Search for faster and shorter proofs using machine generated lemmas

Petr Pudlák
Charles University in Prague
pudlak@artax.karlin.mff.cuni.cz

Abstract

When we have a set of conjectures formulated in a common language and proved from a common set of axioms using an automated theorem prover, it is often possible to automatically construct lemmas that can be used to prove the conjectures in a shorter time and/or with shorter proofs.

We have implemented a system that repeatedly tries to improve the set of assumptions for proofs of given conjectures using lemmas that it extracts from the proofs constructed by an automated theorem prover. In many cases it can significantly reduce the total time or the overall sum of the lengths of the proofs of the conjectures.

We present several examples of such sets of conjectures and show the improvements gained by the system.

1 Motivation

Imagine a professor of mathematics who gives the same lectures every year. She works in a standard theory and she has a fixed set of theorems that she wants to present and prove every year during the course. And because she wants to save her and her students' time, she would like to have as efficient proofs as possible. She would like to find such lemmas that would shorten the total time she has to spend for presenting the proofs.

Our aim will be to search for lemmas in such environment and try to use them to make the proofs of conjectures more efficient.

Finding useful lemmas has of course much more serious applications, namely reducing the amount of resources necessary to prove a particular set of conjectures or to prove conjectures that could not be proved without such lemmas.

Currently the system only restructures and compacts the proofs and doesn't handle conjectures that the prover was not able to prove at the beginning. It can only improve existing proofs. In future we hope to be able to also search for proofs of the conjectures that couldn't be proved from the initial set of assumptions.

2 Previous research

The idea of an automated discovery of lemmas or theorems is not a new one. There were many different approaches to solve this task.

Owen L. Astrachan and Mark E. Stickel [AS92] used the idea of reusing lemmas to speed up a model elimination theorem prover.

Art Quaife [Qua92] used Otter [McC94] to prove many fundamental mathematical theorems. He included the theorems he had already proved as assumptions for the more complicated ones. The sequence in which the theorems were proved was determined by Quaife, based on his mathematical knowledge.

Marc Fuchs, Dirk Fuchs and Matthias Fuchs sought for lemmas using genetic programming to improve tableau-based proof search [FFF99].

The HR system [Col02a], named after mathematicians Hardy and Ramanujan, uses a model generator to construct models based on a set of axioms, attempts to formulate conjectures and then prove them using an automated theorem prover. A brief description can be also found in [Col02b].

Larry Wos and Gail W. Pieper describe the technique of lemma adjunction in [WP03] and also discuss many different approaches for evaluating lemmas.

Article [SGC03] summarizes many different criteria for constructing and identifying quality lemmas. Humans generally use the *inductive approach* – from many similar problems they try to induce a more general conjecture. This approach requires good knowledge of the particular field of mathematic and also good mathematical intuition. Another approach described is *generative*, when more sophisticated techniques (for example syntactic manipulation) are used to construct new conjectures. An example of such a system is the HR [Col02b] program. The *manipulative approach* tries to construct interesting conjectures from already existing theorems. Finally, the *deductive approach* tries to automatically construct many logical consequences from a set of axioms using an automated theorem prover and then filter them and pick those that are interesting for the researcher.

The authors of [SGC03] also categorize different possible filters for interestingness. The filters include non-obviousness, novelty, suprisingness, intensity and usefulness.

Our approach is somewhat different from those mentioned above. It lies in between the manipulative and the deductive approach. We observe proofs of all the given problems and identify lemmas that are common to many of the proofs. The filter we develop and use for selecting good lemmas falls into the *usefulness* category - our measure is, how much each lemma could contribute to the proofs of other conjectures. Unlike other measures, this one can easily be evaluated by comparing different proofs conducted by the prover.

The lemmas that we produce are therefore interesting from the point of view of a machine. Hence, this can give us an interesting comparison between human and machine opinions on the usefulness of a lemma.

The ideas in the article [SGC03] and personal communication with many other researchers inspired our work which we present in this paper together with the empirical results we have obtained.

3 Overview of the system

In sequel we assume that a consistent theory is given with some (possibly infinite) set of axioms. All the formulas we work with are formulated in the language of the theory and the conjectures that are to be proved and the lemmas that are constructed are proved using axioms selected from the axioms of the theory.

The systems starts by proving the conjectures one by one. The proofs of the conjectures that were successfully proved are then analyzed by the system. Formulas that appear multiple times in the proofs are then used as additional assumptions when looking for less costly proofs of the conjectures. The system tries to optimize the set of such formulas. We will call such formulas *lemmas*.

3.1 Basic notions

Let us first define some basic notions we will use throughout the text.

Notation 1 (Axiom, conjecture, lemma, proof) *By an* axiom *we understand either an axiom of the underlying theory or a well known theorem of the theory that we use as an assumption.*

A conjecture *is a formula given on the input that is to be proved by the system. The system tries to find the most efficient proof of the conjecture using different* sets of assumptions*. Each such a set can contain some of the axioms, other conjectures or lemmas.*

A lemma *is a formula constructed by the system that is proved in a similar fashion as the conjectures are, and is used to improve the proofs of the conjectures or the proofs of other lemmas.*

A proof *is an output of a successful run of the prover. As we use a single theorem prover with the same settings on every run, the proof only depends on the conjecture being proved and on the set of assumptions being used.*

The aim of the system is to find such lemmas and such sets of assumptions for the conjectures and the lemmas that either the total time required to prove the conjectures and the lemmas or the overall size or length of the proofs is minimized.

3.2 System input

Initially, the system is given a set $\mathbf{C}$ of conjectures and for each conjecture C from $\mathbf{C}$ the system is given an initial set of assumptions $\mathbf{A}_C$. The set $\mathbf{A}_C$ consists of a subset of axioms of the underlying theory and (possibly) of some other conjectures from $\mathbf{C}$.

The idea is that the proofs of conjectures in $\mathbf{C}$ need not use the set of all the axioms of the theory (which might be infinite) and that some other conjectures from $\mathbf{C}$ might be useful. The user may have his own idea, which conjectures to add as assumptions to $\mathbf{A}_C$.

In the process of computation the system tries to optimize the set of lemmas and for each lemma the set of its assumptions.

3.3 System output

At the end the system outputs the conjectures given at the input along with the lemmas that participate in the fastest/shortest set of proofs. For each conjecture and each lemma it outputs the optimal set of assumptions it has found, in the sense described later.

4 An example

In our example we use a commonly used encoding to encode an axiomatization of propositional logic into terms, thus forcing the prover to use only the specified axioms and rules for inferences. Then we use the system to prove some basic propositional theorems.

Note that the automated theorem provers were already used to look for interesting axiomatics of propositional logic, for example [MVF+02]. However, our aim is not to look for a new or otherwise interesting axiomatic, we use the system to optimize the proofs of several theorems.

We will denote the code of a formula by an over-line. For example for a propositional formula φ we denote the term that codes it by $\overline{\varphi}$. Negation is coded by a unary function n and implication is coded by a binary function i. The fact that φ is a theorem $\vdash \varphi$ is coded by a unary predicate t. The following table summarizes the codes:

$\overline{\neg\varphi}$	$n(\overline{\varphi})$
$\overline{\varphi \to \psi}$	$i(\overline{\varphi},\overline{\psi})$
$\overline{\vdash \varphi}$	$t(\overline{\varphi})$

We use "$\to$" just for implication in propositional logic we are coding. We use "$\Rightarrow$" for implication in predicate logic in which the conjectures are presented and in which the prover actually works. Thus the statement "if φ is provable then $\psi \to \chi$ is provable" would be coded as $t(A) \Rightarrow t(i(B,C))$.

