Sharing HOL4 and HOL Light proof knowledge

Thibault Gauthier and Cezary Kaliszyk

University of Innsbruck, Austria
{thibault.gauthier,cezary.kaliszyk}@uibk.ac.at

Abstract. New proof assistant developments often involve concepts similar to already formalized ones. When proving their properties, a human can often take inspiration from the existing formalized proofs available in other provers or libraries. In this paper we propose and evaluate a number of methods, which strengthen proof automation by learning from proof libraries of different provers. Certain conjectures can be proved directly from the dependencies induced by similar proofs in the other library. Even if exact correspondences are not found, learning-reasoning systems can make use of the association between proved theorems and their characteristics to predict the relevant premises. Such external help can be further combined with internal advice. We evaluate the proposed knowledge-sharing methods by reproving the HOL Light and HOL4 standard libraries. The learning-reasoning system HOL(y)Hammer, whose single best strategy could automatically find proofs for 30% of the HOL Light problems, can prove 40% with the knowledge from HOL4.

1 Introduction

As *Interactive Theorem Prover* (ITP) libraries were developed for decades, today their size can often be measured in tens of thousands of facts [4, 19]. The theorem provers typically differ in their logical foundations, interfaces, functionality, and the available formalized knowledge. Even if the logic and the interface of the chosen prover are convenient for a user's purpose, its library often lacks some formalizations already present in other provers' libraries. Her only option is then to manually repeat the proofs inside her prover. She will then take ideas from the previous proofs and adapt them to the specifics of her prover. This means that in order to formalize the desired theory, the user needs to combine the knowledge already present in the library of her prover, with the knowledge present in the other formalization.

We propose an approach to automate this time-consuming process: It consists of overlaying the two libraries using concept matching and using learning-assisted automated reasoning methods [15], modified to learn from multiple libraries and able to predict advice based on multiple libraries. In this research we will focus on sharing proof knowledge between libraries of proof assistants based on higher-order logic, in particular HOL4 [23] and HOL Light [9]. Extending the approach to learning from developments in provers that do not share the same logic lies beyond the scope of this paper.

Once a sufficient number of matching concepts is discovered, theorems and proofs about these concepts can be found in both libraries, and we can start to implement methods for using the combined knowledge in future proofs. To this end, we will use the AI-ATP system HOL(y)Hammer [15]. We will propose various scenarios augmenting the learning and prediction phases of HOL(y)Hammer to make use of the combined proof library. In order to evaluate the approach, we will simulate incrementally reproving a prover's library given the knowledge of the library of the other prover. The use of the combined knowledge significantly improves the proof advice quality provided by HOL(y)Hammer. Our description of the approach focuses on HOL Light and HOL4, but the method can be applied to any pair of provers for which a mapping between the logics is known.

1.1 Related work

As reuse of mathematical knowledge formalizations is an important problem, it has already been tackled in a number of ways. In the context of higher-order logic, OpenTheory [12] provides cross-prover packages, which allow theory sharing and simplify development. These packages provide a high-quality standard library, but need to be developed manually. The *Common HOL Platform* [1] provides a way to re-use the proof infrastructure across HOL provers.

Theory morphisms provide a versatile way to prove properties of objects of the same structure. The idea has been tried across Isabelle formalizations in the AWE framework by Bortin et al. [5]. It also serves as a basis for the MMT (Module system for Mathematical Theories) framework [22].

With our method, this principle was developed in both directions. We first search for similar properties of structures to find possible morphism between different fields. We then use these conjectured morphisms to translate the properties between the two fields. Our main idea is that we don't prove the isomorphism which is often a complex problem but we learn from the knowledge gained from the derived properties. Moreover, even when the two fields are not completely isomorphic, the method often gives good advice. Indeed, suppose the set of reals in one library were incorrectly matched to the set of rationals in the other, we can still rely on properties of rationals that are also true for reals.

A direct approach is to create translations between formal libraries. This can only be applied when the defined concepts have the same or equivalent definitions. The HOL/Import translation from HOL4 and HOL Light to Isabelle/HOL implemented by Obua and Skalberg [20] already mapped a number of concepts. This was further extended by the second author [13] to map 70 concepts, including differently defined real numbers. HOL Light has also been translated into Coq by Keller and Werner [18]. It is the first translation between systems based on significantly different logics. In each of these imports, the mapping of the concepts has been done manually.

