Lemmatization for Stronger Reasoning in Large Theories

Cezary Kaliszyk[1], Josef Urban[2], and Jiří Vyskočil[3]

[1] University of Innsbruck, Austria
[2] Radboud University Nijmegen
[3] Czech Technical University in Prague

Abstract. In this work we improve ATP performance in large theories by the reuse of lemmas derived in previous related problems. Given a large set of related problems to solve, we run automated theorem provers on them, extract a large number of lemmas from the proofs found and post-process the lemmas to make them usable in the remaining problems. Then we filter the lemmas by several tools and extract their proof dependencies, and use machine learning on such proof dependencies to add the most promising generated lemmas to the remaining problems. On such enriched problems we run the automated provers again, solving more problems. We describe this method and the techniques we used, and measure the improvement obtained. On the MPTP2078 large-theory benchmark the method yields 6.6% and 6.2% more problems proved in two different evaluation modes.

1 Introduction

When solving many problems in a certain theory, mathematicians usually remember and re-use the important lemmas found in related problems. *Lemma* usually denotes a statement that was useful or crucial for proving a (important) theorem, often an important technical step. For example commutativity or distributivity of some algebraic operations under certain assumptions might greatly simplify proofs in algebra.

This paper describes several experiments that attempt to improve the efficiency of automated theorem proving (ATP) over a larger theory by designing *automated* methods for re-using lemmas from related problems. We are interested in proving theorems (and re-using lemmas) in general large-theory mathematics represented in the first-order TPTP format and giving rise to thousands of related problems, containing many formulas. We assume that there are many symbols in such problems and that they are named consistently across all the problems. Such problems are typically neither purely equational nor Horn nor EPR, and the strongest existing tools for them are refutational first-order ATPs such as Vampire [10] and E [18]. Hence our task revolves around the refutational proofs obtained from such ATPs. The important topics that need to be addressed are:

– How do we generate re-usable lemmas automatically from such ATP proofs?

– How do we automatically choose a set of good lemmas from related problems?
– What are good lemmas for a particular new ATP problem?
– How do we evaluate the usefulness of re-using lemmas in ATP?
– How much ATP performance can we gain by re-using lemmas?

There have been several lines of work in ATP related to these questions, we briefly mention those that are most relevant to our work. So far the most successful technique for re-using lemmas from ATP proofs has been Veroff's *hints* method [26]. It extracts lemmas from the (manually selected and semi-manually re-oriented) proofs produced by Prover9 and uses them for internally directing Prover9's given-clause loop on related problems. The main application have so far been problems in equational algebra [15]. A recent example where very long proofs of open conjectures are found thanks to this technique is the project **AIM**-ed at characterizing loops with **A**belian **I**nner **M**appings groups [9]. A similar technique that extracts and generalizes lemmas from previous proofs and uses them for proof guidance was implemented by Schulz in E prover as a part of his PhD thesis [17].

We have tried to experiment with this E technique on large-theory problems, so far without success.[4] Our very initial experiments (done with Veroff) with hints on large-theory problems have shown that unlike the equational proofs, the proofs of large-theory problems contain many (incompatible) skolem constants and steps depending on the negated conjecture, and thus are harder to re-orient into the *strictly-forward* proofs [9] from which lemmas derived only from the axioms and containing only known symbols can be extracted. A related issue is that the large-theory proofs seem to be much more heterogeneous than e.g. the AIM problems, likely requiring targeted selection of hints for a particular problem rather than unrestricted use of all available lemmas as hints. To address such issues, we instead proceed here as follows:

1. We extract all the direct (axiom-derived) skolem-free lemmas used in the proofs. These lemmas can be immediately re-used in other proofs.
2. To make other lemmas re-usable, we first attempt to heuristically *redirect* general refutational ATP proofs into Jaśkowski-style natural deduction proofs using the recent tools developed for Sledgehammer by Blanchette and Smolka [2,19], so that the proof steps (later translated into lemmas) only depend on axioms.
3. Then we *extract* and heuristically *deskolemize* lemmas from the redirected natural-deduction proofs, so that the lemmas only speak about symbols that are known in the original large-theory problems (and thus are re-usable).
4. We verify and optionally interreduce the lemmas.
5. Given a new conjecture C, we use several AI methods to estimate which of the previously extracted lemmas might be most useful for proving C. Various numbers of the best lemmas are then added to the axioms with which we try to prove C.

[4] Schulz confirms that the code has not been maintained and might need various updates.

The evaluation is done on the large-theory MPTP2078 benchmark [1], containing 2078 related problems in general topology (and related fields) extracted from Mizar. Note that large-theory techniques developed on one large-theory benchmark or corpus typically transfer well to other large-theory corpora [22,3]. In the following sections, we first describe in more detail the scenario and the techniques involved, and then we run ATPs on the benchmark with and without using such lemmatization methods, and evaluate their performance.

