Sledgehammer: Judgement Day

Sascha Böhme* and Tobias Nipkow

Institut für Informatik, Technische Universität München

Abstract. Sledgehammer, a component of the interactive theorem prover Isabelle, finds proofs in higher-order logic by calling the automated provers for first-order logic E, SPASS and Vampire. This paper is the largest and most detailed empirical evaluation of such a link to date. Our test data consists of 1240 proof goals arising in 7 diverse Isabelle theories, thus representing typical Isabelle proof obligations. We measure the effectiveness of Sledgehammer and many other parameters such as run time and complexity of proofs. A facility for minimizing the number of facts needed to prove a goal is presented and analyzed.

1 Introduction

Sledgehammer (**SH**) [4, 5, 7], developed by Paulson *et al.* at Cambridge, is a linkup of the interactive theorem prover (**ITP**) Isabelle/HOL [6] (for higher-order logic) and automated first-order provers (**ATP**s). The purpose of this paper is to evaluate SH in various dimensions, but in particular w.r.t. effectiveness: How much benefit can the average Isabelle user expect from SH? *What's in it for Joe Proof?*

In addition to evaluating effectiveness, this paper also presents data for determining weaknesses of SH, improving SH and developing it further. Some of our data have already influenced the SH setup.

Based on a large sample of typical Isabelle proofs, the following aspects of SH are analyzed: success rates; complications when turning ATP proofs into Isabelle proofs; run times; size and difficulty of proofs. We also describe and analyze a new addition to SH for reducing the number of facts used in proofs.

We believe that our analysis is of interest to both the ATP and ITP community. Many of the issues faced by SH will be faced by any linkup with ATPs where a) there is a non-trivial translation from some logic H into the language of the ATPs and b) proofs delivered by the ATPs are translated back and checked as proofs in H. Invariably, such back-and-forth translations introduce problems and lead to lost proofs, one of the issues we investigate in detail.

For a management summary of our findings, please fast-forward to §8.

See `http://www.in.tum.de/~nipkow/pubs/ijcar10.html` for our test data and log files. Many of these problems have been included into the TPTP library (see `http://www.tptp.org`).

* Research supported by BMBF project *Verisoft XT*

1.1 Related work

We do not give a comprehensive overview of combinations of interactive and automatic theorem provers. Relevant related work is already discussed in publications by Paulson *et al.* The main focus of our paper is empirical evaluation. Meng and Paulson present statistics in their reports on the development of SH, too, but those were based on hand-picked problems primarily meant for tuning the setup. This paper is an independent evaluation with a large set of test data coming from a wider collection of theories (= modules for definitions and proofs) in the Isabelle distribution, with no hand selection: all goals arising in the Isabelle proofs are passed to the ATPs.

Another large scale empirical evaluation of automating ITP proofs with ATPs is reported for the Mizar system by Urban [13].

On the empirical side, the CASC [11] has been setting standards for many years now and we refer to it throughout the text. Apart from differences in what data is analyzed, the main difference to CASC is attitude or perspective:

> Although we look at success rates and other data of individual ATPs, SH is not a competition among ATPs but between ATPs and Isabelle: how much can ATPs do automatically where Isabelle requires nontrivial human interaction?

2 Sledgehammer

When SH is activated by an Isabelle user to prove some goal G, it performs the following steps:

1. Based on the syntactic form of G, a set S of relevant **facts** (axioms and lemmas) is extracted from the background theory, that is, all the knowledge available to the user at this point. S typically contains hundreds of facts.
2. $S \cup \{\neg G\}$ is translated from HOL (with a Haskell-like type system) into untyped first-order logic (FOL).
3. One or more ATPs are called.
4. If one of the ATPs returns a proof of G, i.e. a refutation of $\neg G$, SH extracts the facts $R \subseteq S$ used in the proof.

Strictly speaking, at this point SH is over, but we regard the next step as part of SH: The user is given the option to call the internal ATP *Metis* (**M**) that will try to prove G with the help of R. M was designed and implemented by Hurd [3] and adapted for Isabelle by Paulson and Susanto [7]. M is written in SML and produces actual Isabelle/HOL proofs, the whole point of the setup. This double-checking is necessary because the translation from HOL to FOL is potentially unsound (see §4.1). M is slow compared to the leading ATPs, but its performance at CASC is respectable. Typically R contains only a few facts, in rare cases more than 20. As a result, the final step in the SH process, calling M with R, succeeds surprisingly often — more below. Successful M calls become part of the Isabelle proof text. We call the M step **proof reconstruction** because M has to search

for a proof once more, but w.r.t. a very focussed set of facts. Essentially, SH uses the ATPs as very powerful relevance filters for M.

A recent enhancement of SH (implemented by Fabian Immler at TUM) is the ability to call ATPs remotely via Sutcliffe's *SystemOnTPTP* [10].