Compared with manually defined translations, our approach can find the mappings and the knowledge that is shared automatically. It can also be used to prove statements that are slightly different and in some cases even more general. Additionally, the proof can use preexisting theorems in the target library. On

the other hand, when a correct translation is found by hand, it is guaranteed to succeed, while our approach relies on AI-ATP methods which fail for some goals. The possibility of combining the two approaches is left open.

Overview The rest of this paper is organized as follows. In Section 2, we introduce the AI-ATP system HOL(y)Hammer and describe automatic recognition of similar concepts in different formal proof developments. In Section 3, we propose a number of scenarios for combining the knowledge of multiple provers. In Section 4, we evaluate the ability to reprove the HOL4 and HOL Light libraries using the combined knowledge. In Section 5 we conclude and present an outlook on the future work.

2 Preliminaries

2.1 HOL(y)Hammer

HOL(y)Hammer [16] is an AI-ATP proof advice system for HOL Light and HOL4. Given a user conjecture, it uses machine learning to select a subset of the accessible facts in the library, that are likely to prove the conjecture. It then translates the conjecture together with the selected facts to the input language of one of the available ATP systems to find the exact dependencies necessary to prove the theorem in higher-order logic. This method is also followed by the system Sledgehammer [21].

In this section we shortly describe how HOL(y)Hammer processes conjectures, as we will augment some of these steps in Section 3. First, we describe how libraries are exported. Then, we explain how the exported objects and dependencies are processed to find suitable lemmas. Finally, we briefly show how the conjecture can be proven from these lemmas. More detailed descriptions of these steps are presented in [15, 7].

Export We will associate each ITP library with the set of constants and theorems that it contains. In particular, the type constructors will also be regarded as constants in this paper. As a first step, we define a format for representing formulas in type theory, as we aim to support formulas from various provers. A subset of this format is chosen to represent the higher-order logic statements in HOL Light and HOL4. Each object is exported in this format with additional information about the theory where it was created. The theory information will let us export incompatible developments (i.e. ones that can not be loaded into the same ITP session or even originate from different ITPs) into HOL(y)Hammer [14]. Additionally, we can fully preserve the names of the original constants in the export. Finally, the dependencies of each theorem (i.e the set of theorems which were directly used to proved it) are extracted. This last step is achieved by patching the kernels of HOL4 and HOL Light.

Premise selection The premise selection algorithm takes as input an (often large) set of accessible theorems, a conjecture, and the information about previous successful proofs. It returns a subset of the theorems that is likely to prove the conjecture. It involves three phases: feature extraction, learning, and prediction.

The features of a formula are a set of characteristics of the theorem, which we represent by strings. Depending on the choice of characterization, it can simply be the list of the constants and types present in the formula, or the string representation of the normalized sub-terms of the formula, or even features based on formula semantics [17]. The *feature extraction* algorithm takes a formula as input and computes this set.

A relation between the features of conjectures and their dependencies is inferred from the features of all proved theorems and their dependencies by the *learning* algorithm. This step effectively finds a function that given conjecture characteristics finds the premises that are likely to be useful to prove this conjecture. *Prediction* refers to the evaluation of this function on a given conjecture.

These phases will be influenced by the concept matching (see Section 2.2) and differentiated in each of the scenarios (see Section 3).

Translation and reconstruction A fixed number of most relevant predicted lemmas (all the experiments in this paper fix this number to 128, as it has given best results for HOL in combination with E-prover [7]) are translated together with the conjecture to an ATP problem. If an ATP prover is able to find a proof, various reconstruction methods are attempted. The most basic reconstruction method is to inspect the ATP proof for the premises that were necessary to prove the conjecture. This set is usually sufficiently small, so that certified ITP proof methods (such as MESON [8] or Metis [11]) can prove the higher-order counterpart of the statement and obtain an ITP theorem.

2.2 Concept Matching

Concept matching [6] allows the automatic discovery of concepts from one proof library or proof assistant in another. An AI-ATP method can benefit from the library combination only when some of the concepts in the two libraries are related: Without such mappings the sets of features of the theorems in each library are disjoint and premise selection can only return lemmas from the library the conjecture was stated in. As more similar concepts are matched (for example we conjecture that the type of integers in HOL4 `h4/int` and the type of integers in HOL Light `hl/int` describe the same type), the feature extraction mechanism will characterize theorems talking about the matched concepts by the same features. As a consequence, we will also get predicted lemmas from the other library. We will discuss how such theorems from a different library can be used without sacrificing soundness in Section 3.