2 Lemmatization Scenario and Initial Statistics

Our goal is to prove as many problems over a large theory as possible. Concretely on the MPTP2078 benchmark, E prover (version 1.8) can prove in 60 seconds 569 of the 2078 *large* problems containing all previous premises (theorems, definitions and axioms). E can prove 1208 of the *small* versions of these problems, obtained by only giving E the premises that were needed for the (human-assisted) Mizar proofs.

The problems are chronologically ordered by their appearance in the Mizar library. We can assume that for a given problem P in the benchmark, all the lemmas found in all the previous proofs can be used for proving P. We can use both the lemmas from the large problems and from the small (human-assisted) problems, assuming that the mathematician needs to write the human-assisted formal proof regardless, even when the automation fails him, and that such human-assisted proofs can then be given to an ATP, which then may produce useful lemmas when running on such small problems.

The initial statistics of unmodified lemmas extracted from the 1208 proofs of the small problems is shown in Table 1. There are 75044 total lemmas when counting the same lemma multiple times (if it was created in multiple proofs). Only about half of them (38058) do not depend on the negated conjecture. About 60% (43995) of all lemmas contain a skolem symbol. Practically all those (96%) which depend on the negated conjecture contain a skolem symbol, but that is also the case for 22% of those that do not depend on the negated conjecture. This leaves only about 40% (29554) lemmas (with repetition) that are derived without the use of the negated conjecture and that do not contain any skolem symbol.

	lemmas	neg.-conj.-dependent	neg.-conj.-independent
all	75044	36986	38058
all skolem	43995	35671	8324
all no skolem	31049	1315	29554
unique	23764	13660	10104
unique skolem	18189	13616	4573
unique no skolem	5575	44	5531

Table 1: Initial statistics of lemmas from the small problems.

After approximate merging of the same lemmas from different problems,[5] these numbers are even smaller: the ratio of usable lemmas that are independent of the negated conjecture and do not contain skolem symbols drops to 23% of all lemmas, while 77% do contain skolem symbols and 57% depend on the negated conjecture.

This is a good motivation for trying methods that make the lemmas independent of the negated conjecture and remove the skolem symbols, thus making many more lemmas generally applicable in the next problems.

3 Extracting Reusable Lemmas from Refutational Proofs

The task of making proof steps independent of the negated conjecture is closely related to the task of human-level presentation of ATP proofs. There have been several tools attempting such human-level presentation, for example Tramp [13] and P.rex [4]. The most recent one that has been tested on a large number of problems is Blanchette and Smolka's ATP proof presentation toolchain made for the Isabelle/Sledgehammer framework [19]. This toolchain relies on Blanchette's *proof redirector* [2], which tries to reverse proofs by contradiction into direct proofs.

In more detail, Blanchette's tool takes a refutational TPTP proof and creates a natural deduction (Isabelle/Isar) proof which has as many forward steps as possible, i.e., as many steps derived from the axioms as possible. This is roughly done by reversing the part of the derivation graph that depends the negated conjecture. For example, a final step that derives $\bot$ from two lemmas ϕ_1 and ϕ_2 depending on the negated conjecture:

$$\phi_1, \phi_2 \vdash \bot \tag{1}$$

is redirected into:

$$\vdash \phi_1 \wedge \phi_2 \rightarrow \bot \tag{2}$$

Assuming further that ϕ_1 was derived using ϕ_0 which is also dependent on the negated conjecture and using some other lemmas that are not conjecture-dependent, we further get:

$$\vdash \phi_0 \wedge \phi_2 \rightarrow \bot \tag{3}$$

The inference step on the redirected lemmas (2,3) is then justified by referring to the exact same conjecture-independent lemmas that were used in the original proof to derive ϕ_1 from ϕ_0. This mechanism propagates through the lemmas dependent on the negated conjecture, ultimately deriving that the negated conjecture implies $\bot$, i.e., deriving the unnegated conjecture in a forward style.

While Blanchette's redirection tool works on propositional level, the whole framework (due to Blanchette and Smolka) also translates the ATP skolemization steps into natural deduction steps that fix universally or existentially quantified variables as local constants for parts of the proof. In general, the redirected Jaśkowski-style natural deduction proofs may also introduce assumptions.

[5] This merging is only approximate because for the purpose of this initial statistics we do not try to detect if the (serial) skolem names used in the different problems come from the same first-order formula or not.

3.1 Extracting Lemmas from the Natural Deduction Proofs

The above framework assumes that the first-order TPTP problems are a result of translating higher-order facts and conjectures written in Isabelle/HOL, and it ultimately tries to create a legal higher-order natural-deduction proof from the first-order TPTP proof that justifies the higher-order conjecture. In order to instead process an arbitrary first-order TPTP proof and to generate standard first-order lemmas, we do the following:

- We modify the tool to be able to start with arbitrary first-order TPTP proofs that have no Isabelle origin, by using empty internal Isabelle translation tables, not typechecking the terms and formulas in the resulting natural deduction proof, and writing a separate TPTP printer that prints such untyped proofs.
- We add flattening of the assumption and local-constant block structure of the natural deduction proofs, producing globally valid TPTP lemmas. This step is in principal similar to the earlier translation of the Jaśkowski-style Mizar proofs into TPTP derivations [23], however there are several differences discussed below.
- The modified functionality is then compiled into a standalone tool,[6] which can be used as an initial lemma extractor for any TPTP proof.

