Validating Implicit Induction Proofs Using Certified Proof Environments

Sorin Stratulat, Vincent Demange
LITA, Université Paul Verlaine-Metz, 57000, France
LORIA, 54000, France
stratulat@univ-metz.fr demange@univ-metz.fr

Abstract *We give evidence of the possibility to perform implicit induction-based proofs inside certified reasoning environments, as that provided by the Coq proof assistant. This is the first step of a long term project focused on 1) mechanically certifying implicit induction proofs generated by automated provers like Spike, and 2) narrowing the gap between automated and interactive proof techniques by devising powerful proof strategies inside proof assistants.*

Motivations

Implicit induction proof techniques allow for automated reasoning on inductive properties of equational specifications. Up to now, implicit induction theorem provers, as Spike [1], have been successfully used in treating non-trivial case studies, for instance, the validation of the JavaCard Platform [1] and of an ABR conformance algorithm [6]. Spike proofs are highly automated; in [1], almost half of the JavaCard bytecode instructions have been checked completely automatically in a reasonable time. Unfortunately, their implementation is error-prone and their certification still stands for a challenge.

On the other hand, proof assistants like Coq [9] are mostly interactive. Their strength lies in the implementation of the proof checker, small enough to be readable and verified by humans. Any proof validated by such proof checkers is therefore highly reliable.

Basically, best of both worlds can be reached if automated proofs are integrated into interactive proof assistants. This can be done in two different ways: i) the inference system of the automated prover can be developed and certified into the proof assistant, or ii) a proof in the formalism of the proof assistant can be produced using a trace of the automated prover. In this work, we adopt the second approach.

Previous attempts to validate Spike proofs using Coq have been done by Courant [4] and Kaliszyk [5]. Instead of translating Spike proofs using explicit induction techniques built in Coq as in [4, 5], the implicit induction principle should be validated before being directly applied. Our approach would allow to build a one-to-one translation for any Spike proof.

Example

Spike proof

function symbols	axioms	symbol precedence
$0 : nat$	(1) $0 + y = y$	$0 <_F S <_F +$
$S : nat \rightarrow nat$	(2) $S(x) + y = S(x + y)$	
$+ : nat\ nat \rightarrow nat$		

Spike preserves the minimal counter-examples in the derivations, made of couples (E, H), where E contains conjectures to be refuted, and H formulas which do not contain minimal counter-examples.

Spike proof derivation of $x + 0 = x$:

$$
\begin{array}{ll}
 & (\{x + 0 = x\},\ \emptyset) \\
\vdash_{\text{case\_variable}} & (\{\underline{0 + 0} = 0, \underline{S(x') + 0} = S(x'))\},\ \emptyset) \\
\vdash_{\text{rewrite}} & (\{0 = 0, \overline{S(x' + 0)} = S(x'))\},\ \{x + 0 = x\}) \\
\vdash_{\text{delete}} & (\{S(x' + 0) = S(x'))\},\ \{x + 0 = x\}) \\
\vdash_{\text{injection}} & (\{x' + 0 = x')\},\ \{x + 0 = x\}) \\
\vdash_{\text{subsumption}}_{x \mapsto x'} & (\emptyset,\ \{x + 0 = x\})
\end{array}
$$

At the end, no conjecture contains minimal counter-examples, therefore all conjectures encountered during the proof, including $x + 0 = x$, are true.

Coq translation

nat and the function symbols 0, S, $+$ are already defined in Coq, closely following the previous specification. The Coccinelle library is used to represent abstract terms and formulas (denoted by $\langle F \rangle$), and orders on them, respectively $\prec_{rpo}$ and $\prec\!\!\prec_{rpo}$ (multiset extension of $\prec_{rpo}$). In order to validate Spike proofs, Coccinelle was extended with interesting properties of $\prec\!\!\prec_{rpo}$ (well-foundedness, monotonicity, decidability).

Using the Spike proof, we can construct in Coq a set $\mathcal{F}$ of conjectures together with their abstract representation.

$$\mathcal{F} = \left\{ \begin{array}{l} \lambda x.(x + 0 = x, \langle x + 0 = x \rangle), \\ \lambda x.(S(x) + 0 = S(x), \langle S(x) + 0 = S(x) \rangle), \\ \lambda x.(S(x + 0) = S(x), \langle S(x + 0) = S(x) \rangle) \end{array} \right\}$$

Then, we state and prove the following lemma, where $((x, y))_l = x$ and $((x, y))_r = y$ are the left and right projections, respectively.