For a step by step of the concept matching algorithm, we will refer to our previous work [6] and only present here a short summary and the changes that

improve the matching for the scenarios proposed in this paper. Our algorithm is implemented for HOL4 and HOL Light, but we believe the procedure can work for any pair of provers based on similar logics such as Coq [10] and Matita [2].

Summary Our matching algorithm is based on the properties (such as associativity, commutativity, nilpotence, . . .) of the objects of our logic (constants and types). If two objects from two libraries share a large enough number of relevant properties, they will eventually be matched, even though they may have been defined or represented differently. In the description of the procedure, we will consider every type as a constant. Initially, the set of matched constants contains only logical constants. First, we give a highest weight for rare properties with a lot of already matched constants. Second, we look at all possible pairs of constants and find their shared properties. The final score for a pair of constant is the sum of their weights amortised by the total number of properties of each constant. The two constants with the highest similarity score are matched. The previous two steps are repeated until there are no more shared properties between unmatched constants.

Improvements and Limitations The similarity scoring heuristic can be evaluated more efficiently than the ones presented in [6] and is able to map more constants correctly: Thanks to a better representation of the data the time taken to run our implementation of the matching algorithm on the standard library of HOL Light (including complex and multivariate) and the standard library of HOL4 was decreased from 1 hour to 5 minutes. By computing only the initial property frequencies and using them together with the proportion of matched constants to influence the weight of each property in the iterative part the time can be further decreased to 2 minutes. The algorithm now returns 220 correct matches instead of the 178 previously obtained and 15 false positives (pairs that are matched but do not represent the same concept) instead of 32. The better results are a consequence of the inclusion of types in the properties and the updated scoring function.

The proposed approach can only match objects that have the same structure. In the case of the two proof assistants we focus on, it can successfully match the types of natural numbers, integers or real number, however it is not able to match the dedicated HOL Light type `hl/complex` to the complex numbers of HOL4 represented by pairs of real numbers `h4/pair(h4/real,h4/real)`. This issue could be partially solved by the introduction of a matching between subterms combined with a directed matching. The type `hl/complex` could then be considered as pair of reals in HOL4. For the reverse direction, we would need to know if the pair of reals was intended to represent a pair of reals or a complex. One idea to solve this problem could be to create a matching substitution that also depends on the theorems. These general ideas could form a basis for a future extension of the matching algorithm.

3 Scenarios

In this section we propose four ways an AI-ATP system can benefit from the knowledge contained in a library of a different prover. We will call these methods "scenarios" and we will call the library of a different prover "external". All four scenarios require the base libraries to already be matched. This means, that we have already computed a matching substitution from the theorems of both libraries and in all the already available facts in the libraries, the matched constants are replaced by their common representatives.

Throughout our scenarios, we will rely on the notion of equivalent theorems to map lemmas from one library to the other. This notion is defined below, as well as some useful notations.

Definition 1 (Equivalent theorems). *Two theorems are considered equivalent if their conjunctive normal forms are equal modulo the order of conjuncts, disjuncts, and symmetry of equality. Given a theorem t, the set of the theorems equivalent to it in the library lib will be noted $E(lib, t)$.*

Remark 1. This definition only makes sense if the two libraries can be represented in the same logic. This is straightforward if the two share the same logic.

Definition 2 (Notations).
Given a library, we define the following notations:

- *$Dep(t)$ stands for the set of lemmas from which a theorem t was proved. We call them the dependencies of t. This definition is not recursive, i.e. the set does not include theorems used to prove these lemmas.*
- *The function $Learn()$ infers a relation between conjectures and sets of relevant lemmas from the relation between theorems and their dependencies.*
- *$Pred(c, L)$ is the set of lemmas related to a conjecture c predicted by the relation L.*

In each scenario, each library plays an asymmetric role. In the following, the library where we want to prove the conjecture, is called the internal or the initial library. In contrast, the library from which we get extra advice from, is called the external library. In this context, using HOL(y)Hammer alone without any knowledge sharing is our default scenario, naturally named "internal predictions". We illustrate each selection method by giving an example of a theorem that could only be reproved by its strategy. These examples are extracted from our experiments described in Section 4.