The flattening of the natural-deduction proofs proceeds by tracking the assumption and quantification structure leading to a particular statement in the natural-deduction proof, and performing universal quantification for each local constant introduced by Isar's "fix" step, implication for each supposition ("assume") step, and existential quantification for each local constant introduced by Isar's "obtain" step, changing the corresponding Isar local constants to the quantified variables.

This procedure is correct, i.e., it cannot generate a lemma that would not be provable from the initial axioms, and it has the desired property that the generated lemma will not contain new skolem symbols, thus making the lemma usable for proving the next conjectures. In order to achieve this, we however sacrifice completeness in some cases.[7] For example, when in the natural deduction proof a local constant c such that $q(c)$ is obtained from $\exists X : q(X)$, and in its scope statements $p(c)$ and $r(c)$ are proved, the extracted lemmas will be $\exists X : p(X)$ and $\exists X : r(X)$ instead of the stronger version $\exists X : (p(X) \wedge r(X))$. In the Mizar proof export this is handled via additional Henkin axioms about the local constants, however that means proliferation of such new constants in the lemmas, which we want to avoid here. A related completeness issue comes from the fact that proper TPTP skolem functions (not constants) are translated into higher-order constants by the proof presentation framework. During the

[6] `http://cl-informatik.uibk.ac.at/users/cek/frocos15/redirector/`

[7] We obviously do not lose completeness in general, because all the lemmas can be derived from the axioms, however we weaken or lose some of the lemmas during the translation process.

flattening we currently just skip generating all such lemmas instead of trying more advanced transformations. Figures 1,2,3 show a side-by-side example of the transformations done on a simple propositional proof. Figure 1 shows the TPTP proof starting with the conjecture g and eight axioms. The conjecture is negated, and the contradiction (final line) is derived in eight inference steps. In the corresponding Isar proof (Figure 2) we use a compressed notation: the numbers in brackets refer to the serially numbered assumptions (corresponding to the TPTP axioms), that are used to justify a particular step. There are various imperfections (acknowledged by Blanchette), for example $f \Rightarrow g$ is proved (and then extracted by us as a lemma in Figure 3) despite being an axiom. Note that many of the extracted lemmas in Figure 3 are implications whose antecedents correspond to the Isar assumptions (Isar keyword `assume`).

```
fof(0, conjecture, (g)).
cnf(25,axiom,(h|g)).
cnf(26,neg_conj,(~g), [0]).
cnf(32,axiom,(a|~h)).
cnf(33,neg_conj,(h), [25,26]).
cnf(39,axiom,(~a|~b)).
cnf(40,neg_conj,(a), [32,33]).
cnf(45,axiom,(c|b)).
cnf(46,neg_conj,(~b), [39,40]).
cnf(48,axiom,(g|~f)).
cnf(50,axiom,(d|~c)).
cnf(51,neg_conj,(c), [45,46]).
cnf(52,axiom,(f|~e)).
cnf(53,neg_conj,(~f), [48,26]).
cnf(54,axiom,(e|~d)).
cnf(55,neg_conj,(d), [50,51]).
cnf(56,neg_conj,(~e), [52,53]).
cnf(57,neg_conj,(),[54,55,56]).
```

Fig. 1: Original proof

```
lemma assumes
    "a ∨ ¬h" "a ⟹ ¬b"
    "¬b ⟹ c" "c ⟹ d"
    "d ⟹ e" "e ⟹ f"
    "f ⟹ g" "¬h ⟹ g"
  shows "g"
proof -
  have "d ⟶ f" (5,6)
  moreover
  { assume f
    hence g (7) }
  moreover
  { assume "¬ d"
    hence "¬ c" (4)
    hence b (3)
    hence "¬ a" (2)
    hence "¬ h" (1)
    hence g (8) }
  ultimately show g
qed
```

Fig. 2: Isabelle proof

```
fof(53___55_0,plain,(d=>f)).

fof(9_0,plain,(f=>g)).

fof(51_0,plain,(~(d)=>~(c))).
fof(46_0,plain,(~(d)=>b)).
fof(40_0,plain,(~(d)=>~(a))).
fof(33_0,plain,(~(d)=>~(h))).
fof(9_0,plain,(~(d)=>g)).
fof(0_0,plain,g).
```

Fig. 3: Lemmas

4 Filtering Lemmas

We run the lemma extractor on all TPTP proofs obtained from all the small MPTP2078 problems, taking 260 s in total. For five proofs the redirection phase runs out of memory, and from the rest we extract altogether 3394 *plain* (derived from axioms only) lemmas and 6328 *negated* (originally depending on negated conjecture) lemmas. The plain lemmas are in general much smaller (178 bytes on average) than the negated ones (526 bytes), typically because the redirection process adds assumptions to such lemmas.