Lemma (counter-example non-minimality)**.**

$$\forall F \in \mathcal{F}, \forall x, \neg(Fx)_l \rightarrow \exists F' \in \mathcal{F}, \exists y, \neg(F'y)_l \wedge (F'y)_r \prec\!\!\prec_{rpo} (Fx)_r$$

Proof. (using hints from the Spike proof). By case analysis on elements of $\mathcal{F}$:

- $F \equiv \lambda x.(x + 0 = x, \langle x + 0 = x \rangle)$
  Let $x : nat$ and assume $\neg(Fx)_l \equiv \neg(x + 0 = x)$. By cases on x:
  - $x \mapsto 0$
    $0 + 0 = 0 \overset{\rightsquigarrow}{\equiv}_{(1)} 0 = 0$ which is true, so the implication is trivially verified.
  - $x \mapsto S(x')$
    $S(x') + 0 = S(x') \overset{\rightsquigarrow}{\equiv}_{(2)} S(x' + 0) = S(x')$. For $F' := \lambda x.(S(x + 0) = S(x), \langle S(x + 0) = S(x) \rangle)$ and $y := x'$, we have the result because $(F'y)_r \equiv \langle S(x' + 0) = S(x') \rangle \prec\!\!\prec_{rpo} \langle S(x') + 0 = S(x') \rangle \equiv (Fx)_r$.
- $F \equiv \lambda x.(S(x) + 0 = S(x), \langle S(x) + 0 = S(x) \rangle)\ S(x) + 0 = S(x) \overset{\rightsquigarrow}{\rightsquigarrow}_{rpo} S(x + 0) = S(x)$. We take $F' := \lambda x.(S(x + 0) = S(x), \langle S(x + 0) = S(x) \rangle) \in \mathcal{F}$.
- $F \equiv \lambda x.(S(x + 0) = S(x), \langle S(x + 0) = S(x) \rangle)$
  $S(x+0) = S(x) \equiv x+0 = x$. In this case, $F' := \lambda x.(x+0 = x, \langle x+0 = x \rangle) \in \mathcal{F}$.

□

The lemma permits us to prove that all formulas from $\mathcal{F}$ are true.

Theorem (all true)**.**

$$\forall F \in \mathcal{F}, \forall x, (Fx)_l$$

and, trivially,

$$\forall x, x + 0 = x$$

Proofs are done without performing any explicit induction operation.

Future Works

It seems possible to avoid the use of the excluded middle axiom in order to build proofs constructively. In this way, the user would be able to produce 'implicit induction'-style proofs into Coq; the lemma, the theorem and the computation details involving order comparisons could be hidden. We also plan to develop an automated tactic for Coq, using sets of predefined specifications over usual data structures (nat, $list$, etc.).

References

[1] G. Barthe and S. Stratulat. Validation of the JavaCard platform with implicit induction techniques. In Robert Nieuwenhuis, editor, *R*TA, volume 2706 of *L*ecture Notes in Computer Science, pages 337–351. Springer, 2003.

[2] A. Bouhoula and M. Rusinowitch. Implicit induction in conditional theories. *J*ournal of Automated Reasoning, 14(2):189–235, 1995.

[3] E. Contejean, P. Courtieu, J. Forest, O. Pons, and X. Urbain. Certification of automated termination proofs. *F*rontiers of Combining Systems, pages 148–162, 2007.

[4] Judicaël Courant. Proof reconstruction. Research Report RR96-26, LIP, 1996. Preliminary version.

[5] C. Kaliszyk. Validation des preuves par récurrence implicite avec des outils basés sur le calcul des constructions inductives. Master's thesis, Université Paul Verlaine - Metz, 2005.

[6] M. Rusinowitch, S. Stratulat, and F. Klay. Mechanical verification of an ideal incremental ABR conformance algorithm. *J*. Autom. Reasoning, 30(2):53–177, 2003.

[7] S. Stratulat. Automatic descente infinie induction reasoning. In Bernhard Beckert, editor, *T*ABLEAUX, volume 3702 of *L*ecture Notes in Artificial Intelligence, pages 262–276. Springer, 2005.

[8] S. Stratulat. 'Descente Infinie' induction-based saturation procedures. In *S*YNASC '07: Proceedings of the Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pages 17–24, Washington, DC, USA, 2007. IEEE Computer Society.

[9] The Coq Development Team. The Coq reference manual - version 8.2. http://coq.inria.fr/doc, 2009.