3 The setup

Our test data are 7 theories from the Isabelle distribution and the Archive of Formal Proofs (`afp.sf.net`) representative for a range of typical applications. All theories were developed without the use of SH.

Arrow	Arrow's impossibility theorem	8%	LS
NS	Needham-Schroeder shared-key protocol	8%	I
Hoare	Completeness of Hoare logic with procedures	16%	IL
Jinja	Type soundness of a subset of Java	13%	IL
SN	SN of typed λ-calculus with de Bruijn indices	9%	AI
FTA	Fundamental Theorem of Algebra	34%	A
FFT	Fast Fourier Transform	12%	A L

The leftmost column shows the names used below, the % column what percentage of the overall 1240 problems/goals come from each theory, and the rightmost column the logical nature of each theory: A means arithmetic; I means that the theory is dominated by recursive functions defined by equations and inductive relations defined by Horn clauses; L means that λs occur; S means sets.

Our data differs from that of Meng and Paulson in two respects: they consider 153 [4] and 285 [5] goals as opposed to our 1240, and their goals are hand-selected for the purpose of tuning SH, where extremely easy or extremely hard problems are unhelpful. In contrast, all goals arising in our 7 theories are part of our test data, including those proved by induction, of which there are 72. Thus we claim our data is representative for the average goals actually faced by Isabelle users.

We have run all experiments with the 3 ATPs SH currently supports, with their default SH options: E [9] (version 1.0 in auto mode), SPASS [14] (version 3.5 with SOS enabled), and Vampire [8] (version 9.0 in CASC mode). SOS, *set of support*, is a complete resolution strategy [16] but incomplete in the SPASS context. We confirmed that SPASS with SOS worked best for us.

Below we abbreviate the provers as **E**, **S** and **V**.

We conducted our tests on Dual Core Intel Xeon processors running at 3.06 GHz. The ATPs were run with different timeouts, and "timeout" refers to ATPs by default. In contrast, M's timeout was fixed at 30s (for the tests — normally M can run as long as the user likes). The reason for the 30s: During interactive proof development, proof text does not evolve linearly one step after the next, but whole regions are continuously modified (by the user) and rechecked (by the machine). In our experience, 30s is at the limit of what users are prepared to tolerate for rechecking (as opposed to finding) of a proof step.

4 Success rates

Below we present the success rates (in percent) both for the ATPs and for M proof reconstruction runs — we refer to them as ATP-success and M-success. Remember that M-success is what counts for the Isabelle user, because only those proofs can be imported into Isabelle. We have run E, S and V on all 7 sample theories, with ATP timeouts of 5, 10, 30, 60 and 120 seconds. In the following table we show the data obtained for 5s and 120s. The rightmost column, labeled with ∅, gives the average. The table contains for each prover, timeout and theory combination two values, the ATP and M success rates in percent. Both success rates are relative to the total number of problems (in each theory). Hence M-success is never above ATP-success.

	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	∅
E 5	22 19	46 32	46 42	24 23	57 56	50 49	12 12	40 37
E 120	26 19	58 40	51 46	26 24	59 58	58 56	19 17	45 41
S 5	29 26	38 38	50 42	22 20	50 46	53 51	15 12	40 37
S 120	30 27	41 41	51 42	22 20	50 47	55 52	15 12	42 38
V 5	18 16	22 22	35 34	26 24	49 47	49 48	10 10	35 33
V 120	35 29	52 46	57 47	29 26	61 58	62 58	18 14	49 44
ESV 5	33 28	56 44	53 48	28 26	61 58	61 58	17 15	48 44
ESV 120	42 34	65 56	61 53	31 27	63 61	67 63	22 18	54 48

Percentages of problems solved

For example, E with a timeout of 120s can solve 26% of all goals in theory Arrow, but after running M on the solved goals (with the facts identified by the ATP), only 19% of all goals could actually be proved (because M failed to reconstruct $(26-19)/26 \approx 27\%$ of the E proofs). The two bottom rows of the table represent the ATP **ESV**, which runs E, S and V in parallel, and each of them is run until it finds a proof or reaches timeout.

For a subset of theories, the data is shown graphically for all timeouts in Figures 1 to 4, where the two success rates are labeled by P and PM, where $P \in \{\mathsf{E}, \mathsf{S}, \mathsf{V}\}$. The M-success rate of ESV is shown as the grey area. Individual ATP-success rates may lie above the grey area. Note that all graphs in this paper use a logarithmic timeline.

The splendid news is that on average

> Running each of the 3 provers for 5s yields the same M-success rate (44%) as running the most effective one for 120s.