In order to increase the difficulty of the task we've used Meredith's single axiom for propositional logic. It has only a single axiom schema and a single rule.

The axiom schema is defined for any formulas φ, ψ, χ, ξ and η, but the theorem prover computes with their codes, which are represented by the variables A, B, C, D and E in the language of the theorem prover. As these variables are universally quantified, the prover can replace each variable by any coded formula. The same applies to the coded modus ponens rule.

The coded representations of the schema and the rule are:

$$\begin{array}{rl} \text{Meredith:} & \vdash ((((\varphi \to \psi) \to (\neg\chi \to \neg\xi)) \to \chi) \to \eta) \to ((\eta \to \varphi) \to (\xi \to \varphi)) \\ & t(i(i(i(i(i(A,B),i(n(C),n(D))),C),E),i(i(E,A),i(D,A)))) \\ \text{MP:} & \varphi, \varphi \to \psi \vdash \psi \\ & (t(A) \wedge t(i(A,B))) \Rightarrow t(B) \end{array}$$

This set of axioms was used as the initial set of assumptions for all the conjectures. No other dependencies between the conjectures were specified. The system was let to discover all the lemmas by itself from the proofs of the conjectures.

Table 1 shows the conjectures whose proofs the system was improving. The conjectures were proved using E prover[Sch02]. The cost of the proofs was measured in the number of processed clauses. This measure closely corresponds to the time the prover spends on a problem, but it is not dependent on the hardware the prover runs on. In the second column the conjectures are presented in their coded form. The third column shows the number of processed clauses the prover spent on each of the conjectures if they were proved only using the original assumptions. The fourth column shows the cost of the proofs after the system had finished the improvements using lemmas generated from the proofs. Note that to obtain the total cost of the final proofs we also have to include the cost of the proofs of the generated lemmas. The full listing of the generated lemmas is included in Table 10. As we can see, the total cost of proofs needed to prove the conjectures in this case was reduced to 3%.

5 Description of the system

Let us first discuss how the lemmas are generated. The modification of the set of assumptions of the conjectures by using the lemmas will be described in the next part.

conjecture	coding into terms	original cost	final cost
$\neg(\varphi \to \neg\psi) \vdash \varphi$	$t(n(i(A,n(B)))) \Rightarrow t(A)$	938	62
$\neg(\varphi \to \neg\psi) \vdash \psi$	$t(n(i(A,n(B)))) \Rightarrow t(B)$	1673	17
$\varphi \vdash \neg\varphi \to \psi$	$t(A) \Rightarrow t(i(n(A),B))$	31	4
$\psi \vdash \neg\varphi \to \psi$	$t(B) \Rightarrow t(i(n(A),B))$	13	3
$\neg(\varphi \to \varphi) \vdash \psi$	$t(n(i(A,A))) \Rightarrow t(B)$	41	8
$\vdash \varphi \to (\psi \to \varphi)$	$t(i(A,i(B,A)))$	17	4
$\varphi \to \neg(\psi \to \neg\chi), \varphi \vdash \chi$	$(t(i(A,n(i(B,n(C))))) \wedge t(A)) \Rightarrow t(C)$	287	20
$\vdash \varphi \to \varphi$	$t(i(A,A))$	15	2
$\vdash \neg\varphi \to (\varphi \to \psi)$	$t(i(n(A),i(A,B))))$	5731	4
$\vdash \neg\neg\varphi \to \varphi$	$t(i(n(n(A)),A))$	3297	7
$\vdash \varphi \to \neg\neg\varphi$	$t(i(A,n(n(A))))$	3682	4
total cost of the conjectures:		15725	135
cost for proving the lemmas:			375
total cost:		15725	510

Table 1: Formulas proved from the Meredith's axiomatization.

5.1 Generation and evaluation of lemmas

Notation 2 (Subsumption) *By $c \sqsubseteq d$ we denote that the clause c that subsumes the clause d.*

Let a set of proofs[1] $\mathbf{P}$ be given. We collect all the clauses that appear in the proofs and that were derived only from the assumptions into a single set

$$\mathbf{L} = \bigcup_{P \in \mathbf{P}} \{c \,|\, c \text{ is a clause in } P \text{ derived only from assumptions and without Skolem symbols}\}$$

We do not include clauses that were derived from the negated conjecture, because they are not true formulas in our theory. Also we remove those clauses that contain Skolem symbols created by the prover, because these symbols have different meaning in different runs of the prover. (In future we might implement the reverse skolemization algorithm [CP80, CP93] to deal with such formulas.)

Thus, the clauses in $\mathbf{L}$ don't contain any skolem symbols, but they have no special form, they can contain arbitrary number of positive and/or negative literals.

Note that although it would be an obvious thing to do, we don't use the conjectures as lemmas. The reason is that the conjectures are not necessarily clauses, therefore we can't process them the same way as the clauses produced by the prover. As it turns out, many of the conjectures are then anyway discovered as lemmas by the system. However, this issue deserves a better solution in the future.

From the set $\mathbf{L}$ we construct a minimal $\mathbf{L}'$ set with respect to subsumption such that $\mathbf{L}'$ has the following properties:

1. $\mathbf{L}' \subseteq \mathbf{L}$, therefore $\mathbf{L}'$ contains only true formulas.

[1] Recall that for us a proof is a successful run of the prover that proves a particular conjecture from a given set of assumptions.

2. for every $d \in \mathbf{L}$ there is $c \in \mathbf{L}'$ such that $c \sqsubseteq d$.

3. for any pair $c \in \mathbf{L}'$, $d \in \mathbf{L}'$ we know that c doesn't subsume d: $c \not\sqsubseteq d$.

Therefore, $\mathbf{L}'$ contains just the most general variants of the lemmas appearing in $\mathbf{L}$.

The clauses from $\mathbf{L}'$ are then used as lemmas to modify the set of assumptions of the conjectures.

Now let us have a clause c that is a lemma, $c = L_1, \ldots, L_k, \neg L_{k+1}, \ldots, \neg L_n$ where L_i are atomic formulas. Let $|L_i|$ be the number of function symbols appearing in L_i. Let P by a proof whose set of assumptions we want to improve by c. We would like to have an estimate that would tell us, if it is likely that c will contribute to the proof P. A natural idea suggests itself that the more often an atomic formula L_i occurs in the proof P the more likely the lemma will contribute and also that the longer L_i is (measured in the number of symbols) the more likely it will shorten/speed up the proof. Therefore our estimation formula is defined by

$$\mathrm{weight}(c,P) = \sum_{i=1}^{n} |L_i| \cdot (\textit{the number of occurrences of } L_i \textit{ in } P) \tag{1}$$

There are many other possible estimation methods. One may, for example, take into account the number of formulas in the proof which the lemma subsumes, look for similar terms in the lemma and in the proof, etc.

5.2 Evaluation of the proofs

In order to evaluate the applicability of lemmas, we need to have a criterion for the cost of a proof:

Notation 3 (Measure of a proof) *A* proof measure *is a function that maps proofs into non-negative real numbers.*

We use the following proof measures:

the number of processed clauses reported by the prover; this measure is closely related to the time the prover spends while searching for the proof of the conjecture, but is independent of the hardware of the computer the prover runs on;

the length of the proof is the number of formulas appearing in the proof that is constructed by the prover;

the size of the proof is the total number of occurrences of function symbols (not predicate symbols) appearing in the proof that is constructed by the prover.

