HOL Light to Isabelle/HOL Translation, Rebooted

Ghilain Bergeron, Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
Stéphane Glondu, Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
Sophie Tourret, Université de Lorraine, CNRS, Inria, LORIA, Nancy, France and MPI for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

July 11, 2024

Abstract

This paper is a report on the use of a ten-year-old proof translation technique from HOL Light to Isabelle/HOL [6]. We demonstrate the reusability of the technique and provide a detailed account of the steps needed to adapt the code both to a newer version of the HOL Light kernel and to the translation of specific HOL Light theorems. Thus, in addition to the proof of reusability, this paper provides a guideline for anyone wishing to use HOL Light theorems in Isabelle/HOL.

1 Introduction

The variety of languages and tools for ITP is both a strength and a weakness of the domain. On the one hand, it gives users the ability to choose the proof assistant that best suits their needs. On the other hand, the resulting formalizations are only available to a limited amount of users: the ones using the same tool. When wanting to use a result previously formalized in a different proof assistant than your own tool of choice, there are but a few options. The first one is to redo the whole formalization on your own, ignoring or taking loose inspiration from the existing formalization, incurring a lot of time-consuming efforts with few merits. The second one is to rely on an automatic translation tool, but these tend to be brittle research projects that only experts in proof translation really want to use.

If you want to translate HOL Light to Isabelle/HOL as in our case, then if you know the foundations of both languages in and out, a third option is to translate the proof by sight, as these foundations are extremely close to one another.[1] In fact, possibly due to these shared foundations, previous automatic translation attempts from HOL Light to Isabelle/HOL are not so brittle as one might think and it turns out that with a moderate amount of efforts, it is

[1]On this topic, you may find Lawrence Paulson's blog posts [8, 9] of interest.

possible to use HOL Light results in Isabelle/HOL, making the second option outlined previously the most time-efficient one. This is possible thanks to work by Kaliszyk and Krauss [6]. Achieving this is as simple as downloading the archive available at `http://cl-informatik.uibk.ac.at/users/cek/import/` and following the instructions to obtain a file and setting up a link to this file in the Isabelle settings. By including `HOL_Light_Import.thy`, available in the main distribution of Isabelle/HOL, in the imports of your `thy` file, you have access to the whole HOL Light standard library... or rather to the HOL Light standard library as it was at the time the original translation was created, that is around 2013. But what happens if you want a more recent result? The present paper is a report on how we reused Kaliszyk and Krauss's work to import the current HOL Light standard library to Isabelle/HOL, and another HOL Light result that we needed for a different formalization project. It is an account of reusability that stands against the current belief that translation works are doomed to oblivion, as well as a road-map for Isabelle/HOL users wanting to invoke HOL Light theorems in their proofs.

2 Overview of the Translation Process

The proof translation process by Kaliszyk and Krauss proceeds in four steps:

1. collecting theorems to export from HOL Light;
2. exporting the inference trace using a patched HOL Light kernel;
3. trimming the inference trace in an offline garbage collection step;
4. importing the trace to Isabelle/HOL while aligning HOL Light concepts with their Isabelle/HOL counterparts.

We reuse the same 4-step architecture with a few tweaks. The updated software is available online.[2] Our overall translation process is described in Figure 1. In the figure, empty arrowheads indicate isabelle imports while full arrowheads denote standard input/output behavior. Slanted rectangles in blue (darker gray in black and white) denote files that may require changes depending on the proofs being translated.

A HOL Light theory is an OCaml script which defines theorems as OCaml values. In HOL Light, the OCaml toplevel, which is the prompt interface for OCaml, is directly used for interactivity. Step 1 involves OCaml scripting to extract all defined theorems after the run of a theory. Step 2 involves patching the HOL Light kernel to record the inference trace of the theory. Steps 1 and 2 are generic in the sense that nothing specific to the theory needs to be done. Adapting Kaliszyk and Krauss's technique to the current HOL Light version was straightforward. Step 1 simply required to run the current HOL Light on the targeted theory while step 2 required a merge of the old patched HOL Light