This result is crucial for interactive proofs, where every second counts. The gain of 3 ATPs over 1 is impressive: the success rates rise between 4 and 13 percentage points, depending on the timeout (5s or 120s) and the ATP we compare ESV with. Having hard empirical evidence for the effectiveness of ESV has influenced Isabelle's SH setup: ESV is now the default setting. More precisely V is invoked *remotely* (see §2) because a) V is not readily available (for example not on MacOS) and b) it requires only a dual core machine and internet access to

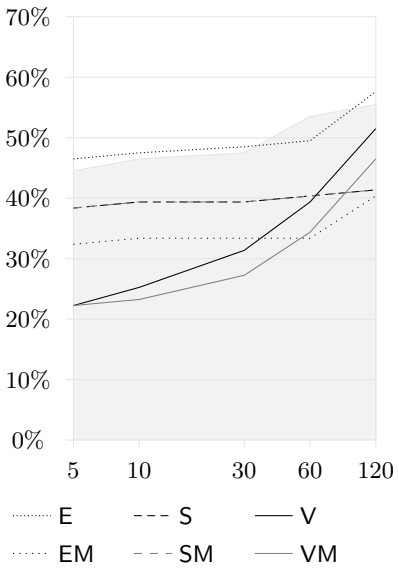


Fig. 1. Theory NS

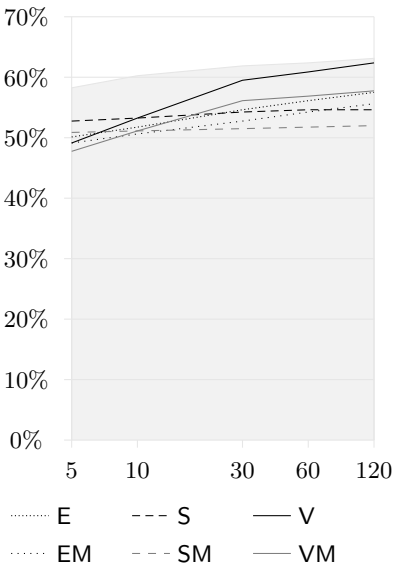


Fig. 2. Theory FTA

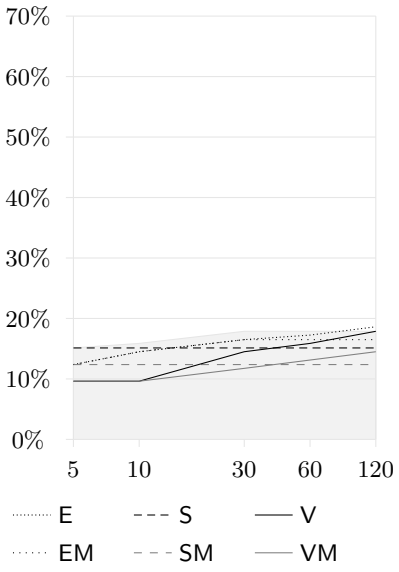


Fig. 3. Theory FFT

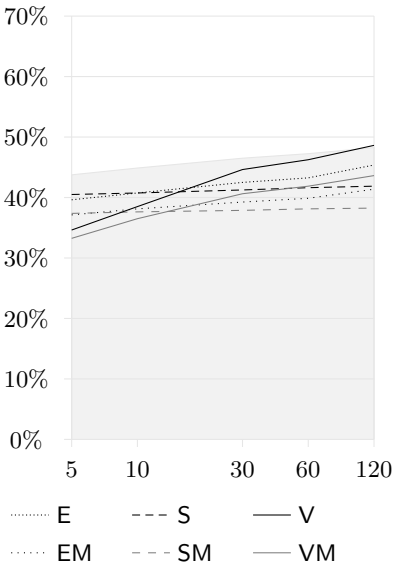
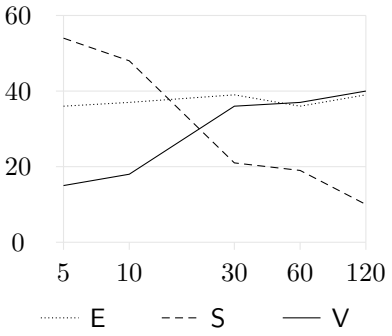


Fig. 4. Average

benefit from this setup without the provers stealing each others' cycles — thanks to Sutcliffe's SystemOnTPTP.

The Figures 1 to 4 bring out the difference between the provers' performance profiles very well. S starts out (at 5s) above E and V but does not improve much, whereas E and V keep on growing and overtake S at some point. The reason for V's behaviour is *strategy scheduling*: V runs multiple strategies in sequence until one finds a proof. Thus V is able to utilize long timeouts more effectively. Neither E nor S employ strategy scheduling. It is to E's credit that its success rate keeps growing almost as well as V's.