The length and the size of a proof are also independent on the hardware of the computer being used.

Notation 4 *If we are proving a conjecture C from assumptions $A_1, \ldots, A_n$, we denote the measure of the proof by*

$$||A_1, \ldots, A_n \rhd C||$$

If the prover is not able to conduct the proof we set

$$||A_1, \ldots, A_n \rhd C|| = +\infty$$

Remark 1 (The proof size/length) *If we could prove all the given conjectures together, the size/length of the resulting hypothetical proof would be of course less than the sum of the sizes/lengths of the individual proofs of the conjectures. But in most cases the prover is not able to prove all the conjectures together, therefore the system proves the conjectures one by one. The system then tries to compact the proofs of the conjectures to make them closer to the size of the hypothetical proof.*

The number of generated lemmas is usually very large, so only some of them will be included in the output. Such lemmas are marked as *accepted*. Each lemma is initially unaccepted. When the system figures out that the lemma is worth including in the output, it marks it as accepted.

For each conjecture or lemma C the system maintains a list $\mathbf{S}_C$ of sets of assumptions that were used to produce different proofs of the conjecture.

Each such a set $S \in \mathbf{S}_C$ is marked as *accepted* iff all the lemmas it contains are accepted. The initial set of assumptions of each conjecture contains no lemmas, hence it is always accepted. Now, we may define:

Notation 5 (Best accepted set of assumptions) *The best accepted set of assumptions*

$$S_C^{best} \in \mathbf{S}_C$$

of a conjecture C is an accepted set of assumptions that produces the best proof of C with respect to the proof measure:

$$\forall S: \ ((S \in \mathbf{S}_C) \wedge (\textit{all the lemmas in S are accepted})) \Rightarrow ||S \rhd C|| \geq ||S_C^{best} \rhd C|| \qquad (2)$$

If there are several sets of assumptions that match the criteria for the best accepted set (they produce proofs of the same measure), we arbitrarily choose one among them.

Let us call the proof of C produced from S_C^{best} the best accepted proof of C.

Remark 2 (System output) *The output of the system consists of all the input conjectures and all the accepted lemmas at the time the system finishes execution. For each of these conjectures or lemmas the best accepted set of assumptions is presented.*

5.3 Modifying the set of assumptions to get better ones

The outline of the work of the system is as follows:

1. The system first tries to prove all the given conjectures one by one. Let $\mathbf{C}$ be the set of those conjectures that were successfully proved. Let $\mathbf{L}$ be the set of constructed lemmas and let $\mathbf{L}^a \subseteq \mathbf{L}$ be the set of lemmas that are accepted. Initially, these sets are empty.

2. The system takes the best accepted proofs of all the conjectures and accepted lemmas. From those proofs the system constructs new lemmas as described in section 5.1. It sets

$$\mathbf{L} = \mathbf{L} \cup \{\text{the newly constructed lemmas}\}$$

3. The system produces pairs consisting of a best assumption set of a conjecture and of a lemma that will be used to improve the set of assumptions of the best proof of the conjecture. It takes all the new lemmas together with the lemmas it already has constructed before and combines them with all the best accepted sets of assumptions of all the conjectures and accepted lemmas. This way it produces every possible pair of

$$\mathbf{L} \times \{S_C^{\text{best}} | C \in \mathbf{C} \cup \mathbf{L}^a\}$$

   However some pairs of a lemma l and the best set of assumptions S_C^{best} of a conjecture C have to be excluded, namely

   - if already $l \in S_C^{\text{best}}$, or
   - if the lemma l is the conjecture C itself, or
   - if the conjecture C is directly or indirectly used to prove l; this means that either C is one of the assumptions in the best accepted set of l, or it is one of the assumptions in the best set of some conjecture that is an assumption of l, and so on.

   Otherwise it could happen for example that there would be two equivalent lemmas, one proving another with a one-step proof.

4. These pairs are sorted according to the estimate described in section 5.1.

5. The system subsequently takes these pairs (l, S_C^{best}) starting with the one with the highest weight:

   (a) It tries to conduct a new proof of C using $S_C^{\text{best}} \cup \{l\}$. The gain of this single improvement is
$$g = ||S_C^{\text{best}} \rhd C|| - ||S_C^{\text{best}}, l \rhd C||$$

   (b) If g is positive, it means that the lemma l has brought an improvement. The lemma l is then checked, if its total gain to all the conjectures whose set of assumptions were improved by l is larger than the cost of the proof of l. If so, the system sets

$$\mathbf{L}^a = \mathbf{L}^a \cup \{l\}$$

   which means that l is marked as accepted.

   If l is accepted, we want to incorporate the lemmas that originate from the proof of l as well as the lemmas that originate from the proofs of the conjectures that use l as an assumption. Hence, in such a case the process is restarted from the beginning and the system goes again to point 2.

   (c) Until there are any unprocessed pairs (l, S_C^{best}) the system takes the next pair and goes again to 5a.

6. When all the pairs are exhausted and no improvement was made the system outputs the conjectures $\mathbf{C}$, the accepted lemmas $\mathbf{L}^a$ and their best accepted sets of assumptions S_C^{best}, $C \in \mathbf{C} \cup \mathbf{L}^a$, as described above, and halts.

It may happen that as the result of adding l, some of the assumptions in S_C^{best} are not needed any more and do not appear in the proof. In such a case the system omits them and tries to prove the conjecture only using this reduced set of assumptions. If the proof attempt succeeds, it is evaluated the same way as described above. This practice helps to reduce the number of assumptions appearing in the proofs. Without it, the prover would soon become overwhelmed by the number of assumptions and no further improvement would be possible. In the current version of the system this check is not iterated, so the proof conducted from the reduced set of assumptions is not checked again for redundant assumptions.

6 Experimental results

The system is still under development, therefore we don't have an in-depth statistics of its behavior. However, we have performed tests on different sets of conjectures from different sources and we present a selection of them.

All experiments were run on a Linux machine with Intel® Pentium® 4 CPU 3.4GHz with 2GB of RAM. The conjectures were proved using E prover with resource limits set to 80 seconds, 512MB RAM and 100000 processed clauses.

We have developed an independent server component that lies between the actual system and the automated prover. The component caches all conducted proofs on a hard-disk and if a client requests the server to perform a proof that has already been processed, the server returns the cached version. This greatly speeds up the development of the system, particularly if we rerun the system many times with different settings or with slight modifications on the same set of problems.

For some cases we include a full list of the conjectures and the lemmas, for others we only summarize the results.

6.1 TPTP problems – set theory

We used several selected TPTP [SS98] problems concerning set theory. These problems originate from [Qua92]. For presentation in this paper we have converted the machine syntax into a more human-readable form. The meaning of the symbols that we use is summarized in Table 2. The conjectures that were proved are shown in Table 3. The table also shows the results obtained when running the system with the proofs being measured by the number of processed clauses. The second column labeled "level" is an inductively defined property defined for both the conjectures and the lemmas. The axioms have level 0. If a conjecture or a lemma is proved just from the axioms, it has level 1. Each lemma or conjecture has its level set to the maximum level of its assumptions plus 1. This can give us an approximation on how complicated the conjecture or the lemma is. The third column labeled $||\cdot||_i$ shows the initial cost of the proof of the conjecture while the fourth column labeled $||\cdot||_f$ shows the final cost of the proof of the conjecture when the system terminates. The fifth column shows the actual formula converted into a human-readable form. The last column shows which lemmas were finally used to prove the conjecture.