[2] `https://gitlab.inria.fr/hol-light-isabelle/import`

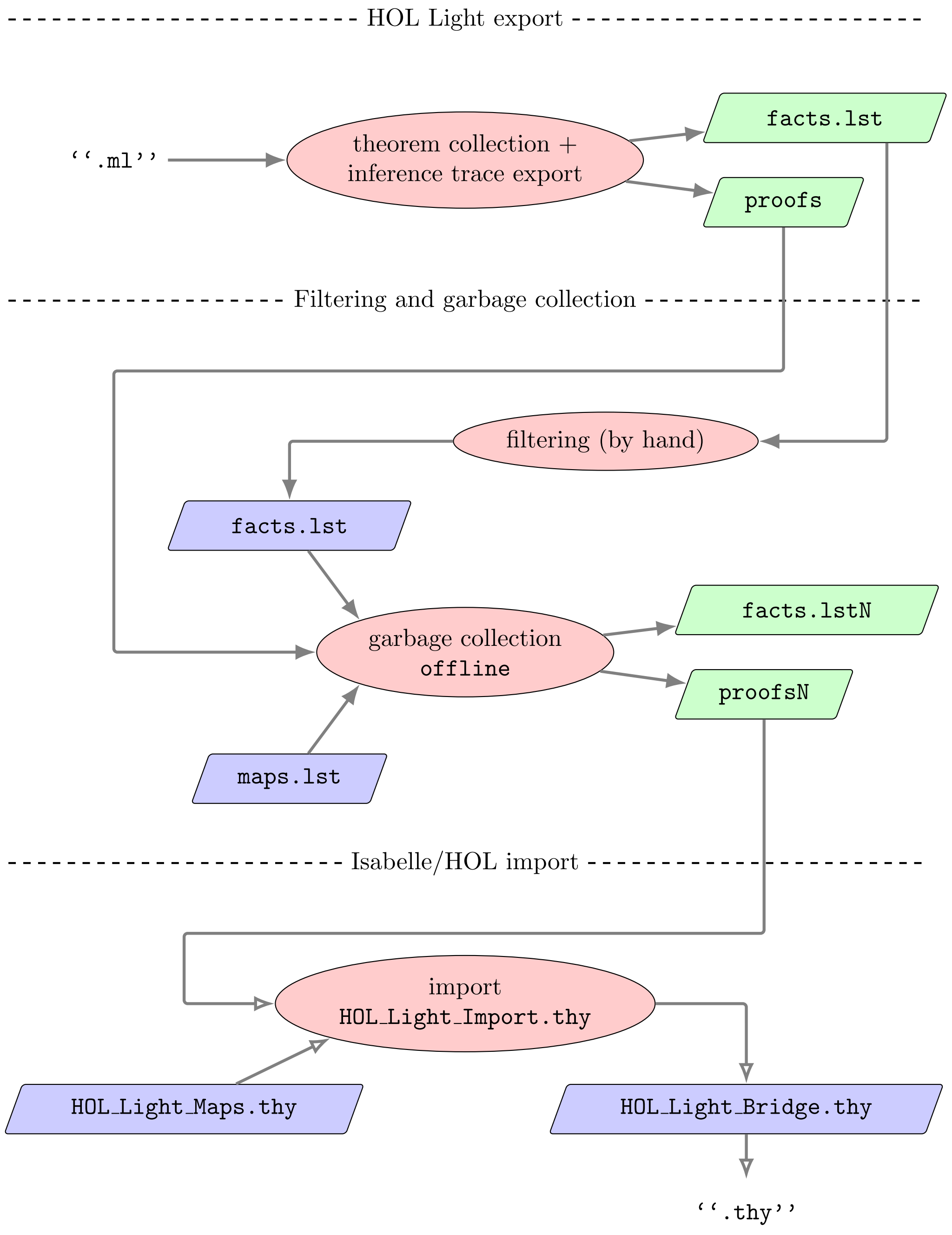

Figure 1: Bird’s-eye view of the overall translation process

kernel with the current one before running the updated patched kernel on the theory. The patch enabling the recording of inference traces was no trouble to merge to the current kernel because its changes are orthogonal to the ones that have occurred since 2013. Step 2 produces two files:

- `proofs`: the inference trace, where each fact is identified by a number;
- `facts.lst`: the list of all facts that can be imported into Isabelle with their number.

Step 2 is rather efficient in terms of speed, but the trace produced can be big: 230 MB for the HOL Light standard library, 641 MB for the whole first-order logic theory[3] that includes the result that motivated this whole work.

Step 3 is performed by a stand-alone program called `offline`. One goal of `offline` is to reduce the trace produced in step 2: given the theorems of interest, listed in `facts.lst`, it will keep them and their transitive dependencies and remove everything else (hence the name *garbage collection*). Running `offline` does not involve HOL Light nor Isabelle; it produces a file `proofsN`, denoted as the *bundle* in the last step. This bundle is what needs to be obtained from Kaliszyk and Krauss's archive for the old translation to work.