Success rates are one thing, indispensability is another. Indispensability of a prover can be judged by the number of goals only it can prove, its *uniqueness number* (an intuitive variant of Sutcliffe's more refined *SOTAC* [11]). In the diagram to the right we show the uniqueness numbers (cumulative over all theories) of our ATPs with varying timeout. Clearly, S is indispensable for users with no patience, V is indispensable for users with a lot of time, and nobody should be without E.



Further important observations concerning success rates are:

- The difference between the ATP and M success rates increases for E and V over time. This is not surprising because M's timeout is fixed. The problem is discussed in detail in the next subsection.
- In theory NS (Fig. 1), E tops all other provers, but EM is 10% below E. This is an exception and turns out to be the result of a typing problem (see §4.1).
- The theory with the lowest success rate is FFT. We conjecture (this is difficult to ascertain) that the culprit is λ: there are many goals that contain the summation operator $\sum$ which gives rise to a λ (internally). We looked at the goals that were proved and only one contained a $\sum$.
- Our average success rates are lower than Meng and Paulson's [4]. We believe that this is due to the fact that they hand-selected their goals whereas we picked entire theories with very variable success rates.

Readers disappointed by the actual ATP success rates should keep in mind that although all Isabelle goals are provable, not all ATP problems are: 72 problems require induction, and an unknown and hard to determine number is unprovable because the extraction of relevant facts from the Isabelle theory (step 1 of SH) does not guarantee to preserve provability.

We have also run the ATPs for 240s and observed the success rates: E/S/V prove an additional 9/0/10 goals, M proves an additional 5/0/5 goals — the success rates increase by a mere 0.4/0/0.4 percent. Hence we stopped at 120s.

We will now examine the problem that M may fail to reconstruct ATP proofs.

4.1 M may fail

On average, the difference between ATP and M success rates is at most 5% (at 120s). This means that at 120s ATP timeout, M fails to reconstruct around 10% of all ATP proofs. The situation is extreme in theory NS, where M fails on 30% of the proofs found by E. There are three reasons why M may fail to reconstruct a proof:

1. The default translation of HOL problems to ATP input is unsound [4]. The reason is that HOL has a Haskell-like type system, which needs to be encoded into unsorted FOL, and by default SH economizes on the amount of type information that is passed to the ATPs for the sake of performance. In some cases this allows the ATPs to find proofs that are no longer sound when translated back to HOL. We call those **type-unsound** below.
2. M is internally a two stage process: the first stage is an ATP that produces proof objects, the second stage translates these proof objects into Isabelle proofs. The first stage is called with the same reduced type information passed to the external ATPs and may also find type-unsound proofs. In such cases the translation to Isabelle proofs throws a type exception.
3. M may time out.

It is hard to distinguish these 3 reasons. For example, if M throws a type exception, the proof found by the external ATP may be type-unsound, and M merely followed suit, or the external ATP may have found a perfectly good proof, but M's first stage found a type-unsound one.

For a timeout of 120s we examined the failed M calls individually and classified them according to the above three-fold distinction. The classification cannot be automatic, as we just explained. We used methods introduced below (e.g. adding full type information) but in some cases it remains approximate. Hence the figures below should be taken with a grain of salt. On average, 66% of failed M calls are genuine M timeouts, 21% are caused by type-unsound ATP proofs and 13% are type-unsound M proofs. But the individual provers differ notably in their profile of failed M calls:

120	Total # of M failures	M timeout	ATP type-unsound	M type-unsound
E	47	34%	51%	15%
S	42	79%	9%	12%
V	56	84%	5%	11%

This confirms V's ability to find difficult proofs, given enough time. And E appears to be the expert at exploiting (type-)unsound axiomatizations.

Of course it is not surprising that M cannot compete with highly optimized ATPs, the CASC already told us that. Hence the SH architecture is often viewed with suspicion by ATP researchers. In fact, we consider M in the SH context surprisingly effective. Of course, we would like to reconstruct all sound ATP proofs. There are two ways to improve the situation:

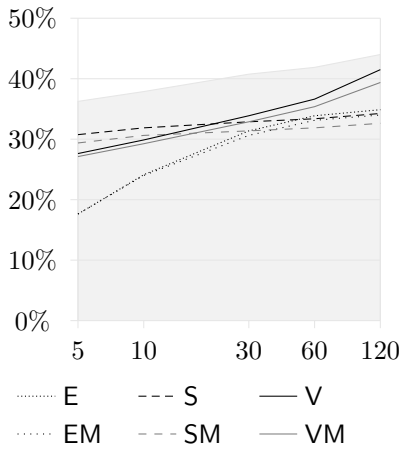
– *Give M more time.* Experimentally we doubled M timeout from 30s (see §3) to 60s. This reduced the number of M timeouts by 20%, but 60s is the limit of what is acceptable in an interactive system.
– *Replay the proof found by the ATP rather than reconstruct it.* Paulson [7] presents an extension of SH with a translation scheme from resolution proofs found by an ATP (and output in TSTP format [12]) into Isabelle proof scripts that replay the resolution proof step by step. At the time where that paper was written only E produced TSTP proofs, and the resulting Isabelle proof scripts were a bit brittle, i.e. they would sometimes fail because of technical problems. Last but not least, it is very unattractive to have long resolution proofs in the middle of nicely structured and readable proofs that are customary in Isabelle [15]. For these reasons the default setup for SH is to call M, and translation of TSTP proofs is currently not used at all, although it is clearly the way to go — but see the Conclusions.