Some of the conjectures in the listings may have levels greater than 1 although they don't use any lemmas or the level of a particular conjecture is higher than one plus the level of the lemmas that are used to prove it. The reason is that the user has specified that some other

$A \subseteq B$	subclass
$\{A,B\}$	unordered pair
$\langle A,B\rangle$	ordered pair
$\{A\}$	singleton
$A_{[1]}$	first member of an ordered pair A
$A_{[2]}$	second member of an ordered pair A
$A \in B$	membership
$\overline{A}$	class complement
$A \times B$	cross product
$\emptyset$	null class
$A \cap B$	intersection
$\mathcal{U}$	universal class
member_of(A)	choice function (from the axiom of choice)

Table 2: Meaning of the symbols used in the TPTP set theory sample

conjectures should be used as assumptions to prove the conjecture and they increase the level of the conjecture.

We can see that often the conjectures that were harder to prove with respect to the proof measure have a higher level. This means that they were finally proved using several layers of lemmas.

Several conjectures that were too complicated are omitted for brevity.

The lemmas that the system generated with the processed clauses count proof measure are in Table 4. The lemmas are clauses, but for the presentation we have converted them into a more readable form using implication notation. As the lemmas have no initial set of axioms given by the user, they also have no initial cost, so only their final cost is shown.

They are sorted by their total accumulated improvement with respect to the proof measure, the more useful lemmas are at the beginning of the table. This is however only an informative ordering, as it depends very much on the order in which the lemmas were tried. For example, consider some two almost same lemmas l_1 and l_2. Whichever is chosen second will bring no improvement as the assumption sets were already improved by the one chosen first.

The lemmas are more or less complicated formulas that the system found useful for proving the conjectures. This can give us an interesting comparison, as the input conjectures were selected by a human, whereas the lemmas were constructed by a machine. Tables 5 and 6 show the results on the same set of conjectures when different proof measures were selected. Many of the lemmas appear in all three tables, although in different positions. Such lemmas seem to be essential for the automated prover when working with this particular theory.

Note that some of the lemmas the system has found are just conjectures reformulated as clauses. In such a case the system usually proves such conjecture using the lemma in a single step and then further improves the proof of the lemma.

Table 7 shows the summary of achieved results. For each proof measure the initial and the final cost of the proofs is shown.

no.	level	$\|\|\cdot\|\|_i$	$\|\|\cdot\|\|_f$	formula	lemmas
C1	1	1	1	$\forall X : X = X$	
C2	1	5	2	$\forall X : (X \subseteq X)$	
C3	1	2	2	$\exists X : \forall Z : \neg(Z \in X)$	
C4	2	6	2	$\forall X : (\emptyset \subseteq X)$	L18
C5	2	16	9	$(\emptyset \in \mathcal{U})$	L5
C6	2	4	3	$\forall X,Y : ((Y \in \{X\}) \Rightarrow Y = X)$	L14
C7	1	3	3	$\forall Z : (Z = \emptyset \vee \exists Y : (Y \in Z))$	
C8	1	2	2	$\forall X : (\{X\} \in \mathcal{U})$	
C9	2	11	4	$\forall X : ((X \subseteq \emptyset) \Rightarrow X = \emptyset)$	L18 L11
C10	2	8	2	$\forall X,Y : (\{X\} \in \langle X,Y\rangle)$	
C11	3	19	4	$\forall X,Y : ((X \in Y) \Rightarrow (\{X\} \subseteq Y))$	L1
C12	1	9	9	$\forall X,Y : \neg(Y \in (\overline{X} \cap X))$	
C13	1	2	2	$\forall X,Y : (\langle X,Y\rangle \in \mathcal{U})$	
C14	1	11	11	$\forall X,Y,Z : (((X \subseteq Y) \wedge (Y \subseteq Z)) \Rightarrow (X \subseteq Z))$	
C15	2	7	3	$\forall X : ((X \in \mathcal{U}) \Rightarrow (X \in \{X\}))$	L12
C16	1	22	21	$\forall X,Y : (\{X,X\} \subseteq \{X,Y\})$	
C17	2	9	4	$\forall X : ((X \in \mathcal{U}) \Rightarrow \{X\} \neq \emptyset)$	L12
C18	2	21	3	$\forall X : (\neg(X \in \mathcal{U}) \Rightarrow \{X\} = \emptyset)$	L8
C19	1	3	3	$\forall X : (\{member\_of(X)\} = X \Rightarrow (X \in \mathcal{U}))$	
C20	2	7	2	$\forall X,Y : (\{X,\{Y\}\} \in \langle X,Y\rangle)$	
C21	1	10	10	$\forall X,Y : ((\{member\_of(X)\} = X \wedge (Y \in X)) \Rightarrow member\_of(X) = Y)$	
C22	6	87	25	$\forall X,Y : ((X \subseteq \{Y\}) \Rightarrow (X = \emptyset \vee \{Y\} = X))$	L16
C23	1	19	10	$\forall X,Y : ((\{X\} = \{Y\} \wedge (Y \in \mathcal{U})) \Rightarrow X = Y)$	
C24	3	15	10	$\forall X,Y : ((\{X\} = \{Y\} \wedge (X \in \mathcal{U})) \Rightarrow X = Y)$	L12 L17
C25	2	12	4	$\forall X,Y : ((X \in \mathcal{U}) \Rightarrow \{X,Y\} \neq \emptyset)$	L9
C26	2	9	4	$\forall X,Y : ((Y \in \mathcal{U}) \Rightarrow \{X,Y\} \neq \emptyset)$	L12
C27	1	3	3	$\forall X : (\langle X_{[1]},X_{[2]}\rangle = X \Rightarrow (X \in \mathcal{U}))$	
C28	6	610	7	$\forall X,Y,Z : (((X \in Z) \wedge (Y \in Z)) \Rightarrow (\{X,Y\} \subseteq Z))$	L1 L2
C30	8	706	213	$\forall W,X,Y,Z : ((\langle W,X\rangle = \langle Y,Z\rangle \wedge (X \in \mathcal{U})) \Rightarrow X = Z)$	L3 L10 L5 L17 L9
C31	2	20	12	$\forall W,X,Y,Z : ((\langle W,X\rangle = \langle Y,Z\rangle \wedge (W \in \mathcal{U})) \Rightarrow W = Y)$	L14 L9
C32	2	32	4	$\forall X,Y : ((\neg(X \in \mathcal{U}) \wedge \neg(Y \in \mathcal{U})) \Rightarrow \{X,Y\} = \emptyset)$	L8
C33	2	16	11	$\forall X,Y,Z : (((Y \in \mathcal{U}) \wedge (Z \in \mathcal{U}) \wedge \{X,Y\} = \{X,Z\}) \Rightarrow Y = Z)$	L12 L14
C34	2	16	11	$\forall X,Y,Z : (((X \in \mathcal{U}) \wedge (Y \in \mathcal{U}) \wedge \{X,Z\} = \{Y,Z\}) \Rightarrow X = Y)$	L14 L9
C35	3	21	4	$\forall X,Y : (\{\{X\},\{X,\emptyset\}\} = \langle X,Y\rangle \vee (Y \in \mathcal{U}))$	L10