An additional filtering step on `facts.lst` may be performed beforehand if one does not want all theorems exported in steps 1 and 2. This filtering simply consists in erasing lines from `facts.lst`. The filtering and garbage collecting process can also be iterated by filtering `facts.lstN` and using it and `proofsN` once more as inputs to `offline`. This is not shown on Figure 1 but justifies the existence of `facts.lstN`.

Another goal of `offline` is to align HOL Light concepts with equivalent Isabelle concepts, for ease of use of the imported theorems and to reduce the size of the bundle. The mapping is given in the `maps.lst` file. For example, HOL Light's logic connectors, numbers and lists can be mapped directly to their Isabelle counterparts.

Step 4 covers the import and replay of the trace in Isabelle/HOL. A skeleton of these is shipped directly with the Isabelle distribution (see the directory `src/HOL/Import`). Thanks to this, the Isabelle part of Kaliszyk and Krauss's work [6] was maintained throughout the times. It involves ML modules that read the bundle, and Isabelle theories to act as glue:

- `HOL_Light_Import.thy` includes the actual call to the importer (written in ML). The importer is called with the import_file command. It reads the bundle and calls Isabelle's low-level functions to build theorems. The importer itself is not part of the trusted computing base of the proof assistant. Thus the bundle can be seen as a certificate that is checked by Isabelle's kernel.
- `HOL_Light_Maps.thy` is the Isabelle counterpart of `maps.lst` used by `offline`. It provides lemmas of equivalence between the aligned HOL Light and Isabelle notions.

[3] https://github.com/jrh13/hol-light/tree/master/Logic

Once `HOL_Light_Import.thy` is added to the includes of an Isabelle theory, all selected HOL Light symbols are available: functions and their defining equations as well as theorems. For example, the equivalent of Isabelle's list_all function is available under the name `HOL_Light_Import`.ALL. The defining equation of this recursive function in Isabelle, also named ALL, states:

$$\mathsf{ALL}\ P\ []=\mathsf{True}\ \wedge\ \mathsf{ALL}\ P\ (h\ \#\ t)=(P\ h\ \wedge\ \mathsf{ALL}\ P\ h)$$

Writing good concept mappings helps reducing the size of the bundle and produces theorems that look more native to Isabelle. However, in practice, it is time-consuming. Indeed each iteration involves stopping Isabelle, tweaking `maps.lst` and `HOL_Light_Maps.thy`, running `offline`, restarting Isabelle, importing the dependent theories and running the importer. Depending on the theory of interest, the last two steps can take several minutes, and this process involves context switches that increase cognitive load and break momentum.

Moreover, some HOL Light notions, although equivalent to Isabelle ones, cannot be directly mapped to Isabelle using the current importer. Therefore, we added a *post-import bridge*, a.k.a. `HOL_Light_Bridge.thy`, that is a collection of equivalence theorems between both worlds, rephrasing HOL Light theorems with Isabelle notions post import. We provide examples of its use in the following section. Proofs in the bridge are usually straightforward and do not involve the aforementioned context switches: everything can be done interactively in the same Isabelle session. Thus, as a user relying preferably on the bridge requires fewer efforts, with the downside that the resulting bundle may end up bigger than otherwise.

3 Two Examples

We first show how to apply the translation on the whole HOL Light standard library. Then we turn our attention to the translation of a single result, namely a theorem stating the compactness of first-order logic, formalized by Harrison in HOL Light more than twenty years ago [3] but available online only since 2018.[4]

3.1 HOL Light Standard Library

Exporting the HOL Light standard library (as of January 2024) produces a 230 MB trace and 3680 facts. The concept mappings done in 2013 [6] could be partially reused. However, in their original forms, some theorems were no longer accepted by the importer. The issues we fixed in the mapping can be separated in two general classes, both related to types.