Scenario 1: External Dependencies The first scenario assumes that the proof libraries are almost identical. We compute the set of theorems equivalent to the conjecture in the external library. For all of their dependencies, we return the lemmas in the library equivalent to these dependencies. The scenario is presented in Fig 1. This scenario would work very well, if the corresponding theorem is present in the external library and a sufficient corresponding subset of its dependencies is already present in the initial library. As this is often not the case (see Section 4), we will use an AI-ATP method next.
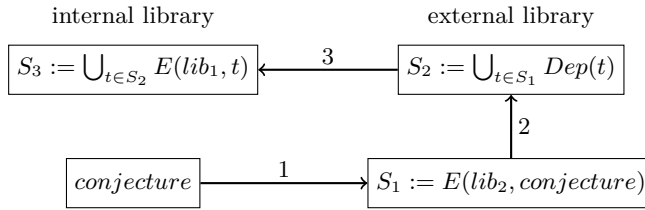
Fig. 1: Finding lemmas from dependencies in the external library.

Example 1. The theorem `REAL_SUP_UBOUND` in HOL4 asserts that each element of a bounded subset of reals is less than its supremum. The equivalent theorem in HOL Light has 3 dependencies: the relation between $<$ and $\leq$ `REAL_NOT_LT`, the antisymmetry of $<$ `REAL_LT_REFL` and the definition of supremum `REAL_SUP`. Each of them have one equivalent in HOL4. The resulting problem was translated and solved by an ATP and the 3 lemmas appeared in the proof.
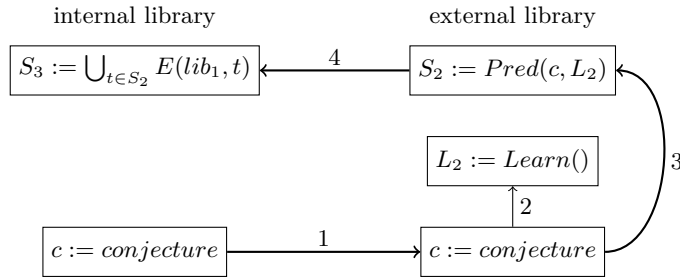

Fig. 2: Learning and predicting lemmas in the external library

Scenario 2: External Predictions The next scenario is depicted in Fig 2. The steps are as follows: We translate the conjecture to the external library (step 1). We predict the relevant lemmas in the external library (steps 2 and 3). We map the predicted lemmas back to the initial library using their equivalents (step 4). To sum up, this scenario proposes an automatic way of proving a conjecture providing that the external library contains relevant lemmas that have equivalents in the internal library. One advantage of this scenario over the standard "internal predictions" is that the relation between features and dependencies is fully developed in the external library, yielding better predictions.

In our experiments, the translation step is not needed because the matching is already applied and the logic of our provers are the same.

Example 2. The theorem `LENGTH_FRONT` from the HOL4 theory `rich_list` states that the length of a non-empty list without its last element is equal to its length

minus one. The subset of predicted lemmas used by the ATP were 6 theorems about natural numbers and 6 theorems about list. These theorems are HOL4 equivalents of selected HOL Light lemmas.
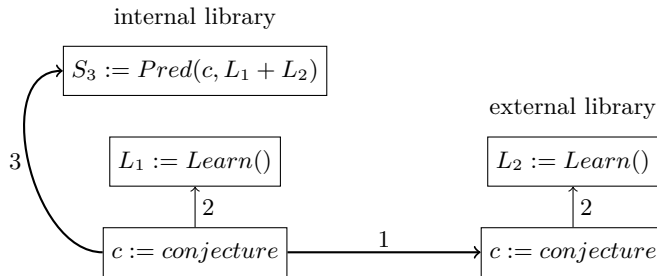
internal library

$$S_3 := Pred(c, L_1 + L_2)$$

external library

$$L_1 := Learn()$$

$$L_2 := Learn()$$

3

2

2

$$c := conjecture$$

1

$$c := conjecture$$

Fig. 3: Learning in both libraries and predicting lemmas in the internal library.