Table 3: Selected conjectures from the TPTP set theory sample with the proof cost measured by the number of clauses processed by the prover.

no.	level	$\|\|\cdot\|\|$	formula	lemmas
L1	4	32	$(A \in C) \wedge (B \in C) \ \Rightarrow\ (\{A,B\} \subseteq C)$	L7
L2	5	4	$(A \in C) \ \Rightarrow\ (\{A,B\} \subseteq C) \vee (B \in \overline{C})$	
L3	7	5	$(B \in D) \wedge (A \in C) \ \Rightarrow\ (\{\{A,A\},\{A,\{B,B\}\}\} \in (C \times D))$	L6
L4	3	6	$(A \in C) \wedge (A \in B) \ \Rightarrow\ (A \in (B \cap C))$	L7
L5	1	8	$(A \in B) \ \Rightarrow\ (A \in \mathcal{U})$	
L6	6	7	$(B \in D) \wedge (A \in C) \ \Rightarrow\ (\langle A,B\rangle \in (C \times D))$	L7
L7	2	8	$(B \in C) \ \Rightarrow\ (A \in \mathcal{U}) \vee (\{B,A\} \subseteq C)$	L13 L5
L8	1	32	$\emptyset = \{A,B\} \vee (A \in \mathcal{U}) \vee (B \in \mathcal{U})$	
L9	1	8	$(A \in \mathcal{U}) \ \Rightarrow\ (A \in \{A,B\})$	
L10	2	3	$\emptyset = \{A,A\} \vee (A \in \mathcal{U})$	L8
L11	1	6	$(B \subseteq A) \wedge (A \subseteq B) \ \Rightarrow\ A = B$	
L12	1	7	$(A \in \mathcal{U}) \ \Rightarrow\ (A \in \{B,A\})$	
L13	1	31	$(A \in C) \ \Rightarrow\ (\{A,B\} \subseteq C) \vee (B \in \{A,B\})$	
L14	1	5	$(A \in \{C,B\}) \ \Rightarrow\ A = B \vee A = C$	
L15	1	2	$(A \subseteq \mathcal{U})$	
L16	5	4	$(A \subseteq \{B,C\}) \wedge (C \in A) \wedge (B \in A) \ \Rightarrow\ A = \{B,C\}$	
L17	2	4	$(A \in \mathcal{U}) \wedge A = B \ \Rightarrow\ (A \in \{B,C\})$	L9
L18	1	6	$(\emptyset \subseteq A)$	

Table 4: Lemmas found for the TPTP set theory sample with the proof cost measured by the number of clauses processed by the prover.

no.	level	$\|\|\cdot\|\|$	formula	lemmas
L1	1	37	$(A \in \{C,B\}) \Rightarrow A = B \vee A = C$	
L2	2	64	$\emptyset = \{A,B\} \vee (A \in \mathcal{U}) \vee (B \in \mathcal{U})$	L1
L3	2	43	$(A \in C) \wedge (B \in C) \Rightarrow (\{A,B\} \subseteq C)$	L1
L4	1	24	$(A \in \mathcal{U}) \Rightarrow (A \in \{B,A\})$	
L5	1	24	$(A \in \mathcal{U}) \Rightarrow (A \in \{A,B\})$	
L6	3	28	$\emptyset = \{A,A\} \vee (A \in \{A,A\})$	L2 L4
L7	3	44	$(A \in \mathcal{U}) \Rightarrow (A \in \overline{B}) \vee (A \in B)$	L2
L8	1	34	$(A \in (B \cap C)) \wedge (A \in \overline{C}) \Rightarrow \mathit{false}$	
L9	1	23	$\{A\} = \{A,A\}$	
L10	1	21	$(A \in B) \Rightarrow (A \in \mathcal{U})$	
L11	1	11	$(A \in \emptyset) \Rightarrow \mathit{false}$	
L12	1	18	$(\{A,B\} \in \mathcal{U})$	
L13	2	75	$\langle A,B \rangle = \{\{A\},\{A,\{B\}\}\}$	L9
L14	1	31	$((A \cap B) \subseteq B)$	
L15	2	42	$(A \in \mathcal{U}) \vee (\{A,A\} \subseteq B)$	L1 L10

Table 5: Lemmas found for the TPTP set theory sample with the proof cost measured by the proof size.

no.	level	$\|\|\cdot\|\|$	formula	lemmas
L1	2	19	$\emptyset = \{A,B\} \vee (A \in \mathcal{U}) \vee (B \in \mathcal{U})$	L2
L2	1	15	$(A \in \{C,B\}) \Rightarrow A = B \vee A = C$	
L3	3	16	$(A \in \mathcal{U}) \Rightarrow (A \in \overline{B}) \vee (A \in B)$	L1
L4	4	20	$(A \in B) \Rightarrow (\{A,A\} \subseteq B)$	L2
L5	1	8	$\{A\} = \{A,A\}$	
L6	1	11	$(A \in \mathcal{U}) \Rightarrow (A \in \{A,B\})$	
L7	1	11	$(A \in \mathcal{U}) \Rightarrow (A \in \{B,A\})$	
L8	1	10	$(A \in \emptyset) \Rightarrow \mathit{false}$	
L9	1	16	$(B \subseteq A) \wedge (A \subseteq B) \Rightarrow A = B$	
L10	2	10	$\langle A,B \rangle = \{\{A\},\{A,\{B\}\}\}$	L5
L11	1	9	$(\{A,B\} \in \mathcal{U})$	
L12	1	15	$(A \in C) \wedge (A \in B) \Rightarrow (A \in (B \cap C))$	
L13	1	16	$(A \in B) \Rightarrow (A \in \mathcal{U})$	

Table 6: Lemmas found for the TPTP set theory sample with the proof cost measured by the proof length.

proof measure	initial cost	final cost
number of processed clauses	3991	1921 (48%)
proof size	3487	2794 (80%)
proof length	1086	878 (81%)

Table 7: Results for the TPTP set theory sample.

6.2 Mizar problems – Boolean properties of sets

These theorems address basic boolean properties of sets in the Mizar database for mathematics. The conjectures were converted from the Mizar language by Josef Urban [Urb04, Urb03] into a form suitable for automated theorem provers.

The system was rather effective for reducing the number of processed clauses required to prove these set of conjectures. Table 8 shows the summary of achieved results. For each proof measure the initial and the final cost of the proofs is shown.

proof measure	initial cost	final cost
number of processed clauses	62706	1852 (3.0%)
proof size	10680	9351 (88%)
proof length	3687	3046 (83%)

Table 8: Results for the Mizar boolean properties of sets.

As the list of the conjectures and the lemmas in this case is rather long, we did not include it in this paper.

6.3 Meredith's axiomatization of propositional logic

In this section we give detailed results for the example in section 4. The formulas are presented using standard logic symbols.

Table 9 shows the conjectures along with the results obtained when running the system with the proofs being measured by the number of processed clauses and Table 10 shows the lemmas that were found.

no.	level	$\|\|\cdot\|\|_i$	$\|\|\cdot\|\|_f$	formula	lemmas
C1	8	15	2	$\vdash (A \to A)$	L27
C2	11	3682	2	$\vdash (A \to \neg\neg A)$	
C3	7	3297	2	$\vdash (\neg\neg A \to A)$	
C4	2	17	2	$\vdash (A \to (B \to A))$	
C5	9	5731	2	$\vdash (\neg A \to (A \to B))$	
C6	8	41	5	$\neg(A \to A) \vdash B$	L27
C7	5	31	3	$A \vdash (\neg A \to B)$	
C8	3	13	3	$B \vdash (\neg A \to B)$	L18
C9	11	1673	16	$\neg(A \to \neg B) \vdash B$	L22 L8 L4
C10	7	938	62	$\neg(A \to \neg B) \vdash A$	L17 L9 L16 L30 L18 L4
C11	12	287	20	$(P \to \neg(Q \to \neg R)), P \vdash R$	L21 L30 L8 L4

Table 9: Conjectures from the Meredith's axiomatization example with the proof cost measured by the number of clauses processed by the prover.