In the first class, we put issues originating from the automatic typing mechanism of HOL Light. HOL Light terms do not necessarily have an explicit type and in that case HOL Light generates a default type automatically. The default

[4]https://github.com/jrh13/hol-light/blob/master/Logic/canon.ml

types are of the form $'t < number >$ where $< number >$ is sequentially increasing as HOL Light proceeds through the ml files. This is the way HOL Light is implemented and not something we can change, at least if we want to avoid a deep dive in HOL Light's implementation. Types in `HOL_Light_Maps.thy` must match exactly their HOL Light counterparts but some of these numbers have shifted since 2013 due to the addition of new results in HOL Light these last eleven years. Unfortunately, this creates baffling incompatibility issues in Isabelle/HOL. Therefore, one has to align the types, either by inspecting theorems in HOL Light or by carefully inspecting error messages from the importer (with printing of full types on, using the "declare[[show_types = true]]" command before import_file). For example the equivalence between ALL in HOL Light and list_all in Isabelle appears in the file `HOL_Light_Maps.thy` from 2013 as:

lemma ALL[import_const ALL : list_all] :
$\mathsf{list\_all}\,(P :: {'t18393} \Rightarrow bool)\ [] = \mathsf{True} \wedge \mathsf{list\_all}\,P\,(h \# t) = (P\,h \wedge \mathsf{list\_all}\,P\,t)$

In the new version, we must have $P :: {'t25411}$ instead of $P :: {'t18393}$ for the lemma to be accepted by the importer. Future changes in the HOL Light standard library are sure to affect the numbering of default types, thus similar fixes will most likely be needed for each new translation.

In the second class, we put issues related to polymorphism. When a polymorphic HOL Light theorem is instantiated with different types (often default ones as above), the Isabelle importer is not able to unify them. We think this could be fixed by making types explicit in HOL Light theorems, but our goal is to import HOL Light theories *as is*. This class of issues must stem from changes to the Isabelle/HOL code base to which the importer has not been adapted, but we have no strong lead for its exact origin. While it would probably be possible to fix these issues by rewriting the ML code of the importer, we chose instead to simply remove the problematic results from the alignment. This choice was made because there was only a limited amount of affected results, mostly list- and set-related lemmas that do not impact the standard library much but rather the other example that we present in the following section. For the standard library, the resulting bundle is 106 MB. As a measure of what is lost, this must be compared to the 30 MB of the bundle obtained by following the instructions in the archive by Kaliszyk and Krauss. This increase in size is the price to pay for the lost alignments.

3.2 Compactness of First-Order Logic

The main goal of our work was to import a HOL Light version of a proof of compactness of first-order logic named COMPACT_LS. To be more precise, we only wanted to obtain the hard direction of the equivalence, stating that if all finite subsets of a set of formulas are satisfiable, then so is the full set. COMPACT_LS is included among other results relating to model theory in first-order logic [3].

Exporting the whole first-order model theory produces a 642 MB trace and 5823 facts. Unlike in the previous section, we focus here on one specific theorem. Thus we can rely on filtering in addition to mapping in order to lighten the bundle. We strove to align as many concepts as possible. When this was not possible, as in the case of list- or set-related results, both needed in the formalization of compactness, we proved the equivalence of concepts in the bridge. The case of sets is particularly interesting.

The HOL Light theory proving compactness of first-order logic uses the notion of a set formalized as a synonym of "$\alpha \Rightarrow$ *bool*". However Isabelle relies on a type constructor "α *set*". This discrepancy makes it impossible to map HOL Light sets directly to Isabelle sets. To repair it, we defined a notion of a set in terms of "$\alpha \Rightarrow$ *bool*" in Isabelle and mapped the HOL Light notion to it in `maps.lst`. Then we proved the equivalence of the native Isabelle notion to the Isabelle "$\alpha \Rightarrow$ *bool*" notion in the bridge. Despite this discrepancy, the alignment was straightforward to prove and we do not expect that there would be issues for other such alignments with different native representations, thanks to the shared foundations of HOL Light and Isabelle/HOL.

In addition to aligning standard concepts we also wanted to align concepts specific to the theory. The definition of first-order terms was imported from the Archive of Formal Proofs [10]. We could map these in the usual way. Notions appearing in COMPACT_LS, namely the notions of a *language*, an *interpretation* and of an interpretation *satisfying* a formula, were already available to us in the theories `FOL_Syntax.thy` and `FOL_Semantics.thy`, thanks to an earlier attempt at hand-translation. Thus we relied on the bridge. These notions could not be mapped due to depending on sets and lists and using the related functions that have the polymorphic type issue mentioned in subsection 3.1.