Scenario 3: Combined Learning In this and the next scenario we will combine the knowledge from the external library with the information already present in the internal library. The scenario is presented in Fig 3. First, the conjecture is translated to the external prover. Second, the features suitable for proving the conjecture are learned from the dependencies between the theorems in both systems. Third, lemmas from the original library containing these features are predicted. In a nutshell, this scenario defines an automatic method, that enhances the standard "internal predictions" by including advice from the external library about the relevance of each feature.

Example 3. This example and the next one are using advice from HOL4 in HOL Light which means that the roles of the two provers are reversed compared to the first two examples. The HOL Light theorem SQRT_DIV asserts that the square root of the quotient of two non-negative reals is equal to the quotient of their square roots. In this scenario no external theorems are translated but learning form the HOL4 proofs still improved the predictions directly made in HOL Light. The proof found for this theorem is based on the dual theorems for multiplication SQRT_MUL and inversion SQRT_INV and basic properties of division real_div, multiplication REAL_MUL_SYM, inversion REAL_LE_INV_EQ and absolute value REAL_ABS_REFL.

Scenario 4: Combined Predictions The last and most developed scenario, shown in Fig 4, associate the strategies from the two preceding scenarios, effectively learning and predicting lemmas from both libraries. The first and second steps are the same as in "combined learning". The third step predicts lemmas in both libraries from the whole learned data. Finally, we map back the external predictions and return them together with the internal predictions.
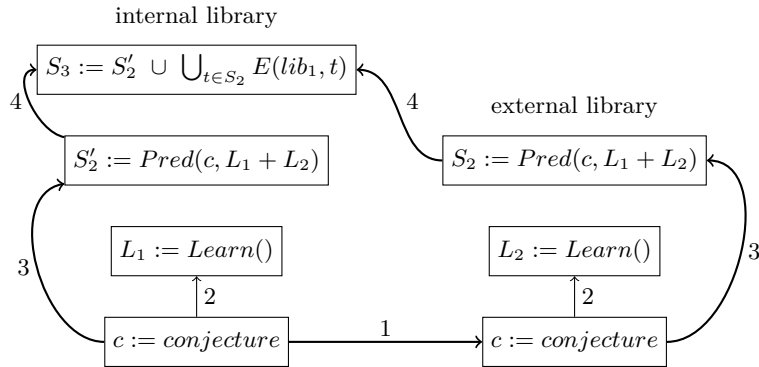
internal library

$$S_3 := S_2' \;\cup\; \bigcup_{t \in S_2} E(lib_1, t)$$

4   4   external library

$$S_2' := Pred(c, L_1 + L_2)$$ $$S_2 := Pred(c, L_1 + L_2)$$

$$L_1 := Learn()$$ $$L_2 := Learn()$$ 3

3   2   2

$$c := conjecture$$ 1 $$c := conjecture$$

Fig. 4: Learning and predicting lemmas from both libraries.

Example 4. Let n, m, p be natural numbers.
The **HOL Light** theorem **HAS_SIZE_DIFF** declares that if a set A has n elements and B is a subset of A that has m elements then the difference $B \setminus A$ has $n - m$ elements. The first two lemmas necessary for the proof were directly found in **HOL Light**. One is the definition of the constant **HAS_SIZE** which asserts that a set has size p if and only if it is finite and has cardinality p. The other **CARD_DIFF** is almost the same as the theorem to be proved but stated for the cardinality of finite sets. The missing piece **FINITE_DIFF** is predicted inside the **HOL4** library. Its equivalent in **HOL Light** declares that the difference of two finite sets is a finite set, which allows the ATP to conclude.

3.1 Unchecked scenarios

In each of the previous scenarios, the final predicted lemmas come from the initial library. This means that our approach is sound with respect to the internal prover. The application of the matching substitution on one library renames the constants in all theorems injectively because no non-trivial matching is performed between two constants of the same library.