When running ESV, M failures are greatly reduced: there are 121 problems where M fails to reconstruct at least one of the ATP proofs, but for 46 of them M succeeds to reconstruct a proof found by a different ATP.

We will now consider how type-unsound ATP and M proofs can be avoided.

4.2 The fully-typed translation

To avoid type-unsound ATP proofs, SH also offers a fully-typed translation from HOL to FOL (**FT** below). The figure to the right shows the resulting average success rates. Compared with Fig. 4, we observe a decrease of the success rate of around 10% by going to FT. Meng and Paulson [4] measured 10–20%, but even 10% is not acceptable. Hence FT remains only an option. It can be useful in case an ATP appears to have found a type-unsound proof. Switching to FT can sometimes allow the ATP to find a valid proof. In most cases, however, the ATP will time out instead, which leaves one none the wiser. The purpose of FT, to avoid M failures, is largely fulfilled: the M-success rates are barely below those for the ATPs. They are not identical, because M may still genuinely time out (or find type-unsound proofs, see §4.3). With increasing ATP timeout, the gap between V and VM is widening because V starts to outperform M again (whose timeout is fixed, see §3).



4.3 Metis with full types

M is run by default with the same reduced type information as the ATPs. Thus M also finds type-unsound proofs, which are rejected by Isabelle. Therefore Paulson

recently added a version of M with full type information, $\mathsf{M_{FT}}$ below. $\mathsf{M_{FT}}$ cannot replace M just like the fully-typed ATP translation cannot replace the default one, because of performance reasons. Instead we have evaluated how often $\mathsf{M_{FT}}$ would reconstruct a proof where M failed. That is, how much benefit we can expect from $\mathsf{M_{FT}}$ in addition to M. All figures below were obtained for 120s ATP timeout and refer to all failures over all theories and provers. The figures for M are given in §4.1 above.

Of the 18 valid ATP proofs where M runs into typing problems, only 6 can be reconstructed by $\mathsf{M_{FT}}$. In most of the other cases $\mathsf{M_{FT}}$ times out (because the type information overwhelms it), but in a few cases $\mathsf{M_{FT}}$ itself runs into typing problems (a somewhat unfortunate situation). Interestingly, $\mathsf{M_{FT}}$ also manages to reconstruct 5 (of 96) proofs where M had timed out. Taken together, $\mathsf{M_{FT}}$ succeeds on 9% of the M failures. These figures are cumulative over all 3 ATPs.

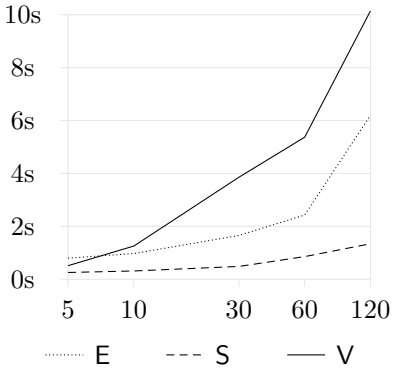
When running ESV, type unsoundness of M almost ceases to be an issue: only two such failed proofs remain. $\mathsf{M_{FT}}$ fails on both of them, too.

5 Time

SH performs the following steps: extracting and translating relevant facts, running the ATPs, and running M. Extraction and translation take 3.2s on average.

Below we show the run times for successful ATP runs in seconds. We exclude the failed runs because they generally end in timeout, thus distorting the statistics. The table on the left is restricted to 30s, 60s and 120s. The average over all theories is shown graphically on the right.

	Arrow	NS	Hoare	Jinja	SN	FTA	FFT
E 30	3.6	2.1	1.0	1.0	0.7	1.9	3.7
E 60	5.0	2.5	1.0	1.0	1.4	3.0	5.0
E 120	5.2	15.3	8.4	1.0	2.4	4.8	10.0
S 30	0.4	0.9	0.4	0.1	0.3	0.6	0.6
S 60	0.4	1.9	0.9	0.1	0.3	1.0	0.6
S 120	0.4	3.9	1.9	2.5	0.3	0.9	0.6
V 30	6.9	5.8	5.7	1.9	2.4	2.9	7.0
V 60	7.8	15.7	7.0	1.9	2.4	3.8	10.0
V 120	18.3	35.0	10.5	2.0	2.5	6.5	20.5



- We can see once again the effect of V's strategy scheduling. S is the most economical with its time.
- E and V agree on what are hard theories and what are easy ones.