The theorem COMPACT_LS is available from `HOL_Light_Import` as:

$$\begin{aligned}
&(\forall t.\ \texttt{HOL\_Light\_Maps\_Set}.\mathsf{finite}\ t\ \wedge\ \texttt{HOL\_Light\_Maps\_Set}.\mathsf{subset}\ t\ s\\
&\quad \rightarrow (\exists M.\ \mathsf{interpretation}\ (\texttt{HOL\_Light\_Import}.\mathsf{language}\ s)\ M\\
&\qquad\qquad \wedge \mathsf{Dom}\ M \neq \texttt{HOL\_Light\_Maps\_Set}.\mathsf{empty}\\
&\qquad\qquad \wedge \texttt{HOL\_Light\_Import}.\mathsf{satisfies}\ M\ t))\\
&\rightarrow (\exists C.\ \mathsf{interpretation}\ (\texttt{HOL\_Light\_Import}.\mathsf{language}\ s)\ C\\
&\qquad \wedge \mathsf{Dom}\ C \neq \texttt{HOL\_Light\_Maps\_Set}.\mathsf{empty}\\
&\qquad \wedge \texttt{HOL\_Light\_Import}.\mathsf{satisfies}\ C\ s)
\end{aligned}$$

and in `HOL_Light_Bridge.thy`, following the additional alignments, it is available in the form we expect, as:

$$\begin{aligned}
&(\forall t.\ \texttt{Finite\_Set}.\mathsf{finite}\ t\ \wedge\ t \subseteq s\\
&\quad \rightarrow (\exists M.\ \mathsf{is\_interpretation}\ (\texttt{FOL\_Syntax}.\mathsf{language}\ s)\ M\\
&\qquad\qquad \wedge \texttt{FOL\_Semantics}.\mathsf{satisfies}\ M\ t))\\
&\rightarrow (\exists C.\ \mathsf{is\_interpretation}\ (\texttt{FOL\_Syntax}.\mathsf{language}\ s)\ C\\
&\qquad \wedge \texttt{FOL\_Semantics}.\mathsf{satisfies}\ C\ s)
\end{aligned}$$

The resulting bundle is only 19 MB, plus 800 lines of Isabelle glue code.

4 Concluding Discussion

Proof translation is the cornerstone of proof assistant interoperability, a feature of ITP that is much desired but extremely challenging to obtain in practice. In addition to various one-to-one translation techniques[5], some going as far as to combine proof assistants [5], several projects ambition to provide a single generic meta-language for all proof translations. One of them is OpenTheory [4], that aims at interoperability between all systems built on HOL.[6] There are also ongoing efforts, like EuroProofNet [1] that coalesces translation efforts centred around lambda-Pi [2]. Another ongoing project is OMDoc/MMT [7] that uses XML to capture not only proofs but also proof documentation.

In this paper, we described how to reuse the one-to-one HOL Light to Isabelle/HOL translation method by Kaliszyk and Krauss. With two examples, we illustrated the pitfalls we encountered and how we faced them. As future work, we will investigate whether the `HOL_Light_Maps.thy` file currently included with Isabelle can be replaced by our new version.

References

[1] EuroProofNet COST Action. https://www.cost.eu/actions/CA20111/. 8

[2] Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-Pi-calculus modulo. In *TLCA*, volume 4583 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2007. 8

[3] John Harrison. Formalizing basic first order model theory. In *TPHOLs*, volume 1479 of *Lecture Notes in Computer Science*, pages 153–170. Springer, 1998. 5, 6

[4] Joe Hurd. The OpenTheory standard theory library. In *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2011. 8

[5] Fabian Immler, Jonas Rädle, and Makarius Wenzel. Virtualization of HOL4 in isabelle. In *ITP*, volume 141 of *LIPIcs*, pages 21:1–21:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 8

[6] Cezary Kaliszyk and Alexander Krauss. Scalable LCF-style proof translation. In *ITP*, volume 7998 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2013. 1, 2, 4, 5, 8

[7] Michael Kohlhase and Florian Rabe. Experiences from exporting major proof assistant libraries. *J. Autom. Reason.*, 65(8):1265–1298, 2021. 8

[5]See https://europroofnet.github.io/tools/ for a non-exhaustive but consequent list.

[6]Kaliszyk and Krauss's approach is compared with OpenTheory in Section 1 of their paper [6].

[8] Lawrence Paulson. Porting libraries of mathematics between proof assistants. `https://lawrencecpaulson.github.io/2022/09/14/Libraries.html`. 1

[9] Lawrence Paulson. Porting the HOL Light metric space library. `https://lawrencecpaulson.github.io/2023/07/12/Metric_spaces.html`. 1

[10] Christian Sternagel and René Thiemann. First-order terms. *Archive of Formal Proofs*, February 2018. `https://isa-afp.org/entries/First_Order_Terms.html`, Formal proof development. 7