Towards a Lightweight Front-end for Isabelle/Isar

Mikhail Kazdagli
Boston University
Boston, USA
kazdagli@cs.bu.edu

Andrei Lapets
Boston University
Boston, USA
lapets@bu.edu

1 Introduction

This work describes an attempt to assemble a lightweight prototype front-end for verifying propositional logic proofs that relies on the Isabelle/Isar [Isa, NPW02, wcfLB, Wen11, Wen02, WP06, Wen99a] proof authoring and verification system. This prototype serves as an opportunity to become familiar with some of Isabelle/Isar's verification capabilities and limitations, and provides a starting point for future work incorporating Isabelle/Isar as one of the underlying component tools in the Aartifact [And, LK10, Lap10] accessible integrated environment for formal reasoning.

2 Implementation

An input representation for simple formal arguments involving propositional calculus is presented in Figure 1. This representation is a subset of the syntax for formal arguments supported by user-friendly formal verification environments developed in earlier work [LK10, Lap10].

$$
\begin{array}{rccl}
\text{natural number} & n & \in & \mathbb{N} \\
\text{predicate} & p & \in & \mathcal{P} \\
 & & & \\
\text{formula} & f & ::= & p \\
 & & | & \textbf{not } f \\
 & & | & f_1 \textbf{ implies } f_2 \\
 & & | & f_1 \textbf{ and } f_2 \\
 & & | & f_1 \textbf{ or } f_2 \\
 & & & \\
\text{step} & s & ::= & \textbf{assume } f \\
 & & | & \textbf{assert } f \\
 & & & \\
\text{argument} & a & ::= & s_1 \ \ldots \ s_n
\end{array}
$$

Table 1: The input language: a simple propositional logic variant.

The prototype front-end (1) parses arguments represented using the input representation in Figure 1, (2) uses a saturation approach to determine (if possible) what elimination rules must be applied to generate each assertion in the sequence of steps (i.e., formulas) in the argument, and (3) translates this information into a target subset of the syntax supported by Isabelle/Isar (presented in Figure 2). The output conforming to this target syntax is then passed as a raw string into the Isabelle/Isar processor in order to generate Isabelle/Isar verification output.

natural number	n	$\in$	$\mathbb{N}$
predicate	p	$\in$	$\mathcal{P}$
identifier	i	$\in$	$\mathcal{I}$
formula	f	::=	p
		|	**not** f
		|	f_1 `-->` f_2
		|	f_1 `&` f_2
		|	f_1 `|` f_2
rule	r	::=	**by simp** | **by ImpE** | **by ConjE** | **by DisjE**
step	ℓ	::=	**lemma** i "f" **proof have** "f" **sorry qed**
		|	**lemma** i_1 "f" **proof from** i_2 ... i_n **have** i_{n+1}: "f" r **qed**
theory	t	::=	**theory** i **imports Main begin** ℓ_1 ... ℓ_n **end**

Table 2: Target subset of the Isabelle/Isar syntax.

Comments on Isabelle/Isar syntax. The typical structure of a step within a proof represented using the Isabelle/Isar proof syntax [Wen99b, Nipa] is:

$$\textbf{from } i_1 \ \ldots \ i_n \textbf{ have } i_{n+1} : \ \text{“}f\text{”} \textbf{ by } r,$$

where $i_1 \ \ldots \ i_n$ is the list of identifiers corresponding to the formulas used as premises, i_{n+1} is an identifier for the new formula being derived from the premises, and r is the logical inference rule being employed in the derivation. If it is a standard rule that Isabelle/Isar can try to determine automatically, it is possible to use the "double dots" syntax **..** instead of the **by** clause.

Comments on saturation and translation algorithms. The most significant difference between the input syntax and the target syntax is the requirement that within the target syntax every step must have its own identifier, and that each new derived step (i.e., **assert**) must specify the identifiers of the steps used to derive it. To accomplish this, during the saturation step that builds a proof for each assertion, the prototype implementation maintains a context data structure of derived formulas. Each formula added to the context data structure is coupled with its corresponding identifier and a proof specifying the derivation rule (corresponding to **simp**, **ImpE**, **ConjE**, and **DisjE** in the Isabelle/Isar syntax) and the identifiers of existing formulas used to derive it. In this implementation, we considered only elimination rules during the saturation step in order to guarantee convergence. Then, as the prototype implementation traverses the steps of the argument from beginning to end during the translation step, at each assertion step (i.e., **assert**) the algorithm emits the appropriate Isabelle/Isar syntax using the information in the context data structure.

3 Future Work

This work can contribute to planned future efforts to turn Aartifact into a cloud-based web application that allows end-users to employ multiple underlying techniques and tools (including Isabelle/Isar) seamlessly while engaging in formal reasoning tasks. Another possible avenue of future work is the development of a translator from formulas and proofs represented in Isabelle/Isar syntax into a format that makes it possible to populate the static context data structure in Aartifact. This would make it possible to extract and utilize within the integrated environment being developed the existing libraries of facts already assembled for Isabelle/Isar [Nipb].

References

[And] Andrei Lapets. AARTIFACT. `http://www.aartifact.org/`.

[Isa] Isabelle. `http://isabelle.in.tum.de/`.

[Lap10] Andrei Lapets. User-friendly Support for Common Concepts in a Lightweight Verifier. In *Proceedings of VERIFY-2010: The 6th International Verification Workshop*, Edinburgh, UK, July 2010.

[LK10] Andrei Lapets and Assaf Kfoury. A User-friendly Interface for a Lightweight Verification System. In *Proceedings of UITP'10: 9th International Workshop On User Interfaces for Theorem Provers*, Edinburgh, UK, July 2010.

[Nipa] Tobias Nipkow. A Tutorial Introduction to Structured Isar Proofs. `http://isabelle.in.tum.de/website-Isabelle2011-1/dist/Isabelle2011-1/doc/isar-overview.pdf`.

[Nipb] Tobias Nipkow. What's in Main. `http://isabelle.in.tum.de/website-Isabelle2011-1/dist/Isabelle2011-1/doc/main.pdf`.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[wcfLB] Florian Haftmann with contributions from Lukas Bulwahn. Code generation from Isabelle/HOL theories. `http://isabelle.in.tum.de/doc/codegen.pdf`.

[Wen99a] Markus Wenzel. Isar - a generic interpretative approach to readable formal proof documents. In *TPHOLs '99: Proceedings of the 12th International Conference on Theorem Proving in Higher Order Logics*, pages 167–184, London, UK, 1999. Springer-Verlag.

[Wen99b] Markus M. Wenzel. Isabelle/isar - a versatile environment for human-readable formal proof documents. In *TPHOLS*, pages 167–184, 1999.

[Wen02] Markus M. Wenzel. *Isabelle/Isar - A versatile environment for human-readable formal proof documents.* PhD thesis, Institut für Informatik, Technische Universität München, 2002.

[Wen11] M. Wenzel. *The Isabelle/Isar Reference Manual.* January 2011.

[WP06] Markus Wenzel and Lawrence C. Paulson. Isabelle/Isar. In Freek Wiedijk, editor, *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*, pages 41–49. Springer, 2006.