We will now consider the possibility of returning matched lemmas from the external library even if they do not have an equivalent in the internal one. This means giving advice to the user in the form: "your conjecture can be proved using the theorems th_1 and th_2 that you already have and an additional hypothesis with the given statement which you should be able to prove." To verify that these scenarios are well-founded, a user would need to prove the proposed hypotheses. That could be achieved by either importing the theorems or applying the approach recursively. If a constant contained in these lemmas is matched inconsistently then each method would fail to reprove the lemmas, preserving the coherence of the internal library. We do not yet have an import mechanism from **HOL4** to **HOL Light** (and conversely) or a recursive mechanism for our scenarios. In this recursive approaches, the predicted facts in the external library should be

restricted to those proved before the conjecture when it has an equivalent in the external library. Otherwise, a loop in the recursive algorithm may be created.

We will still evaluate the "unchecked" scenarios to see what is the maximum added value such mechanisms could generate.

4 Evaluation

We perform all the experiments on a subset of the standard libraries of HOL Light and HOL4. The HOL4 dataset includes 15 type constructors, 509 constants, and 3935 theorems. The HOL Light dataset contains 21 type constructors, 359 constants and 4213 theorems. The subsets were chosen to include a variety of fields ranging from list to real analysis. The most similar pairs of theories are listed by their number of common equivalent classes of theorems in Table 1. The number of theorems in each theory is indicated in parenthesis.

HOL4 theory	HOL Light theory	common theorems
pred_set(434)	sets(490)	128
real(469)	real(291)	81
poly(87)	poly(142)	72
bool(177)	theorems(90)	61
transc(229)	transc(355)	58
arithmetic(385)	arith(245)	57
integral(83)	transc(355)	48

Table 1: The seven most similar pairs of theories by their number of common equivalent classes of theorems according to our matching

The matching, predictions, and the preparation of the ATP problems have been done on a laptop with 4 Intel Core i5-3230M 2.60GHz processors and 3.6 GB RAM. All ATP problems are evaluated on a server with 48 AMD Opteron 6174 2.2 GHz CPUs, 320 GB RAM and 0.5 MB L2 cache per CPU. A single core is assigned to each ATP problem. The ATP used is E-prover version 1.8 running in the automatic mode with a time limit of 30 seconds.

Simulation We will try to prove each theorem in an environment, where information is restricted to the one that was available when this theorem was proved. This amounts to:

- forgetting that it is a theorem and the knowledge of its dependencies,
- finding the subset of facts in the library that are accessible from this theorem,
- computing the matching with the other library based on this subset only,
- predicting lemmas from this subset (plus the other library in the "unchecked" scenarios).

For the purpose of our simulation, the external library is always completely known, as we suppose that it was created previously. In reality, the two libraries were developed in parallel, with many HOL4 theories available before similar formalizations in HOL Light have been performed.

In Fig. 5, we show the evolution of the number of matched constants and compare it to the number of declared constants in the theory during the incremental reproving of two theories. The first graph shows that the number of matched constants stagnate whereas the declared constants continue to increase in the second half of the theory. This suggests that theories formalizing the same concepts may be developed in different directions for each prover. The second graph indicates a better coverage of the HOL Light theory `lists`. In the beginning, the number of matched constants grows even more rapidly than the number of declared constants because new matches are found for constants defined in previous theories.
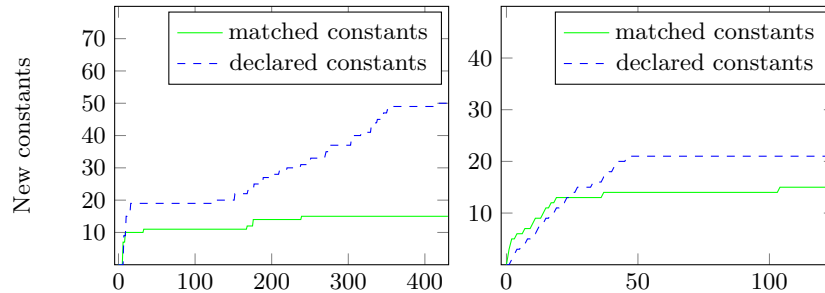

Fig. 5: Evolution of the number of matched constants in the HOL4 theory `list` and in the HOL Light theory `lists`

Scenario	checked(%)	unchecked(%)
empty	4.19	
external dependencies	5.06 (23.50)	10.75 (49.94)
external predictions	17.49	34.42
external any	18.07	34.74
internal predictions	**43.57**	
combined learning	44.03	
combined predictions	44.59	53.46
any	**50.06**	55.73
any checked or unchecked	62.80	

Table 2: Percentage of reproved theorems in the HOL4 library (internal) with the knowledge from the HOL Light library (external).