Because we code propositional formulas into terms of predicate logic, we can express much more than just that a propositional formula is a theorem. We can also express meta-theorems that speak about provability of different formulas and what are the relations between them. For example, recall how that modus ponens rule was coded as $(t(A) \wedge t(i(A,B))) \Rightarrow t(B)$. The system derived many lemmas of similar nature, thus discovering many admissible rules. This fact becomes much more interesting in the case of modal logic, described in the next section, where the deduction theorem does not hold, hence the admissible rules have much greater importance.

no.	level	$\|\| \cdot \|\|$	formula	lemmas
L1	6	7	$C \vdash (A \rightarrow (\neg\neg B \rightarrow B))$	L14 L4
L2	6	24	$D \vdash (A \rightarrow (B \rightarrow (\neg\neg C \rightarrow C)))$	L12 L4 L3
L3	3	8	$(((D \rightarrow B) \rightarrow (E \rightarrow B)) \rightarrow (B \rightarrow C)) \vdash (A \rightarrow (B \rightarrow C))$	L23 L4
L4	1	4	$A, (A \rightarrow B) \vdash B$	
L5	3	5	$A \vdash ((A \rightarrow B) \rightarrow (C \rightarrow B))$	L18 L12
L6	8	6	$((((B \rightarrow C) \rightarrow (\neg D \rightarrow \neg A)) \rightarrow D) \rightarrow B) \vdash (A \rightarrow B)$	L27 L12
L7	7	4	$\vdash ((((\neg A \rightarrow \neg B) \rightarrow A) \rightarrow C) \rightarrow (B \rightarrow C))$	L12
L8	8	6	$C \vdash (A \rightarrow ((\neg B \rightarrow \neg A) \rightarrow B))$	
L9	5	28	$((C \rightarrow E) \rightarrow (\neg B \rightarrow \neg D)) \vdash (((A \rightarrow B) \rightarrow C) \rightarrow (D \rightarrow C))$	L18 L4 L19
L10	8	7	$(((\neg C \rightarrow \neg A) \rightarrow C) \rightarrow B) \vdash (A \rightarrow B)$	L7 L4
L11	8	4	$\vdash (((A \rightarrow B) \rightarrow \neg(\neg B \rightarrow \neg C)) \rightarrow (C \rightarrow \neg(\neg B \rightarrow \neg C)))$	L7 L12
L12	1	6	$((((B \rightarrow D) \rightarrow (\neg E \rightarrow \neg C)) \rightarrow E) \rightarrow A) \vdash ((A \rightarrow B) \rightarrow (C \rightarrow B))$	
L13	6	13	$(((B \rightarrow C) \rightarrow D) \rightarrow (\neg C \rightarrow \neg A)) \vdash (A \rightarrow (B \rightarrow C))$	L9 L12 L4
L14	5	4	$\vdash (A \rightarrow (B \rightarrow (\neg\neg C \rightarrow C)))$	L2
L15	10	4	$B \vdash (A \rightarrow \neg\neg A)$	
L16	2	7	$D, C \vdash (A \rightarrow (B \rightarrow C))$	L18
L17	5	7	$((C \rightarrow C) \rightarrow B) \vdash (A \rightarrow B)$	L31
L18	2	6	$B \vdash (A \rightarrow B)$	
L19	4	7	$C, A \vdash (\neg A \rightarrow B)$	
L20	6	6	$\vdash (((A \rightarrow (B \rightarrow B)) \rightarrow C) \rightarrow (D \rightarrow C))$	L17 L31 L12
L21	11	4	$B \vdash (A \rightarrow \neg\neg A)$	L15
L22	6	14	$((C \rightarrow D) \rightarrow B), (\neg D \rightarrow \neg A) \vdash (A \rightarrow B)$	L9 L18 L4
L23	2	4	$\vdash ((((A \rightarrow B) \rightarrow (C \rightarrow B)) \rightarrow (B \rightarrow D)) \rightarrow (E \rightarrow (B \rightarrow D)))$	L12
L24	6	4	$\vdash (((A \rightarrow \neg\neg B) \rightarrow C) \rightarrow (B \rightarrow C))$	L12
L25	3	4	$\vdash (((A \rightarrow (\neg B \rightarrow C)) \rightarrow D) \rightarrow (B \rightarrow D))$	L23 L12
L26	3	11	$(D \rightarrow C), D \vdash (A \rightarrow (B \rightarrow C))$	L18 L4
L27	7	4	$\vdash (A \rightarrow A)$	L32
L28	1	6	$C \vdash (A \rightarrow (B \rightarrow A))$	
L29	7	3	$C \vdash (A \rightarrow (B \rightarrow B))$	
L30	6	5	$(\neg B \rightarrow \neg D) \vdash (((A \rightarrow B) \rightarrow C) \rightarrow (D \rightarrow C))$	L9 L18
L31	4	4	$\vdash (((A \rightarrow A) \rightarrow B) \rightarrow (C \rightarrow B))$	L25 L12
L32	6	4	$\vdash (A \rightarrow (B \rightarrow (C \rightarrow C)))$	L17 L31
L33	9	2	$\vdash (((A \rightarrow B) \rightarrow \neg\neg C) \rightarrow (C \rightarrow \neg\neg C))$	L9 L13 L11

Table 10: Lemmas found for the Meredith's axiomatization with the proof cost measured by the number of clauses processed by the prover.

proof measure	initial cost	final cost	
number of processed clauses	15725	510	(3.2%)
proof size	1299	1024	(79%)
proof length	314	277	(88%)

Table 11: Results for the Meredith's axiomatization of propositional logic.

Table 11 shows the summary of achieved results. Again, for each proof measure the initial and the final cost of the proofs is shown.

6.4 S5 modal logic

S5 modal logic uses meta-theorems (mentioned above), since the theorem of deduction doesn't hold in S5. From this point of view S5 is an interesting example.

This set of conjectures is similar to the previous example. We have used [Hal05] to construct an axiomatization for S5 modal logic with the three Hilbert's axioms for propositional logic, axioms K, T and 5 and modus ponens and necessitation rule. We have used the same formula coding with an additional unary function symbol $l(\ldots)$ for the modal operator $\Box$.