Note that the number of plain lemmas produced by the redirector is much less than the 29554 plain skolem-free lemmas obtained by the direct extraction in Section 2. This is mainly due to de-duplication (not) applied at different stages and other small differences.

Re-proving and tautology removal: The first filter that we apply is fast (1 second) proving of each lemma from its problem's axioms, removing those that do not need any axiom (tautologies), and ending up with 5183 unique directly extracted provable lemmas (**pla**), 961 unique plain redirected lemmas (**plr**), 3538 unique negated lemmas (**neg**) of which 1617 are unique negated lemmas that do not contain conjunctions (**nmu**). In total there are 4377 unique redirected lemmas (**red**), and combined with pla there are 9057 lemmas in total (**all**). 768 of these lemmas are (after α-normalization) identical to some of the 4564 original MPTP2078 formulas.

The re-proving is done to be sure that we do not introduce unsoundness (which could then prove all the remaining problems) in the extraction phase. In Isabelle, an occasional error in the proof translation would not be an issue, because the translated proofs are ultimately checked by Isabelle's LCF kernel. Another reason for the re-proving is that we want to determine the exact proof dependencies of each lemma. This is an important information for learning how to use the lemmas and other formulas for future proofs.

Making lemmas usable for future problems: Each lemma is inserted into the chronological sequence of all premises, right *after* the theorem in which it proved for the first time (and possibly other lemmas generated in its proof). Even though the lemmas were proved *before* their theorem, inserting them before it would often result in very simple new proofs of the lemma-enriched large problems, because some of the new lemmas are very close to their theorems. In general, the lemma only becomes known after the proof is found, so we only allow to use the lemma for the theorems that follow the problem from which the lemma was first extracted.

Updating the dependencies: One of the main factors when selecting the most suitable lemmas for a problem is the information about how each lemma was previously used, and also how it was proved. This *dependency information* is added to the set of proof dependencies that we already have for the main theorems and lemmas. After this addition, we have three sets of dependencies for further experiments:

old: We add the new lemmas (if any) into the chronological sequence as described above, but do not add any information about their dependencies neither about dependencies on them.

all: For each new lemma, we also add the dependency on the axioms from which it was proved. So far we do not use dependencies between the lemmas.

fut: For each original proved theorem, we also add its dependency on all the lemmas that were extracted from its proof. We call these *future* dependencies, because as mentioned above, we allow these lemmas to be used only after the theorem is proved.

4.1 Additional Filters

On the set of all (possibly redirected) reproved lemmas (**all**) combined with the original MPTP2078 formulas we further apply the three additional filters

explained below. We do not filter the other sets (**pla, plr, red, neg, nmu**) here because they are already sufficiently small.

Forward subsumption: We use the MoMM [21] subsumption tool derived from E's perfect discrimination trees on the lemmas in their order, disallowing backward subsumption, so that future stronger lemmas cannot remove earlier weaker versions. Such weaker version might be useful before the stronger lemma is proved later. This optional filter can remove 6956 of the new lemmas which are subsumed by another (older) lemma or by an existing axiom/theorem. 2101 of the **all** lemmas are left after this optional phase (**mom**). The disadvantage of such interreduction is that suitable frequently derived instances disappear in this way, and that such instances may contain symbols that make them more eligible for selection when using further similarity-based filters.

PageRank: One of the graph-based filters we experiment with is PageRank [14] used on the three graphs of direct proof dependencies (each lemma/theorem points to those used in its proof). It takes 0.574 s to compute the ranks of all the (about 13k) nodes. We then choose only the best 2048 new lemmas according to their PageRanks (**pgr**), which are then handed over to the final problem-specific premise selectors.

AGIntRater [16] is a tool that tries to compute important characteristics of the lemmas in ATP proofs, producing an aggregated *interestingness* rating for each lemma. AGIntRater fails to rate the complete set of new lemmas with dependencies (fut), likely because of the size of the dependency graph. Instead, we run AGIntRater on all the small proofs. We have considered computing the sums, averages and maximums of the lemma ratings across all the proofs for each unique lemma, however it turns out that many of the positive ratings are for the CNF transformations that do not give rise to new lemmas. Only 3564 of the new lemmas ever have nonnegative rating (**ag0**). We have also created an even more strict selection (**ag5**), where only the 1150 lemmas with average interestingness rating at least 0.5 were added.