Scenario	checked(%)	unchecked(%)
empty	3.14	
external dependencies	6.08 (29.22)	10.11 (48.63)
external predictions	12.74	33.94
external any	13.55	34.32
internal predictions	**30.92**	
combined learning	35.13	
combined predictions	35.56	44.06
any	**40.19**	47.07
any checked or unchecked		54.71

Table 3: Percentage of reproved theorems in the HOL Light library (internal) with the knowledge from the HOL4 library (external).

In the first column, scenarios are listed based on their predicted lemmas.
empty: no lemmas
external dependencies: dependencies of equivalent external theorems
external predictions: external lemmas from external advice
external any: problems solved by any of the two previous scenarios
internal predictions: internal lemmas from internal advice
combined learning: internal lemmas from external and internal advice
combined predictions: external and internal lemmas from external and internal advice
any: problems solved by at least one scenario of the same column
In the second column, we restrict ourself from using external theorems that do not have an internal equivalent, where as we allow it in the third column. The last line combines all the problems solved by at least one checked or unchecked scenario.

Results The success rates for each scenario and each proof assistant are compiled in Tables 2 and 3. The scenario "empty" gives the number of facts provable without lemmas and is fully subsumed by the other methods.

The external dependencies scenario is the only one that is not directly comparable to the others, as it was performed only on the theorems that have an equivalent in the other library (876 in HOL Light and 847 in HOL4). The percentage of theorems proved by this strategy relative to its experimental subset is shown in parentheses. This strategy is quite efficient on its subset but contributes weakly to the overall improvement. These results are combined with the "external predictions" scenario to evaluate what can be reproved with external help only. In HOL4, the combined learning and predictions increases the number of problems solved over the initial "internal predictions" approach only by one percent. The improvement is sharper in HOL Light. It suggests that HOL4 provides a better set for the learning algorithm. The improvement provided by all scenarios can be combined to yield a significant gain compared to the performance of HOL(y)Hammer alone, namely additional 6.5% of all HOL4 and 9.3%

of all HOL Light theorems. Another 10–15% could be added by the "unchecked" scenarios.

Results by theory In Table 4, we investigate the performance of the "external dependencies" scenario on the largest theories in our dataset. Some theories only minimally benefit from the external help. This is the case for `rich_list` and `iterate`, where only few correct mappings could be found. We can see asymmetric results in pairs of similar theories. For example, the `real` theory in HOL Light can be 72.16% reproved from HOL4 theories whereas the similar theory in HOL4 does not benefit as much. This suggest that the `real` theory HOL4 is more dense than its counterpart. A similar effect is observed for the `transc` formalization. The theories `pred_set` and `sets` seem to be comparably dense.

Scenario	real	pred_set	list	arithmetic	rich_list	transc
external dependencies	30.91	24.65	10.23	18.18	1.52	5.24

Scenario	sets	analysis	transc	int	iterate	real
external dependencies	25.51	27.1	25.91	52.61	5.47	72.16

Table 4: Reproving success rate in the six largest theories in HOL4 using HOL Light and the "checked external dependencies" scenario, as well as in the six largest HOL Light theories using HOL4.

5   Conclusion

We proposed several methods for combining the knowledge of two ITP systems in order to prove more theorems automatically. The methods adapt the premise selection and proof advice components of the HOL(y)Hammer system to include the knowledge of an external prover. In order to do it, the concepts defined in both libraries are related through an improved matching algorithm. As the constants in two libraries become related, so are the statements of the theorems. Machine learning algorithms can combine the information about the dependencies in each library to predict useful dependencies more accurately.

We evaluated the influence of an external library on the quality of advice, by reproving all the theorems in a large subset of the HOL4 and HOL Light standard libraries. External knowledge can improve the success from 43% to 50% in HOL4 and from 30% to 40% in the number of HOL Light solved goals. This number could reach 54% for HOL4 and 62% for HOL Light if we include the "unchecked" scenarios, where the user is not only suggested known theorems, but also hypotheses left to prove. Proving such proposed lemmas, either with

the help of a translation or by calling an AI-ATP method with shared knowledge is left as future work.