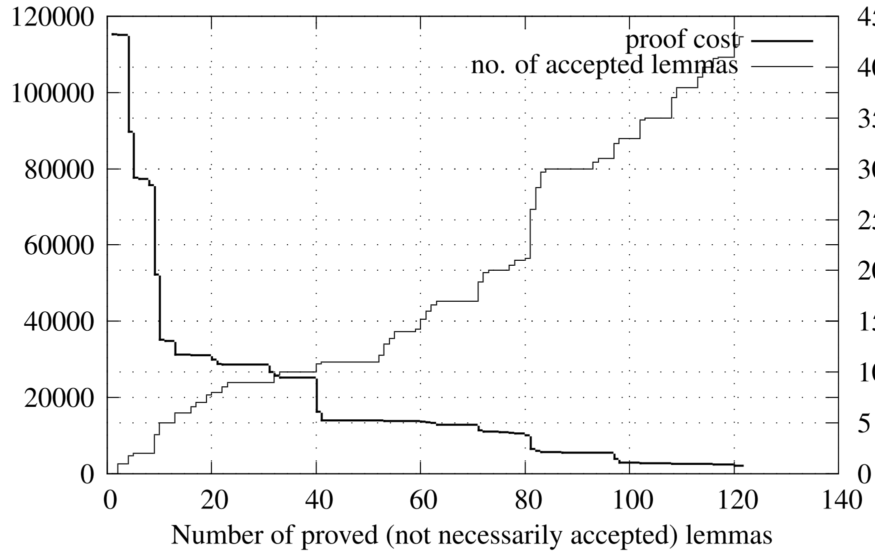

Figure 1: Run of the system on the S5 axiomatization with with the proof cost measured by the number of clauses processed by the prover.

For this example we also show a graph that illustrates performance of the system, see Figure 1. The x axis shows time points distinguished by the total number of lemmas (both accepted and unaccepted) the system has used at least in one proof. The thick line shows how the total cost of the proofs evolved and corresponds to the tick marks on the left. The thin line shows the number of lemmas that were marked as accepted and corresponds to the tick marks on the right. As we can see, the cost of the proofs was reduced to about 1/3 with the first 10 lemmas. The cost of the proofs then gradually decreased and the lemmas that were accepted often brought only a slight gain. There were two more significant improvements at the points 40, 81 and 97, when interesting lemmas were discovered and sudden advancements were made.

Most of the the lemmas that were discovered say that a particular proposition is a theorem of S5. But the system also discovered several lemmas that describe admissible rules of S5. For example the lemma L40 in Table 15 states that from $\Box B$ and $B \rightarrow A$ we can derive $\Box A$.

Table 16 shows the summary of achieved results. For each proof measure the initial and the final cost of the proofs is shown.

no.	level	$\|\|\cdot\|\|_i$	$\|\|\cdot\|\|_f$	formula	lemmas
C1	6	11	2	$\vdash (A \to A)$	L19
C2	1	2	2	$\vdash (\Box P \to P)$	
C3	9	1086	2	$\vdash (A \to \neg\neg A)$	
C4	2	5	3	$\vdash A \lor \neg\Box A$	L43
C5	1	3	3	$\vdash \neg A \lor \Box A$	
C6	9	1089	2	$\vdash (\neg\neg A \to A)$	
C7	7	3586	2	$\vdash ((\neg A \to A) \to A)$	
C8	7	44	2	$\vdash (\neg A \to (A \to B))$	L25
C9	8	680	15	$\vdash \Box(A \to \neg\Box\neg A)$	L37 L2 L36 L33
C10	5	15332	72	$\vdash (P \to \Box\neg\Box\neg P)$	L17 L2 L1
C11	6	43	5	$\neg(A \to A) \vdash B$	L19
C12	8	58	3	$A \vdash (\neg A \to B)$	
C13	2	5	3	$B \vdash (\neg A \to B)$	L10
C14	6	90	10	$\neg(A \to \neg B) \vdash B$	
C15	7	1293	6	$\neg(A \to \neg B) \vdash A$	L20 L23
C16	8	30413	2	$\vdash ((A \to B) \to (\neg B \to \neg A))$	
C17	9	23254	3	$\vdash (A \to (\neg B \to \neg(A \to B)))$	L19
C18	10	9580	4	$A, B \vdash \neg(A \to \neg B)$	L14
C19	1	3	3	$\vdash \Box(\neg\Box\neg P \to \Box\neg\Box\neg P)$	
C20	7	5471	171	$\neg(B \to \neg A) \vdash \neg(A \to \neg B)$	L28 L2 L13 L22 L1 L42 L9 L24
C21	1	8	6	$\Box(A \to B), \Box(B \to A) \vdash \Box(\Box A \to \Box B)$	
C22	10	23271	26	$\neg(A \to \neg\neg(B \to \neg C)) \vdash \neg(\neg(A \to \neg B) \to \neg C)$	L20 L23 L14

Table 12: Conjectures in the S5 axiomatization, prover processed clauses count measure

no.	level	$\|\|\cdot\|\|$	formula	lemmas
L1	2	59	$(A \to C), (A \to (C \to B)) \vdash (A \to B)$	L4
L2	3	40	$(A \to C), (C \to B) \vdash (A \to B)$	L1
L3	3	40	$C, (A \to (C \to B)) \vdash (A \to B)$	L1
L4	1	39	$\Box\Box B, (B \to A) \vdash \Box A$	
L5	5	25	$\vdash (\neg\neg A \to A)$	L3
L6	6	44	$(\neg B \to A) \vdash (\neg A \to B)$	
L7	4	59	$(A \to (\neg C \to \neg B)) \vdash (A \to (B \to C))$	L2
L8	1	33	$(\neg A \to \neg B), B \vdash A$	

Table 13: Lemmas found for the S5 axiomatization with the proof cost measured by the proof size.

no.	level	$\|\|\cdot\|\|$	formula	lemmas
L1	1	13	$(A \to C), (A \to (C \to B)) \vdash (A \to B)$	
L2	2	12	$(A \to C), (C \to B) \vdash (A \to B)$	
L3	2	13	$C, (A \to (C \to B)) \vdash (A \to B)$	L1
L4	4	10	$(\neg B \to A) \vdash (\neg A \to B)$	L7
L5	3	8	$\vdash (\neg\neg A \to A)$	
L6	1	13	$\Box\Box B, (B \to A) \vdash \Box A$	
L7	1	10	$(\neg B \to \neg A) \vdash (A \to B)$	

Table 14: Lemmas found for the S5 axiomatization with the proof cost measured by the proof length.