5 Problem-specific Premise Selection

Each of the above dependency sequences (**old, all, fut**) restricted to the preselected lemmas (**all, pla, plr, red, neg, nmu, mom, pgr, ag0, ag5**) provides information about how theorems and lemmas were proved. This information, together with suitable characterization (features) of the theorems and lemmas is incrementally learned from and used for each MPTP2078 theorem T to rank the preceding theorems and lemmas according to their estimated relevance for proving T.

For this we run two fast and scalable learning-based premise selectors: our currently strongest version of distance-weighted k-nearest-neighbor (k-NN) learner and our implementation of the naive Bayes learner [3]. Both methods use an

IDF-weighted combination of symbol and term features for formula characterizations [5]. For each original MPTP2078 theorem we thus obtain (by training the learners on the previous proof dependencies) a ranking of the set of previously available MPTP2078 formulas and the added lemmas. Table 2 shows how often the k-NN predicts the new lemmas among the first n predictions when using the **all** proof dependencies (only several interesting sequences are shown). For

Lemma selection	first	10-first	100-first	1000-first
pla	0.57	3.46	16.64	32.56
red	0.48	4.06	22.44	40.61
ag0	0.68	4.94	20.89	36.09
mom	0.72	3.61	15.32	28.91
pgr	0.34	5.18	20.43	32.78

Table 2: Ratio (in %) of the new lemmas in the first n k-NN predictions for several lemma-selection methods and the **all** sequence.

comparison we also add to the evaluation the normal non-lemmatizing premise selection method (**non**).

6 Evaluation

All the data, tools, and statistics for this paper are available at our web page.[8] In particular, the full versions of the tables shown in this section are online.[9]

There are several approaches to evaluating the improvement. First, we can compare the ATP performance of the best methods with and without lemmas, i.e., in both cases after choosing the best-performing combination of learning-based selection with the underlying lemmatizing method. To find such best combinations, we try to prove each theorem with the best-ranked selections (segments) of 16, 32,64,...,2048 MPTP2078 formulas and lemmas, using a 30 s time limit. As the underlying ATP we always use E 1.8 running in its automated mode. Note that E itself runs many ATP strategies for each problem in its automated mode. These strategies are selected for each problem individually by a machine-learning system developed by Schulz, based on suitable problem characteristics and performance of the strategies on a large set of problems.

Table 3 compares the best results achieved with and without lemmas for each number of the best-ranked premises tried, showing also the relative improvement for each premise number. The complete table of the 336 combinations is available online.[10] The best lemma-based method (k-NN, fut, all lemmas, 128 best-ranked

[8] `http://cl-informatik.uibk.ac.at/users/cek/frocos15/`

[9] `http://cl-informatik.uibk.ac.at/users/cek/frocos15/statistics/`

[10] `http://cl-informatik.uibk.ac.at/users/cek/frocos15/statistics/all-single-statistics.html`

premises) proves 936 theorems, while the best non-lemmatizing method (k-NN, old, no lemmas, 128 best-ranked premises) proves 878 theorems, i.e., 6.6% less.

The improvement from lemmatization is relatively low – 3.5% – when using only 16 best-ranked premises (618 by k-NN/all/ag0 versus 597 by k-NN/old/non). This rises to 14.8% when using 256 premises and peaks at 20.9% when using 512 premises (851 by k-NN/fut/all versus 705 by NB/old/non), which is a very significant improvement. Figure 4 shows the success rates for these premise numbers

Premises	16	32	64	128	256	512	1024	2048
Lemmas	618	820	926	**936**	915	851	724	657
Old	597	797	877	**878**	797	705	627	551
Improvement (%)	3.5	2.9	5.6	6.6	14.8	**20.7**	15.5	19.2

Table 3: Comparison of the best methods for the 8 premise-selection sizes.

for the different lemmatizing strategies, each aggregated across the two premise selectors and the various methods of constructing the dependencies.

There are several effects involved in these results. At the low premise selection numbers, the main challenge is to select premises that really justify the conjecture, i.e., which do not leave any countermodels left (see e.g., SRASS [20] and MaLARea-SG1 [24] for more detailed experiments). For this, the original Mizar library theorems seem to be quite well designed, and only a few of the strongest lemmas – in particular the ones chosen by AGIntRater and PageRank – help to increase the performance by 3.5%. On the other hand, when allowing many premises, insufficient logical power of the premises is usually no longer an issue, and the main problem is to focus the proof search towards the conjecture. Such focusing is the core of Veroff's hints method, which is likely to some extent being emulated at the higher premise numbers by adding some previously useful conclusions of the main library theorems. A related effect that we have quite often observed, is that during the premise selection the more conjecture-related or more instantiated lemmas replace some less related or more general library theorems, which in the no-lemma case more easily confuse the proof search. Lastly, even if no new lemma is eventually used in a proof, it happens quite often that the new proof dependencies created with the help of the added lemmas make it easier for the machine learners to choose the right Mizar theorems for the proof. The latter effects – useful instantiations, many conjecture-related lemmas replacing other theorems, and more data for learning – likely also explain the relatively low performance of the lemmas interreduced by forward subsumption (MoMM) compared to only α-normalized lemmas (all).