For a theoretical analysis of $T(t)$, the total time for successful ATP runs with timeout t, observe in Fig. 4 that the average success rate is roughly linear w.r.t. a logarithmic t. This means that when the timeout increases from t to $2t$, a fixed number k of new goals are proved, on average in time $1.5t$: $T(2t) = T(t) + 1.5kt$. The master theorem for recurrence relations tells us that $T(t)$ is linear; the

solution is $T(t) = 1.5kt + c_0$, for some c_0. Since the t axis is logarithmic, the expected graph for $T(t)$ would be an exponential function. Our data supports this merely vaguely.

We have also measured the average time spent by the ATPs on failed proof attempts. Since E and V try so hard, this figure is close to timeout. In some of the theories, S departs from this pattern and its average failure time is 30% below timeout. This could be induced by SOS's incompleteness (see §3).

The average run times for successful runs of M turn out to be moderate, that is, below 1 second, even at 120s ATP timeout:

120	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	∅
E	0.0	0.5	0.3	0.5	0.0	0.0	0.2	0.2
S	0.3	0.1	0.2	0.6	0.0	0.1	0.5	0.2
V	0.3	0.1	0.4	0.3	0.0	0.1	0.1	0.2

This is perfectly acceptable for interactive use.

6 Proof complexity

How difficult are the proofs found by the ATPs? We will look at it both from the ATP's and the user's point of view. From the ATP's point of view, the time taken to find a proof is one measure already covered. Another one is the number of facts used in the proof (i.e. the cardinality of R, see §2). This is a very crude measure as one can easily have long and difficult to find proofs with only a few facts. But we will observe a strong correlation between fact and time complexity. Below we show the average number of facts for the ATP timeouts of 5s and 120s.

For each prover and theory we show a pair $i\ j$ where i is the average number of facts returned from ATP proofs and j the average number of facts in (successful!) M proofs. We have $i \geq j$ in most cases because M tends to fail on proofs involving many rather than a few facts. All figures are rounded.

5	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	∅σ
E	3 2	6 5	2 2	3 2	2 2	3 3	5 5	3 3 3
S	2 2	3 3	2 2	2 1	1 1	2 2	6 7	2 2 3
V	2 3	3 3	2 2	2 2	2 2	3 3	5 5	3 3 3
120	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	∅σ
E	9 2	6 5	3 2	3 2	2 2	4 4	6 5	4 3 4
S	2 2	3 3	3 2	2 1	1 1	3 2	6 7	3 2 3
V	3 3	6 5	4 3	3 2	2 2	4 4	7 6	4 3 4

The rightmost column contains triples $i\ j\ s$ where i and j are the averages and s is the standard deviation of the j's. We observe the following:

– The average number of facts in M proofs is the same at 5s and 120s, namely 2–3. The standard deviation σ of 3–4, however, indicates that there is quite a bit of variation.

– Raising the timeout from 5s to 120s, the average number of facts in ATP proofs rises from 2–3 to 3–4, just 1 above the fact complexity of M proofs. This confirms the expectation that M failures lose the more difficult proofs, which are of course the ones users would most like to see automated.
– There is a strong correlation between the average fact complexity and run times of ATP proofs: proofs in theories SN and Jinja are particularly short and fast, in theories FFT and NS proofs are particularly long and slow.

Just like success rates, when the timeout is increased from 5s to 120s, fact complexity of proofs increases only slowly. However, the high standard deviation tells another story, which is confirmed when we look at the *maximum* fact complexity. In the table below, pairs i j are the maximum fact complexities of all ATP and all M proofs in a given theory, and *max* is the maximum of the maxima.

5	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	*max*
E	7 7	15 13	12 8	10 5	6 6	15 15	12 12	15 15
S	5 5	6 6	11 10	10 6	8 3	25 25	21 21	25 25
V	7 7	8 8	9 9	7 7	8 8	23 23	10 10	23 23
120	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	*max*
E	47 7	15 15	18 14	10 5	7 7	33 33	37 12	47 33
S	5 5	7 7	24 10	10 6	8 5	28 25	21 21	28 25
V	10 7	22 15	52 35	8 7	8 8	44 44	21 18	52 44

From the *max* column we can tell that at 5s, M keeps up with the ATPs, but that at 120s, E and V outperform M significantly. The rest of the table merely illustrates the considerable variation depending on theories and provers.