The proposed approach evaluated the influence of an external proof assistant library for the quality of learning and prediction. An extension of the approach could be used inside a single library: mappings of concepts inside a single library, such as those the work of Autexier and Hutter [3], could provide additional knowledge for a learning-reasoning system.

Acknowledgments

References

1. Mark Adams. The common HOL platform. In Cezary Kaliszyk and Andrei Paskevich, editors, *Proceedings Fourth Workshop on Proof eXchange for Theorem Proving (PxTP 2015)*, volume 186 of *EPTCS*, pages 42–56, 2015.
2. Andrea Asperti, Wilmer Ricciotti, and Claudio Sacerdoti Coen. Matita tutorial. *J. Formalized Reasoning*, 7(2):91–199, 2014.
3. Serge Autexier and Dieter Hutter. Structure formation in large theories. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume 9150 of *LNCS*, pages 155–170. Springer, 2015.
4. Jasmin C. Blanchette, Maximilian Haslbeck, Daniel Matichuk, and Tobias Nipkow. Mining the archive of formal proofs. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume 9150 of *LNCS*, pages 3–17. Springer, 2015.
5. Maksym Bortin, Einar Broch Johnsen, and Christoph Lüth. Structured formal development in Isabelle. *Nordic Journal of Computing*, 13:1–20, 2006.
6. Thibault Gauthier and Cezary Kaliszyk. Matching concepts across HOL libraries. In Stephen Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors, *Proc. of the 7th Conference on Intelligent Computer Mathematics (CICM'14)*, volume 8543 of *LNCS*, pages 267–281. Springer Verlag, 2014.
7. Thibault Gauthier and Cezary Kaliszyk. Premise selection and external provers for HOL4. In Xavier Leroy and Alwen Tiu, editors, *Proceedings of the 4th ACM-SIGPLAN Conference on Certified Programs and Proofs*, pages 49–57, 2015.
8. John Harrison. Optimizing proof search in model elimination. In M. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction*, number 1104 in LNAI, pages 313–327. Springer-Verlag, 1996.
9. John Harrison. HOL Light: An overview. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs*, volume 5674 of *LNCS*, pages 60–66. Springer, 2009.
10. Gérard Huet and Hugo Herbelin. 30 years of research and development around Coq. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 249–250. ACM, 2014.

11. Joe Hurd. First-order proof tactics in higher-order logic theorem provers. In Myla Archer, Ben Di Vito, and César Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics (STRATA 2003)*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, September 2003.

12. Joe Hurd. The OpenTheory standard theory library. In Mihaela Gheorghiu Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, volume 6617 of *LNCS*, pages 177–191. Springer, 2011.

13. Cezary Kaliszyk and Alexander Krauss. Scalable LCF-style proof translation. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Proc. of the 4th International Conference on Interactive Theorem Proving (ITP'13)*, volume 7998 of *LNCS*, pages 51–66. Springer, 2013.

14. Cezary Kaliszyk and Florian Rabe. Towards knowledge management for HOL Light. In Stephen Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors, *Proc. of the 7th Conference on Intelligent Computer Mathematics (CICM'14)*, volume 8543 of *LNCS*, pages 357–372. Springer Verlag, 2014.

15. Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.

16. Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015.

17. Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Efficient semantic features for automated reasoning over large theories. In Qiang Yang and Michael Wooldridge, editors, *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 3084–3090. AAAI Press, 2015.

18. Chantal Keller and Benjamin Werner. Importing HOL Light into Coq. In Matt Kaufmann and Lawrence C. Paulson, editors, *ITP*, volume 6172 of *LNCS*, pages 307–322. Springer, 2010.

19. The Mizar Mathematical Library. `http://mizar.org/`.

20. Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *LNCS*, pages 298–302. Springer, 2006.

21. Lawrence C. Paulson and Jasmin C. Blanchette. Three years of experience with Sledgehammer, a practical link between automated and interactive theorem provers. In *8th IWIL*, 2010. Invited talk.

22. Florian Rabe. The MMT API: A generic MKM system. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Proc. of the 6th Conference on Intelligent Computer Mathematics (CICM'13)*, volume 7961 of *LNCS*, pages 339–343. Springer, 2013.

23. Konrad Slind and Michael Norrish. A brief overview of HOL4. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *TPHOLs*, volume 5170 of *LNCS*, pages 28–32. Springer, 2008.