Another way to compare the methods is to look at the aggregated results (unions of problems solved) across the two machine learners and the eight premise numbers. This is shown in Table 4. The best new method – k-NN/ag0 – solves in total 1268 problems, compared to 1217 solved without lemmas, i.e., 4.2% more. Note that just using all lemmas, relying only on learning-based premise selection

without any further filtering does not perform much worse (1262 problems). In total, the union of all problems solved by the new and old methods is 1375 problems, compared to 1217 without lemmas, i.e., 13.0% more. Such comparison is however unrealistic, because the total time spent on all the new combinations together is much higher than the total time spent on the old ones.

A fairer way how to do such total comparison is to give the new methods only as much time as is needed to solve the 1217 problems by the old methods, i.e., in our case allowing only 14 most complementary new methods, see Table 5. As common in such evaluations [6,8,12] the most complementary methods are computed by a greedy algorithm, and the resulting greedy sequences are shown from top to bottom in the table. The total improvement is in this case 6.2%, i.e. a comparable result to the 6.6% improvement obtained by comparing only the best single methods.

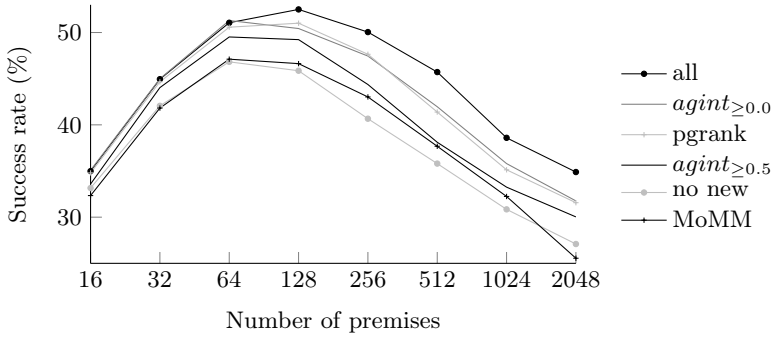

Fig. 4: ATP success rates over 8 premise-selection sizes for several strategies.

7 Examples of New Lemmas

While the statistics in the previous section gives an global overview of the strength of the methods, it is also interesting to inspect several examples of new proofs found thanks to the added lemmas.

1. The first Mizar theorem about basic set operations (in this case symmetric difference), `XBOOLE_0:1`,[11] states that:

   ```
   x in X \+\ Y iff ((x in X & not x in Y) or (x in Y & not x in X)).
   ```

   The ATP proof of this fact includes a CNF statement that in the Mizar syntax would read:

   ```
   X1 in X3 implies not X1 in X2 \ X3.
   ```

[11] `http://mizar.cs.ualberta.ca/~mptp/7.11.07_4.160.1126/html/xboole_0.html#T1`

Strategy	Dependencies	Proved	%	Unique
ag0	fut	1268	61.020	3
all	fut	1262	60.731	1
all	all	1253	60.298	1
ag0	all	1247	60.010	1
pla	all	1247	60.010	0
pgr	all	1242	59.769	5
pgr	fut	1240	59.673	3
pla	fut	1236	59.480	2
ag5	fut	1235	59.432	1
ag5	all	1233	59.336	0
red	fut	1230	59.192	0
red	all	1228	59.095	0
neg	fut	1227	59.047	1
plr	fut	1225	58.951	1
mom	all	1222	58.807	1
plr	all	1222	58.807	2
non	old	1217	58.566	0
all	old	1216	58.518	3
neg	all	1215	58.470	0
any		1375	66.169	

Table 4: Aggregated ATP results across the premise-selection sizes.

	With lemmas					No lemmas			
Strategy	Predict	Deps	Prems	Proved	%	Predict	Prems	Proved	%
all	knn	fut	0128	936	45.043	knn	0128	878	42.252
all	knn	all	0032	1046	50.337	knn	0032	1031	49.615
pla	knn	all	0256	1141	54.909	nba	0256	1084	52.166
ag5	nba	fut	0064	1175	56.545	nba	0064	1124	54.090
mom	knn	all	0128	1197	57.603	knn	0016	1145	55.101
ag0	knn	fut	0016	1218	58.614	nba	0512	1164	56.015
all	knn	fut	2048	1235	59.432	knn	0064	1176	56.593
pgr	nba	fut	0512	1248	60.058	nba	0016	1185	57.026
nmu	nba	fut	0064	1258	60.539	nba	0128	1193	57.411
pgr	knn	all	0064	1267	60.972	nba	0032	1200	57.748
nmu	nba	fut	0256	1274	61.309	knn	2048	1207	58.085
all	knn	old	0016	1280	61.598	knn	0256	1213	58.373
nmu	knn	fut	0064	1286	61.886	knn	0512	1215	58.470
pla	knn	all	0128	1292	62.175	nba	1024	1217	58.566

Table 5: Greedy sequence of aggregated ATP results with lemmas compared with the numbers of lemmas proved by running the ATP without lemmas.