no.	level	$\|\|\cdot\|\|$	formula	lemmas
L1	3	8	$(A \to C), (A \to (C \to B)) \vdash (A \to B)$	L32
L2	4	11	$(A \to C), (C \to B) \vdash (A \to B)$	
L3	1	9	$(B \to (C \to D)) \vdash (A \to ((B \to C) \to (B \to D)))$	
L4	6	8	$C, (C \to B) \vdash (A \to B)$	L24
L5	2	11	$\Box\Box B, (B \to A) \vdash \Box\Box A$	L43
L6	8	5	$A \vdash (\neg\neg(A \to B) \to B)$	L13
L7	7	247	$\vdash (\neg A \to ((B \to A) \to \neg B))$	L8 L25 L1 L15
L8	5	30	$((C \to A) \to ((C \to B) \to D)) \vdash ((A \to B) \to ((C \to A) \to D))$	L3 L31 L1
L9	5	11	$(A \to (B \to D)), (B \to (D \to C)) \vdash (A \to (B \to C))$	L2 L32
L10	1	6	$B \vdash (A \to B)$	
L11	3	21	$((C \to A) \to B) \vdash (A \to B)$	L10 L32 L15
L12	6	10	$D, (D \to C) \vdash (A \to (B \to C))$	L10
L13	4	8	$(A \to (C \to B)), C \vdash (A \to B)$	L10 L1
L14	9	6	$B, A \vdash \neg(A \to \neg B)$	L20 L6
L15	1	4	$B, (B \to A) \vdash A$	
L16	1	5	$A, B \vdash \Box\Box A$	
L17	4	5	$(B \to (A \to C)) \vdash (A \to (B \to C))$	L11 L32
L18	6	5	$((A \to B) \to A) \vdash ((A \to B) \to B)$	L19 L1
L19	5	4	$\vdash (A \to A)$	
L20	2	7	$(\neg A \to \neg B), B \vdash A$	L37 L15
L21	6	7	$((C \to C) \to B) \vdash (A \to B)$	L19
L22	5	9	$(A \to (C \to B)), ((C \to B) \to C) \vdash (A \to B)$	L2 L1
L23	6	11	$\neg(A \to C) \vdash (\neg A \to B)$	L29 L24
L24	5	10	$(A \to C), \neg C \vdash (A \to B)$	L28 L2
L25	6	4	$\vdash (\neg A \to (A \to B))$	L29
L26	6	32	$(C \to (A \to D)) \vdash (A \to (B \to (C \to D)))$	L10 L9
L27	2	7	$\Box B, (B \to A) \vdash A$	L43
L28	1	4	$\neg A \vdash (A \to B)$	
L29	5	6	$(A \to (\neg C \to \neg B)) \vdash (A \to (B \to C))$	L2
L30	6	5	$(\neg A \to (B \to \neg C)) \vdash ((\neg A \to B) \to (C \to A))$	L29 L32
L31	4	4	$((A \to (B \to C)) \to (((A \to B) \to (A \to C)) \to D)) \vdash ((A \to (B \to C)) \to D)$	L1
L32	2	7	$(A \to (B \to C)) \vdash ((A \to B) \to (A \to C))$	L15
L33	7	4	$\vdash (\neg\neg A \to (B \to A))$	L29 L25
L34	8	4	$\vdash (A \to (\neg\neg B \to B))$	L17 L33
L35	1	9	$(\neg C \to \neg B) \vdash (A \to (B \to C))$	
L36	4	4	$(A \to ((C \to A) \to B)) \vdash (A \to B)$	L1
L37	1	6	$(\neg B \to \neg A) \vdash (A \to B)$	
L38	8	6	$(A \to (\neg B \to (C \to B))) \vdash (A \to (\neg B \to \neg C))$	L7 L9
L39	7	6	$(\neg A \to A) \vdash (\neg A \to B)$	L25 L1
L40	2	9	$\Box B, (B \to A) \vdash \Box A$	L43
L41	8	6	$\vdash (A \to (B \to \neg\neg B))$	L17 L29 L33
L42	6	11	$(A \to D), (D \to C) \vdash (A \to (B \to C))$	L2 L24
L43	1	5	$\Box A \vdash A$	

Table 15: Lemmas found for the S5 axiomatization with the proof cost measured by the number of clauses processed by the prover.

proof measure	initial cost	final cost
number of processed clauses	115327	2091 (1.8%)
proof size	632	379 (60%)
proof length	642	389 (61%)

Table 16: Results for S5 modal logic.

The system again performed very well in the case when the measure was the number of processed clauses. This time, the total cost of the proofs was reduced to less than 2%.

7 Future work

As the system is particularly efficient in speeding up the prover, we believe that it could be modified to search for lemmas that would make it possible to prove conjectures that the prover alone wasn't able to prove. This will require a change of strategy, because currently the system looks primarily for lemmas that improve already existing proofs of the conjectures and therefore are not general enough to prove some new unknown conjecture.

We would also like to investigate the nature of the lemmas that help to improve particular proof measures in order to develop a better strategy for their evaluation.

Finally, we plan to perform a in-depth testing of the system on various sets of conjectures from different sources.

8 Conclusion

Given a related set of conjectures, it is possible to automatically construct lemmas that can significantly reduce the cost of the proofs of the conjectures. The results are summarized in Table 17 for convenience.

	processed clauses		proof size		proof length	
set of conjectures	$\|\|\cdot\|\|_i$	$\|\|\cdot\|\|_f$	$\|\|\cdot\|\|_i$	$\|\|\cdot\|\|_f$	$\|\|\cdot\|\|_i$	$\|\|\cdot\|\|_f$
TPTP set theory	3991	1921 (48%)	3487	2794 (80%)	1086	878 (81%)
Mizar set properties	62706	1852 (3.0%)	10680	9351 (88%)	3687	3046 (83%)
Meredith's axiomatization	15725	510 (3.2%)	1299	1024 (79%)	314	277 (88%)
S5 modal logic	115327	2091 (1.8%)	632	379 (60%)	642	389 (61%)

Table 17: Summary of the results of the system on the presented sets of conjectures.

The system that we have developed performs well on different sets of conjectures, particularly if the cost of the proofs of the conjectures is measured in the number of clauses processed by the prover. The system can also improve the size and/or the length of the proofs, although it is not as effective in these cases.

References

[AS92] Owen L. Astrachan and Mark E. Stickel. Caching and lemmaizing in model elimination theorem provers. In Deepak Kapur, editor, *CADE*, volume 607 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 1992.

[Col02a] Simon Colton. *Automated Theory Formation in Pure Mathematics*. Distinguished Dissertations. Springer, 2002.

[Col02b] Simon Colton. The HR program for theorem generation. In Andrei Voronkov, editor, *CADE*, volume 2392 of *Lecture Notes in Computer Science*, pages 285–289. Springer, 2002.

[CP80] P. T. Cox and T. Pietrzykowski. *A complete, nonredundant algorithm for reversed skolemization*, volume 87 of *Lecture Notes in Computer Science*. Springer, May 1980.

[CP93] Ritu Chadha and David Plaisted. *Finding logical consequences using unskolemization*, volume 689 of *Lecture Notes in Computer Science*. Springer, May 1993.

[FFF99] Marc Fuchs, Dirk Fuchs, and Matthias Fuchs. Generating lemmas for tableau-based proof search using genetic programming. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1027–1032, Orlando, Florida, USA, July 1999. Morgan Kaufmann.

[Hal05] John Halleck. Logic systems, 2005. http://www.cc.utah.edu/˜nahaj/logic/structures/.

[McC94] W. W. McCune. OTTER 3.0 reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, Argonne, Illinois, 1994.

[MVF$^+$02] William McCune, Robert Veroff, Branden Fitelson, Kenneth Harris, Andrew Feist, and Larry Wos. Short single axioms for boolean algebra. *J. Autom. Reasoning*, 29(1):1–16, 2002.

[Qua92] Art Quaife. *Automated Development of Fundamental Mathematical Theories*. Kluwer Academic Publishers, 1992.

[Sch02] S. Schulz. E – A brainiac theorem prover. *Journal of AI Communications*, 15(2-3):111–126, 2002.

[SGC03] G. Sutcliffe, Y. Gao, and S. Colton. A Grand Challenge of Theorem Discovery. In J. Gow, T. Walsh, S. Colton, and V. Sorge, editors, *Proceedings of the Workshop on Challenges and Novel Applications for Automated Reasoning, 19th International Conference on Automated Reasoning*, pages 1–11, 2003.

[SS98] G. Sutcliffe and C. B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[Urb03] Josef Urban. Translating Mizar for first order theorem provers. In *MKM*, volume 2594 of *Lecture Notes in Computer Science*, pages 203–215. Springer, 2003.

[Urb04] Josef Urban. MPTP - motivation, implementation, first experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.

[WP03] Larry Wos and Gail W. Pieper. *Automated Reasoning and the Discovery of Missing and Elegant Proofs*. Rinton Press, 2003.