Automation of Boolos' Curious Inference in Isabelle/HOL

Christoph Benzmüller, David Fuenmayor, Alexander Steen and Geoff Sutcliffe

March 17, 2025

Abstract

Boolos' Curious Inference is automated in Isabelle/HOL after interactive speculation of a suitable shorthand notation (one or two definitions).

Contents

1 Introduction

In his article *A Curious Inference* [6], George Boolos discusses an example of the hyper-exponential speedup of proofs when moving from first-order logic to higher-order logic. The first-order proof problem he presents (hereafter called BCP, for Boolos' Curious Problem) has no short proof in first-order logic, but it has an elegant short proof in higher-order logic.

The feasibility of interactive reconstructions of proofs of the BCP at the same level of proof granularity as exercised in Boolos' original paper was shown already one and half decades ago by Benzmüller and Brown [2]. Such an exercise has just recently been repeated in Isabelle/HOL by Ketland [8]. However, as demonstrated in [4], interactive proof for solving BCP is no longer needed, since proof automation in higher-order logic has progressed to the extent that short proofs for BCP can now be found by state-of-the-art higher-order automated theorem provers nearly fully automatically: the only extra human input is to provide one or two suitable shorthand notations.

In this AFP paper, we provide Isabelle/HOL sources related to experiments performed in [4], showing that interactive proof development for BCP (and related problems), as recently practiced by Ketland [8], can now be replaced by almost fully automated proofs. The availability of a powerful hammer tool, such as *Sledgehammer* [5], is of course an essential prerequisite.

In the formalisation presented below, we stick as closely as possible to the syntax of Boolos' original work; this is made easy in the user interface of Isabelle/HOL [11].

2 Boolos Curious Proof Problem

theory *Boolos-Curious-Inference-Automated* **imports** *Main*
begin

First declare a non-empty type i of objects (natural numbers in the context of this paper).

typedecl i

The signature for BCP consists of four uninterpreted constant symbols.

consts
e :: i (‹**e**›) — one
s :: $i \Rightarrow i$ (‹**s**-›) — successor function
f :: $i \Rightarrow i \Rightarrow i$ (‹**f**--›) — binary function; axiomatised below as Ackermann function
d :: $i \Rightarrow$ *bool* (‹**d**-›) — arbitrary uninterpreted unary predicate

Axioms *A1*-*A3* model the Ackermann function and Axioms *A4* and *A5* stipulate the properties of predicate d.

axiomatization where
A1: $\forall\, n.$ **f**n**e** = **se** **and** — Axiom 1 for Ackermann function f
A2: $\forall\, y.$ **fes**y = **ssfe**y **and** — Axiom 2 for Ackermann function f
A3: $\forall\, x\ y.$ **fs**x**s**y = **f**x**f**(**s**x)y **and** — Axiom 3 for Ackermann function f
A4: **de** **and** — d (an arbitrary predicate) holds for one
A5: $\forall\, x.$ **d**$x \longrightarrow$ **ds**x — if d holds for x it also holds for the successor of x

Trying to prove automatically with *Sledgehammer* [5] that d holds for fssssesssse still fails at this point. As Boolos' explains, a naive first-order proof would require more modus ponens steps (with A5 and A4) than there are atoms in the universe.

lemma dfssssesssse — sledgehammer ⟨*proof*⟩

3 Automated Proof: Using Two Definitions

We interactively provide two shorthand notations *ind* and *p*. After their introduction a proof can be found fully automatically with *Sledgehammer*. *ind X* is defined to hold if and only if X is 'inductive' over **e** and *s*. *pxy* holds if and only if *pxy* is in smallest inductive set over **e** and *s*. Note that the symbols *ind* and *p* do not occur in the BCP problem statement.

definition *ind* (‹**ind**-›) **where** $ind \equiv \lambda X.\ X\mathbf{e} \wedge (\forall x.\ X\ x \longrightarrow X\ \mathbf{s}x)$
definition *p* (‹**p**-›) **where** $p \equiv \lambda x\ y.\ (\lambda z{::}i.\ (\forall X.\ \mathbf{ind}\ X \longrightarrow X\ z))\ \mathbf{f}\ x\ y$

Using these definitions, state-of-art higher-order ATPs integrated with Isabelle/HOL can now fully automatically prove Boolos' Curious Problem:

theorem dfssssesssse — *sledgehammer*
⟨*proof*⟩

In experiments (using a MacBook Pro (16-inch, 2019), 2,6 GHz 6-Core Intel Core i7, 16 GB 2667 MHz DDR4) running Isabelle2022 automatically found proofs were reported by various theorem provers, including *Z3* [7], *Vampire* [9], *Zipperposition* [1], *E* [10], and Leo-II (*remote-leo2*) [3].

4 Automated Proof: Using a Single Definition

definition p' (‹$\mathbf{p}'$-›) **where**
$p' \equiv \lambda x\ y.\ (\lambda z{::}i.\ (\forall X.\ (X\mathbf{e} \wedge (\forall x.\ X\ x \longrightarrow X\ \mathbf{s}x)) \longrightarrow X\ z))\ \mathbf{f}\ x\ y$

theorem dfssssesssse — *sledgehammer* (*A1 A2 A3 A4 A5 p′-def*)
⟨*proof*⟩

In experiments (using the same environment as above) several provers reported proofs, including *Z3* and *E*.

