



Formalization of SASyLF

Sneha Popley

Robert Simmons (Carnegie Mellon University)

Advisor: Dr. Jonathan Aldrich (Carnegie Mellon University)

Honors Committee: Dr. Rhonda Hatcher, Dr. Antonio Sanchez, & Dr. Dick Rinewalt

Department of Computer Science



Motivation

- A flaw in TurboTax may have caused thousands of retired federal employees to overstate their medical deductions and unwittingly underpay the Internal Revenue Service.¹
- Toyota has recalled more than 8 million vehicles around the world since October for problems including braking software glitches.²
- Software plays an integral role in our lives. Its reliability and accuracy is invaluable to us. Operations are complex and depend on complicated data handling.
- Formalizing a programming language interactively with a computer provides a solid basis for software development.

¹ Kocieniowski, David. *Taxpayer could have kept the \$600, but he put his country first.* The New York Times. April 6, 2010.

² The Associated Press. *Toyota promises prompt response on quality issues.* The New York Times. March 30, 2010.

Why Formalize SASyLF?

- A formal definition provides a foundation for
 - Building a community for users
 - Proving properties of the language and programs written in the language.
- It is essential to
 - A complete understanding of the behavior of the language
 - Convincing others about soundness of design
 - Improving reliability of any compiled program in the language.
- SASyLF is a programming language used to teach computer science concepts at the graduate level.
- Formalizing SASyLF shows that a proof in SASyLF is a valid proof.

Goal

- Show relationship between SASyLF and a representation of its underlying logic, M_2^+ .
- Create a translation system between the two languages.



Process

- Write down the abstract syntax of SASyLF, a representation of all programs written with SASyLF.
- Compare it to the abstract syntax of M_2^+ .
- Write down equivalence between the two.
- But, SASyLF is semantically too different from M_2^+ to write down such equations directly (see red highlights in examples).
- So, we introduce a Core Calculus that acts as a transition between SASyLF and M_2^+ .
- Core Calculus is semantically similar to SASyLF and syntactically similar to M_2^+ .

The example below declares natural numbers, the inference rules necessary to prove properties about them, and a theorem that shows that $n + 0 = n$, for any natural number n . These concepts are demonstrated side-by-side as they would look on paper as well as in SASyLF, the Core Calculus, and M_2^+ . The gradual evolution is shown, and the differences between them are highlighted.

Proof on paper

Definitions

Natural Numbers $n ::= z \mid s(n)$

Inference Rules

$$\frac{}{z + n = n} \text{sum-z}$$

$$\frac{n_1 + n_2 = n_3}{s(n_1) + n_2 = s(n_3)} \text{sum-s}$$

Theorem [sum-z-right]: $\forall n \ n + z = n$

Proof. By induction on n :

Case z :
 $z + z = z$ by rule sum-z

Case $s(n)$:
 $n + z = n$ by induction hypothesis
 $s(n) + z = s(n)$ by rule sum-s

q.e.d.



terminals $z \ s$

syntax

$n ::= z \mid s \ n$

judgment sum: $n_1 + n_2 = n_3$

----- sum-z
 $z + n = n$

$n_1 + n_2 = n_3$
 ----- sum-s
 $s \ n_1 + n_2 = s \ n_3$

theorem sum-z-right: forall n exists $n + z = n$.

d1: $n + z = n$ by induction on n :

case z is
d2: $z + z = z$
by rule sum-z
end case

case $s \ n'$ is
d3: $n' + z = n'$
by induction hypothesis on n'
d4: $s \ n' + z = s \ n'$
by rule sum-s on d3
end case
end induction
end theorem

Core Calculus

$\Sigma = n:\text{type},$

$z:n,$
 $s:n \rightarrow n,$

$\text{sum}:n \rightarrow n \rightarrow n \rightarrow \text{type},$

$\text{sum-z} : \Pi n_1:n. \text{sum } z \ n_1 \ n_1,$

$\text{sum-s} : \Pi n_1:n. \Pi n_2:n. \Pi n_3:n. \Pi d:\text{sum } n_1 \ n_2 \ n_3. \text{sum } (s \ n_1) \ n_2 \ (s \ n_3).$

$\square. \forall n_1 : n. \exists d_1:\text{sum } n_1 \ z \ n_1.$

box. $\mu g \in F. \Lambda n_1 : n. \text{case } (n_1) \text{ of}$

$(\triangleright (z/n_1) \rightarrow \langle \text{sum-z } z \rangle)$

$(n'_1 : n \triangleright (s \ n'_1/n_1) \rightarrow$

let $\langle d_1 : \text{sum } n'_1 \ z \ n'_1 \rangle = g \ n'_1$ in
 $\langle \text{sum-s } n'_1 \ z \ n'_1 \ d_1 \rangle$

M_2^+

$\Sigma = n:\text{type},$

$z:n,$
 $s:n \rightarrow n,$

$\text{sum}:n \rightarrow n \rightarrow n \rightarrow \text{type},$

$\text{sum-z} : \Pi n_1:n. \text{sum } z \ n_1 \ n_1,$

$\text{sum-s} : \Pi n_1:n. \Pi n_2:n. \Pi n_3:n. \Pi d:\text{sum } n_1 \ n_2 \ n_3. \text{sum } (s \ n_1) \ n_2 \ (s \ n_3).$

$\square. \forall n_1 : n. \exists d_1:\text{sum } n_1 \ z \ n_1.$

box. $\mu g \in F. \Lambda n_1 : n. \text{case } (n_1/n'_1, g/g') \text{ of}$

$(\triangleright (z/n'_1) \rightarrow \langle \text{sum-z } z, \langle \rangle \rangle)$

$(n''_1 : n \triangleright (s \ n''_1/n'_1) \rightarrow$

let $\langle d_2 : \text{sum } n''_1 \ z \ n''_1, _ \rangle = g \ n''_1$ in
 $\langle \text{sum-s } n''_1 \ z \ n''_1 \ d_2, \langle \rangle \rangle$

Conclusions

- Formal proof that SASyLF is based on a sound logic, M_2^+ .
- Basis for further development of SASyLF
 - More features
 - Added robustness.
- Formal proof of the reliability of SASyLF.
- Suitable for use in teaching graduate-level computer science concepts.
- Possibility of future formalization of SASyLF in other logic systems such as Beluga and Delphin.

Works Cited

Jonathan Aldrich, Robert J. Simmons, and Key Shin. *SASyLF: An Educational Proof Assistant for Language Theory*. In FDPE '08: Proceedings of the 2008 International Workshop on Functional and Declarative Programming in Education, pages 31–40, New York, NY, USA, 2008.

Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, Volume 2283 of LNCS. Springer, 2002.

Frank Pfenning and Carsten Schürmann. *System description: Twelf — a meta-logical framework for deductive systems*. In Proceedings of the 16th International Conference on Automated Deduction (CADE-16, pages 202–206.) Springer-Verlag LNAI, 1999.

Acknowledgments

I would like to thank Dr. Aldrich and Rob for advising me throughout the course of the project. I would also like to thank the LF and Plaid groups at Carnegie Mellon University (CMU) for their contributions throughout the formalization process. This work was initiated by an NSF REU-supported research internship at CMU. I thank Dr. Spice along with my honors committee for helping me throughout the year.