were notable for showing that certain positions can be advantageous and for providing an example of how one might use a theorem prover to analyze a more complex game like chess.

Across all of these cases, the point one must take away is that automated theorem provers make the tasks of analyzing software, hardware, and even games a little easier. In a more romantic sense, one could argue that automated theorem provers are the artificial intelligence community's current "solution" to the ever-growing pool life's unanswered questions. At any given time, we have a relatively small subset of cases on which to run sophisticated theorem provers and solve particular problems. But in the bigger picture, theorem provers really just give imperfect human beings a method to decipher our own creative madness. And in that, they themselves become somewhat of a technological mystery.

Works Cited

Brock, Bishop, Matt Kauffman, and J. Strother Moore. *ACL2 Theorems about Commercial Microprocessors.* Tech. Computational Logic, Inc., n.d. Web. 5 May 2015. <https://www.cs.utexas.edu/users/moore/publications/bkm96.pdf>.

Allis, Victor. "A Knowledge-based Approach of Connect-Four: The Game Is Solved: White Wins." Thesis. Vrije Universiteit, Amsterdam, The Netherlands, 1988. *A Knowledge-based Approach of Connect-Four: The Game Is Solved: White Wins*. Vrije Universiteit, Oct. 1988. Web. 5 May 2015. <http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf>.

Kauffman, Matt, and J. Strother Moore. *Industrial Proofs with ACL2*. Tech. Advanced Micro Devices, University of Texas at Austin, n.d. Web. 5 May 2015. <http://www.cs.utexas.edu/users/moore/publications/how-to-prove-thms/intro-to-acl2.ps>.

Garland, Stephen J., and John V. Guttag. A Guide to LP, The Larch Prover. Rep. Hewlett-Packard Co., 31 Dec. 1991. Web. 5 May 2015. <http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-82.pdf>.

Leveson, Nancy. *Medical Devices: The Therac-25*. Rep. University of Washington, n.d. Web. 5 May 2015. <http://sunnyday.mit.edu/papers/therac.pdf>.

"LP, the Larch Prover -- Design Philosophy." LP: Introduction. Massachusetts Institute of Technology, n.d. Web. 05 May 2015. <http://www.sds.lcs.mit.edu/spd/larch/LP/misc/philosophy.html>.

Moler, Cleve. "A Tale of Two Numbers." *A Tale of Two Numbers*. MathWorks, n.d. Web. 05 May 2015. <http://www.mathworks.com/company/newsletters/articles/a-tale-of-two-numbers.html>.

Thomas, Muffy. "A Proof of Incorrectness Using the LP Theorem Prover: The Editing Problem in the Therac." Thesis. University of Glasgow, 1993. *A Proof of Incorrectness Using the LP Theorem Prover: The Editing Problem in the Therac*. University of Glasgow, 11 Aug. 1993. Web. 5 May 2015. <http://www.dcs.gla.ac.uk/~muffy/papers/HIS2.ps>.

Works Consulted

http://www.cs.miami.edu/~tptp/OverviewOfATP.html

http://www.hasbro.com/common/documents/dad2614d1c4311ddbd0b0800200c9a66/1ef6874419b9f36910222eb9858e8cb8.pdf