5 Proof Reconstruction: E's Proof from [6]

In this section we reconstruct and verify in Isabelle/HOL the proof argument found by E as reported in [6]. Analysing E's proof we can identify the following five lemmata:

lemma *L1a*: **ind** *d* ⟨*proof*⟩
lemma *L1b*: $\forall x.\ \mathbf{p}x\mathbf{e}$ ⟨*proof*⟩
lemma *L1c*: **ind pe** ⟨*proof*⟩
lemma *L2*: $\forall x\ Y.\ (\mathbf{ind\ p}x \wedge \mathbf{ind\ ps}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{fs}x\mathbf{ssse}$ ⟨*proof*⟩
lemma *L3*: $\forall x.\ \mathbf{ind\ p}x \longrightarrow \mathbf{ind\ ps}x$ ⟨*proof*⟩

Using these lemmata E then constructs the following refutation argument:

theorem dfssssesssse

⟨*proof*⟩

This refutation argument can alternatively be replaced by:

theorem dfssssesssse
⟨*proof*⟩

Lemma *L2* can actually be simplified (this was also hinted at by an anonymous reviewer of [4]):

lemma *L2-1*: $\forall x\ Y.\ (\mathbf{ind}\ \mathbf{p}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{f}x\mathbf{e}$ ⟨*proof*⟩
lemma *L2-2*: $\forall x\ Y.\ (\mathbf{ind}\ \mathbf{p}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{f}x\mathbf{se}$ ⟨*proof*⟩
lemma *L2-3*: $\forall x\ Y.\ (\mathbf{ind}\ \mathbf{p}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{f}x\mathbf{sse}$ ⟨*proof*⟩
lemma *L2-4*: $\forall x\ Y.\ (\mathbf{ind}\ \mathbf{p}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{f}x\mathbf{ssse}$ ⟨*proof*⟩
lemma *L2-5*: $\forall x\ Y.\ (\mathbf{ind}\ \mathbf{p}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{f}x\mathbf{sssse}$ ⟨*proof*⟩

etc.

The following statement, however, has a countermodel.

lemma *L2-1*: $\forall x\ y\ Y.\ (\mathbf{ind}\ \mathbf{p}x \wedge \mathbf{ind}\ Y) \longrightarrow Y\mathbf{f}\ x\ y$
nitpick[*user-axioms*,*expect=genuine*] ⟨*proof*⟩

Instead of using *L2* we can now use *L2-5* in our proof of BCP from above:

theorem dfssssesssse
⟨*proof*⟩

6 Conclusion

Isabelle/HOL data sources were provided in relation to [4], which describes recent progress and remaining challenges in automating Boolos' Curious Problem. The interactive introduction of lemmas, as still exercised in [8] and earlier in [2], is no longer necessary, since higher-order theorem provers are now able to speculate the required lemmas automatically, provided that appropriate shorthand notations are provided (see the definitions of *ind* and *p*).

Since Boolos' example for speeding up proofs is interesting in several respects, it is now being used as an exercise in lecture courses at several universities (including University of Bamberg, University of Greifswald, University of Luxembourg, Free University of Berlin); having our source files permanently maintained in AFP hence makes sense.

end

References

[1] A. Bentkamp, J. Blanchette, S. Tourret, P. Vukmirovic, and U. Waldmann. Superposition with lambdas. *Journal of Automated Reasoning*, 65(7):893–940, 2021.

[2] C. Benzmüller and C. Brown. The curious inference of Boolos in MIZAR and OMEGA. In R. Matuszewski and A. Zalewska, editors, *From Insight to Proof – Festschrift in Honour of Andrzej Trybulec*, volume 10(23) of *Studies in Logic, Grammar, and Rhetoric*, pages 299–388. The University of Bialystok, Poland, 2007. http://mizar.org/trybulec65/20.pdf.

[3] C. Benzmüller, N. Sultana, L. C. Paulson, and F. Theiss. The higher-order prover LEO-II. *Journal of Automated Reasoning*, 55(4):389–404, 2015.

[4] C. Benzmüller, D. Fuenmayor, A. Steen, and G. Sutcliffe. Who finds the short proof? *Logic Journal of the IGPL*, 32:442–464, 2024. arXiv preprint: http://doi.org/10.48550/arXiv.2208.06879.

[5] J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *Journal of Formalized Reasoning*, 9(1):101–148, 2016.

[6] G. Boolos. A curious inference. *J. Philos. Log.*, 16(1):1–12, 1987.

[7] L. M. de Moura and N. S. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[8] J. Ketland. Boolos's curious inference in Isabelle/HOL. *Archive of Formal Proofs*, pages 1–19, June 2022. https://isa-afp.org/entries/Boolos_Curious_Inference.html.

[9] E. Kotelnikov, L. Kovács, G. Reger, and A. Voronkov. The vampire and the fool. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2016, pages 37–48. ACM, 2016.

[10] S. Schulz, S. Cruanes, and P. Vukmirovic. Faster, higher, stronger: E 2.3. In P. Fontaine, editor, *CADE 2019*, volume 11716 of *Lecture Notes in Computer Science*, pages 495–507, Cham, 2019. Springer.

[11] M. Wenzel. Isabelle/jedit - A prover IDE within the PIDE framework. In J. Jeuring, J. A. Campbell, J. Carette, G. D. Reis, P. Sojka, M. Wenzel, and V. Sorge, editors, *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings*, volume 7362 of *Lecture Notes in Computer Science*, pages 468–471. Springer, 2012.