This lemma could be easily derived as a consequence of more general facts already present in the Mizar library, however it does help in a number of fu-

ture ATP proofs. For examples it lets the ATPs prove XBOOLE_1:40,[12] which states:

```
(X \/ Y) \ Y = X \ Y,
```

and to prove ZFMISC_1:72:[13]

```
{x,y} \ X = {x,y} iff (not x in X & not y in X).
```

2. Another example is a new lemma derived by the ATP in the proof of XBOOLE_-1:1:[14]

   ```
   X1 \ (X2 \ X1) = X1,
   ```

   which is useful in the proofs of four more theorems (three of them in XBOOLE_-1, one in TOPS_1).
3. A more complicated new lemma is derived in WAYBEL_7:9:[15]

   ```
   with_infima(BoolePoset X).
   ```

   This is a simple consequence of two facts already present in the Mizar library, but it enables three new ATP proofs in the formalization of prime ideals and filters in WAYBEL_7. This is likely because a large number of such simple facts can be derived in this rich domain, and pointing out the relevant one makes the three proofs achievable.

8 Related Work

Some relevant related work such as Veroff's and Schulz's work is already mentioned in the introduction. A more extensive summary of the related methods is given in our paper on extracting and re-using the millions to billions lemmas arising in Interactive Theorem Proving (ITP) [7]. Some issues discussed here overlap to some extent with the ITP setting: for example the need for fast methods for filtering a large number of lemmas. The need for further fast filters is however not so big here: we can easily handle all lemmas (thousands) by the learning-based premise selectors and only use the additional filters to get better predictions, whereas in ITP (millions to billions lemmas) fast pre-filtering is crucial.

A number of further issues differ in the ATP setting: the lemmas need reorienting and deskolemizing, and the ATP proofs are short and suitable for ATP-style tools like AGIntRater. It is also worth mentioning that even in the ITP

[12] http://mizar.cs.ualberta.ca/~mptp/7.11.07_4.160.1126/html/xboole_1.html#T40

[13] http://mizar.cs.ualberta.ca/~mptp/7.11.07_4.160.1126/html/zfmisc_1.html#T72

[14] http://mizar.cs.ualberta.ca/~mptp/7.11.07_4.160.1126/html/xboole_1.html#T1

[15] http://mizar.cs.ualberta.ca/~mptp/7.11.07_4.160.1126/html/waybel_7.html#T9

setting, ATP proofs are typically a valuable source of training data for learning premise selection [11,6]. This means that the ATP and ITP lemma extraction could likely be fruitfully combined in the various strong [AI]TP "hammer" systems.

9 Conclusion and Future Work

We have introduced a toolchain for re-using lemmas across many related ATP problems, and evaluated it on the MPTP2078 large-theory benchmark. The main challenges are extraction of reusable context-independent lemmas from the ATP proofs, their subsequent filtering, and extracting suitable proof dependencies for learning premise selection. To make lemmas reusable, we first redirect them, using a modified version of the Isabelle/Isar proof presentation tools, and then we heuristically deskolemize them. The subsequent filtering is done by several tools – AGIntRater, PageRank, MoMM – that make use of different aspects of the proofs and lemmas. The filtered lemmas and their proofs result in modified proof dependencies, from which we learn along with the old proof dependencies, and use such learned knowledge to select premises for each MPTP2078 theorem. The 30 s improvement over the best old method is 6.6% more problems proved, and the improvement when using 14 most complementary methods is 6.2%. This comparison is done against the strategy-scheduling E prover, which itself runs a customized selection of strategies on each problem, choosing these strategies from a large portfolio. This means that the lemmatizing strategies add nontrivial performance to the E strategies. We have found that the new lemmas are particularly useful when using many premises, improving over the no-lemma case by about 15% and 20% when using 256 and 512 premises, respectively.

In the future we would like to experiment with running the toolchain on consistently pre-skolemized problems and on harder problems that miss some of the original MPTP2078 theorems. Another direction is to combine the filters and to use more of them, possibly on a larger dataset such as the whole MPTP-translated MML, using also methods for selecting lemmas from the ITP (Mizar) proofs [25,7]. We can also try more MaLARea-style iterations of the lemma-enrichment, i.e., extracting lemmas from the newly found problems and trying unsolved problems with such new lemmas added. And yet another direction is to re-use the filtering methods for Veroff-style hints selection and for improving given-clause guidance in ATPs by a large pool of previous lemmas.

Acknowledgments

We thank the CADE-25 referees for a number of useful comments to an early version of this paper. Kaliszyk was supported by the Austrian Science Fund (FWF): P26201, Urban's work was funded by NWO grant 612.001.208: *Knowledge-based Automated Reasoning*, Vyskočil's work was supported by institutional resources for research by the Czech Technical University in Prague, Czech Republic.