We now look at proof complexity from a user perspective. For a user, a proof is **trivial** if it consists of the invocation of one of the proof methods `simp`, `auto`, `arith` or `blast`. They are the standard proof tools familiar to all Isabelle users. Their functionality is irrelevant here. The point is that automating such proofs is of little help to Isabelle users, because they are already automatic. Or at least almost so, because the form of the goal will almost always narrow the choice down to one or two of the methods. Note that we no longer consider `simp` and friends trivial if they need to be augmented somehow, e.g. by supplying additional facts, because in that case the user had to figure out what those facts are. It turns out that the percentage of trivial proofs among those found by the ATPs does not vary much with timeout and is around 64% at 30s timeout. In contrast, only 53% of all proofs in the considered theories are trivial. Clearly, ATPs have a predilection for trivial goals, i.e. goals with a trivial proof. *A priori*, this is not at all evident because triviality is in the eye of the beholder! For example, `simp` can generate long chains of conditional rewrites and `arith` complicated proofs in linear arithmetic. But this means that the success rates we measured so far are skewed. What the user really wants to know is this:

How many non-trivial goals can the ATPs prove for me?

Let G be the set of goals, $T \subseteq G$ the trivial ones (for Isabelle) and $A \subseteq G$ the automatically provable ones (by an ATP). Let $t = |T|/|G|$, $t_a = |A \cap T|/|A|$ and

$s = |A|/|G|$ (the success rate). Then the *non-trivial success rate* $|A-T|/|G-T|$ turns out to be $s(1-t_a)/(1-t)$ by a simple calculation. In our setting $t = 0.53$ and $t_a = 0.64$ on average (see the two tables above). Thus we need to adjust the success rate by 0.77 to obtain the non-trivial success rate. Since s is around 45% (M-success, see Fig. 4), it means that in fact only around 34% of all non-trivial goals are proved by SH.

Finally, we focus on the textually most complex Isabelle proofs, the compound ones. Those are subproofs consisting of a begin–end (`proof`–`qed` in Isabelle parlance) block. They are the last resort for most users, if everything else fails. There are 49 compound proofs in our theories (excluding inductions), of which E/S/V solved 8/8/3 with 5s and 9/8/9 with 120s timeout, roughly 17%, which is not bad.

7 Minimization

A set of facts returned by an ATP as a proof of some goal is **redundant** iff some proper subset of the facts proves the goal. Many proofs found by ATPs are redundant, for two very different reasons: certain facts are genuinely useless and can lead to unnecessary detours (if used), but other facts enable shortcuts and their removal forces a detour. Since genuinely useless facts can lead to M timeouts, we have investigated **minimization**, the process of reducing a given set of facts to an irredundant subset that still proves the goal. This is a new extension of SH that we developed at TUM. The implementation is due to Philipp Meyer.

7.1 Minimization algorithms

The obvious linear algorithm removes one fact after another from the initial set, calls the ATP with the reduced fact set, and puts the fact back if that proof fails now. This requires as many calls of the ATP as there are facts in the initial set.[1] But there is also a clever algorithm based on bisecting the set [1, §4.3]. It can take as little as $\log_2 n$ calls of the ATP (if only 1 fact is needed) and as much as $2n$ (if all are needed). The beauty and lure of the binary algorithm was such that we implemented it right away. It was only after using it for some time that we suspected that it performed worse than the linear one on our data. After measuring the number of iterations of the binary algorithm, it turned out we were right: on average it required 1.15 times as many iterations as the linear one. This agrees with a simulation by Jasmin Blanchette of the binary algorithm on random data with the same redundancy rate as our data (about 1/3, see §7.2). Of course we switched to the linear algorithm as a result. The simulation predicts superiority of the binary algorithm above 40% redundancy.

[1] Since the initial set is quite small (see §6), a default ATP timeout of 5s during minimization suffices most of the time and leads to very acceptable run times, in particular because minimization only needs to be performed once.

7.2 Benefits

The following table shows how many additional goals could be proved due to minimization: each entry is the difference between the number of M failures before and after minimization. Negative numbers indicate a loss of proofs.

120	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	$\sum$
E	3	1	4	1	0	−2	0	7
S	2	0	6	1	0	1	2	12
V	5	2	4	1	0	−3	1	10

Most of the time we gain proofs, but in FTA we lose a few. Losses are of no consequence: minimization is optional, and if M succeeds on the original set of facts but fails on the minimized set, the user would simply stick with the original set and no harm is done. Hence the negative numbers should be disregarded. In a nutshell: 9–13 (out of about 50, see the table in §4.1) M failures can be avoided by minimization. In FTA the net effect is negative because FTA is very algebraic: there is a large redundant set of facts that often allow short proofs; the removal of some such derived shortcut may just push M over the edge.

How redundant are proofs found by the ATPs? How much does minimization reduce the set of facts used? Depending on the theory, 10–50% of the facts can be dropped (30% on average). This does not vary much with ATP timeout, but increases for large proofs. For example, the record 52-fact proof found by V in Hoare (see §6) collapses to 3 facts after minimization.

