Human Aspects of SASyLF, an Educational Proof Assistant for Type Theory

JONATHAN ALDRICH, Carnegie Mellon University, USA

An increasing number of programming language courses include content on type theory. But learning about type theory is hard: students must learn what constitutes a rigorous proof and how to construct one. We have designed the SASyLF proof assistant to help students learn to write proofs about programming languages. SASyLF allows students to write proofs in any text editor, using a syntax very similar to what is used in classrooms—ensuring that instructors need not spend a lot of time teaching details of the tool. SASyLF's built-in support for variable binding, inherited from its LF foundations, avoids the need for students to learn arcane variable encodings and allows us to provide good error messages. We provide an overview of SASyLF's design, plans to integrate it into a course and widely used textbook, and the feedback we've gotten from students so far.

CCS Concepts: • **Theory of computation → Interactive proof systems**.

Additional Key Words and Phrases: SASyLF, proof assistant, education, human factors, type theory

ACM Reference Format:
Jonathan Aldrich. 2021. Human Aspects of SASyLF, an Educational Proof Assistant for Type Theory. 1, 1 (October 2021), 3 pages.

1 INTRODUCTION

Teaching and learning type theory is hard. It is difficult for students to state things formally and precisely without making small errors. Unfortunately, those small errors can easily derail student learning: they may prevent the student from completing a proof entirely, or conversely they may make completing the proof too easy, meaning the student never has a chance to learn from their work.

A great deal of effort has gone into developing proof assistants and methodologies that provide automated support for researchers doing proofs in type theory, including the POPLmark challenge from 15 years ago [3]. Fewer have focused on tools for more effectively teaching type theory, which raises an additional set of challenges. To learn effectively, students must focus on higher-level concepts like induction, yet many mistakes are made at a much lower level, e.g. skipping a step in a proof or applying an inference rule when the facts used do not match the rule's premises. Students may not even recognize they have made a mistake, and so do not seek out help. They only learn of their mistake a week or two later, when the TA hands back a paper splashed with red ink. At that point, the student may have forgotten why the mistake was made, and the learning opportunity is lost. A tool that could provide *immediate feedback* would help students get it right in the first place, and also help them to realize when they need to ask an instructor for help with the more challenging concepts.

Author's address: Jonathan Aldrich, jonathan.aldrich@cs.cmu.edu, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, Pennsylvania, USA, 15213.

XXXX-XXXX/2021/10-ART $15.00
https://doi.org/

Proof assistants like Isabelle/HOL [5], Coq [4], and Twelf [6] have been used to formalize language semantics and prove meta-theorems. However, even in the research community, mechanically checked proofs are the exception rather than the rule. This may be partly a productivity issue, but the steep learning curve of these tools, and the non-trivial techniques for encoding program semantics in them, likely play a role. The Ott tool [8] allows users to write down language syntax and semantics in a convenient paper-like notation, but does not support expressing or proving theorems—this must be done in a separate tool, and users must pay the cost of learning it. Unfortunately, due to learning curve issues, the use of these assistants in teaching formal language theory is rare, despite the help they could in theory be to students.

In our talk, we will provide an overview of the SASyLF ("Sassy Elf") proof assistant [1]. SASyLF has a simple design philosophy: language and logic syntax, semantics, and meta-theory should be written as closely as possible to the way it is done on paper. Proofs are very explicit, for the benefit of teaching. Error messages are given in terms of the source proof, not in terms of the assistant's underlying theory. Finally, SASyLF is specialized for reasoning about languages, programs, and logics—more generally, any system with variable binding. Variable binding is a persistent source of complication for proving language meta-theorems in most theorem proving systems because it must be encoded in some way, presenting yet another barrier to using proof assistants in coursework.

The talk will also discuss how SASyLF has been used in graduate type theory courses at two institutions, with positive feedback. In a course offering at UCLA in 2008, students rated the following statements on a Likert scale (1-strongly disagree to 5-strongly agree):

- Would like to use SASyLF in another PL course: 4.2
- Able to learn SASyLF quickly: 3.8
- SASyLF improved my ability to prove theorems, even on paper: 4.0
- SASyLF enabled me to accomplish tasks more quickly: 3.3

Since then, Boyland and his collaborators have made a number of improvements to the tool and related infrastructure, including development of an Eclipse plugin, where clauses that better explain substitutions [2], and a number of extensions that increase the power of the prover. We will talk about some prospective future extensions, including a potential notation and tool for writing derivation trees, and facilities that will make it easier for students to study imperative languages, in addition to the lambda calculus.

We will discuss how SASyLF will be used in a programming language course taught in the Fall of 2021 at Carnegie Mellon University. We are looking at optional integration of the tool with the next edition of Michael Scott's *Programming Language Pragmatics* [7], a popular programming language textbook. We hope to apply qualitative HCI techniques to make further improvements of the tool during the course offering this fall. The talk will end with an invitation to the audience to briefly discuss challenges participants see in this area, both with SASyLF and with other tools.

ACKNOWLEDGMENTS

Many other people have contributed to SASyLF, including John Boyland, Robert Simmons, and Key Shin. Some text in this talk proposal is adapted from prior papers on SASyLF written by the author.

REFERENCES

[1] Jonathan Aldrich, Robert J. Simmons, and Key Shin. 2008. SASyLF: An Educational Proof Assistant for Language Theory. In *Functional and Declarative Programming in Education.*

[2] Michael D. Ariotti and John Tang Boyland. 2017. Making substitutions explicit in SASyLF. In *Logical Frameworks and Meta-Languages: Theory and Practice.*

[3] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. 2005. Mechanized Metatheory for the Masses: The PoplMark Challenge. In *Theorem Proving in Higher Order Logics*. 50–65.
[4] Yves Bertot and Pierre Castéran. 2004. *Interactive Theorem Proving and Program Development.* Springer-Verlag.
[5] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. 2002. Isabelle/HOL: A Proof Assistant For Higher-Order Logic. *Lecture Notes in Computer Science* 2283 (2002).
[6] Frank Pfenning and Carsten Schürmann. 1999. System Description: Twelf — A Meta-Logical Framework for Deductive Systems. In *International Conference on Automated Deduction*. 202–206.
[7] Michael Lee Scott. 2000. *Programming language pragmatics.* Morgan Kaufmann.
[8] Peter Sewell, Francesco Zappa Nardelli, Scott Owens, Gilles Peskine, Tom Ridge, Susmit Sarkar, and Rok Strnisa. 2007. Ott: Effective Tool Support for the Working Semanticist. In *International Conference on Functional Programming*. 1–12.