References

1. J. Alama, T. Heskes, D. Kühlwein, E. Tsivtsivadze, and J. Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2):191–213, 2014.
2. J. C. Blanchette. Redirecting proofs by contradiction. In J. C. Blanchette and J. Urban, editors, *PxTP@CADE*, volume 14 of *EPiC Series*, pages 11–26. EasyChair, 2013.
3. J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *J. Formalized Reasoning*, 2015. in press.
4. A. Fiedler. P.rex: An interactive proof explainer. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR*, volume 2083 of *LNCS*, pages 416–420. Springer, 2001.
5. C. Kaliszyk and J. Urban. Stronger automation for Flyspeck by feature weighting and strategy evolution. In J. C. Blanchette and J. Urban, editors, *PxTP 2013*, volume 14 of *EPiC Series*, pages 87–95. EasyChair, 2013.
6. C. Kaliszyk and J. Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014.
7. C. Kaliszyk and J. Urban. Learning-assisted theorem proving with millions of lemmas. *Journal of Symbolic Computation*, 69:109–128, 2015.
8. C. Kaliszyk and J. Urban. MizAR 40 for Mizar 40. *J. Automated Reasoning*, 2015. in press.
9. M. K. Kinyon, R. Veroff, and P. Vojtechovský. Loops with abelian inner mapping groups: An application of automated deduction. In M. P. Bonacina and M. E. Stickel, editors, *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *LNCS*, pages 151–164. Springer, 2013.
10. L. Kovács and A. Voronkov. First-order theorem proving and Vampire. In N. Sharygina and H. Veith, editors, *CAV*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.
11. D. Kuehlwein and J. Urban. Learning from multiple proofs: First experiments. In P. Fontaine, R. A. Schmidt, and S. Schulz, editors, *PAAR-2012*, volume 21 of *EPiC Series*, pages 82–94. EasyChair, 2013.
12. D. Kühlwein, J. C. Blanchette, C. Kaliszyk, and J. Urban. MaSh: Machine learning for Sledgehammer. In S. Blazy, C. Paulin-Mohring, and D. Pichardie, editors, *ITP 2013*, volume 7998 of *LNCS*, pages 35–50. Springer, 2013.
13. A. Meier. System description: Tramp: Transformation of machine-found proofs into nd-proofs at the assertion level. In D. A. McAllester, editor, *CADE*, volume 1831 of *LNCS*, pages 460–464. Springer, 2000.
14. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.
15. J. D. Phillips and D. Stanovský. Automated theorem proving in quasigroup and loop theory. *AI Commun.*, 23(2-3):267–283, 2010.
16. Y. Puzis, Y. Gao, and G. Sutcliffe. Automated generation of interesting theorems. In G. Sutcliffe and R. Goebel, editors, *FLAIRS*, pages 49–54. AAAI Press, 2006.
17. S. Schulz. *Learning search control knowledge for equational deduction*, volume 230 of *DISKI*. Infix Akademische Verlagsgesellschaft, 2000.
18. S. Schulz. System description: E 1.8. In K. L. McMillan, A. Middeldorp, and A. Voronkov, editors, *LPAR*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013.
19. S. J. Smolka and J. C. Blanchette. Robust, semi-intelligible Isabelle proofs from ATP proofs. In J. C. Blanchette and J. Urban, editors, *PxTP 2013*, volume 14 of *EPiC Series*, pages 117–132. EasyChair, 2013.

20. G. Sutcliffe and Y. Puzis. SRASS - a semantic relevance axiom selection system. In F. Pfenning, editor, *CADE*, volume 4603 of *LNCS*, pages 295–310. Springer, 2007.
21. J. Urban. MoMM - fast interreduction and retrieval in large libraries of formalized mathematics. *Int. J. on Artificial Intelligence Tools*, 15(1):109–130, 2006.
22. J. Urban. BliStr: The Blind Strategymaker. *CoRR*, abs/1301.2683, 2014. Accepted to PAAR'14.
23. J. Urban and G. Sutcliffe. ATP-based cross-verification of Mizar proofs: Method, systems, and first experiments. *MCS*, 2(2):231–251, 2008.
24. J. Urban, G. Sutcliffe, P. Pudlák, and J. Vyskočil. MaLARea SG1 - Machine Learner for Automated Reasoning with Semantic Guidance. In *IJCAR*, pages 441–456, 2008.
25. J. Urban, G. Sutcliffe, S. Trac, and Y. Puzis. Combining Mizar and TPTP Semantic Presentation and Verification Tools. *Studies in Logic, Grammar and Rhetoric*, 18(31):121–136, 2009.
26. R. Veroff. Using hints to increase the effectiveness of an automated reasoning program: Case studies. *J. Autom. Reasoning*, 16(3):223–239, 1996.