What is the impact of minimization on M run times? We have measured by how much the run time changes on average, i.e. the average of all a_i/b_i where a_i (b_i) is the time M takes after (before) minimization of proof i, provided both runs succeed. However, run times for the same call of M easily fluctuate by 10ms. In particular, if a_i or b_i is below 10ms, a_i/b_i becomes somewhat random. Hence we have replaced a_i/b_i by 1 if $a_i - b_i \leq 10$ms. The following table contains the averages of all a_i/b_i in each theory and the weighted average of the averages:

120	Arrow	NS	Hoare	Jinja	SN	FTA	FFT	∅
E	0.9	0.9	1.2	1.0	1.0	0.9	0.7	1.0
S	1.1	1.0	0.9	1.0	1.0	0.9	1.0	1.0
V	0.8	0.7	0.9	1.0	1.0	1.0	0.8	0.9

Minimization can cut both ways but has only a small effect on average.

Finally we look at the impact of minimization on $\mathsf{M_{FT}}$ (see §4.3). It turns out that because proofs are often simplified by minimization, M fails less often and $\mathsf{M_{FT}}$ is less in demand. If $\mathsf{M_{FT}}$ is called where M failed (with minimized sets of facts), a mere additional 4 proofs over all theories and provers is obtained. This is down from 11 additional proofs without minimization (§4.3)

8 Conclusions

With respect to our realistic test data we have established the following:

– Success rates for Sledgehammer are around 45% but vary enormously from theory to theory (from below 20% up to 60%).
– The more meaningful rate of how many non-trivial goals (by Isabelle standards) are solved by the ATPs is around 34%. *ATPs can help ITPs!*
– Running all 3 ATPs together for 5s yields the same success rate (44%) as running the most effective one for 120s. Therefore Sledgehammer now calls all 3 ATPs concurrently.
– SPASS is indispensable for short timeouts, Vampire for long ones, and E in any situation, according to the number of goals proved only by that ATP.
– CASC results for the FOF and CNF divisions (Vampire just ahead of E) correctly predict ATP success on Isabelle theories, although CASC test data is not dominated by Isabelle problems: at CASC-22, none of the FOF problems and 59 of the 200 CNF problems came from Isabelle theories.
– Proof reconstruction in Isabelle using Metis works well most of the time but loses up to 10% of sound ATP proofs, mainly because Metis times out.
– Minimization of ATP proofs reduces the required number of facts by 1/3 (and in an extreme case from 52 down to 3 facts), thus helping 20% of the failed Metis proofs to succeed. Our measurements showed that the naive algorithm was faster than the clever binary one we had implemented first.

For these reasons we plan the following future work items:

– In order to avoid loosing up to 10% of ATP proofs because Metis fails, we intend to reactivate proof replay [7] while ironing out the implementation and presentation problems. We aim at producing truly readable (natural deduction) proofs along the lines of Huang [2].
– We plan an "auto Sledgehammer" mode for Isabelle were each goal is automatically passed to all 3 ATPs with a low timeout like 5s. The low timeout avoids the current effect of users going into sleep mode and waiting for the default 60s timeout of all ATPs before they reactivate their brain.
– Our test harness will also be used for further tuning of Sledgehammer (the default filter parameters determined by Meng and Paulson [4]) and Metis (whose parameters have never been tuned for Isabelle). It will also help in continuous performance monitoring by gathering key figures like success rates in our daily regression tests.

Acknowledgement Mike Gordon and Larry Paulson generously hosted a visit by Tobias Nipkow to the Cambridge Computer Lab where much of this research was conducted. Geoff Sutcliffe answered numerous questions. Jasmin Blanchette, Alex Krauss and Geoff Sutcliffe helped to improve the paper considerably.

References

1. A. Bradley and Z. Manna. Property-directed incremental invariant generation. *Formal Asp. Comput.*, 20:379–405, 2008.

2. X. Huang. Reconstructing proofs at the assertion level. In A. Bundy, editor, *Automated Deduction, CADE-12*, volume 814 of *Lect. Notes in Comp. Sci.*, pages 738–752. Springer-Verlag, 1994.
3. J. Hurd. First-order proof tactics in higher-order logic theorem provers. In M. Archer, B. Di Vito, and C. Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, 2003.
4. J. Meng and L. C. Paulson. Translating higher-order clauses to first-order clauses. *J. Automated Reasoning*, 40:35–60, 2008.
5. J. Meng and L. C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *J. Applied Logic*, 7:41–57, 2009.
6. T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 2002. `http://www.in.tum.de/~nipkow/LNCS2283/`.
7. L. C. Paulson and K. W. Susanto. Source-level proof reconstruction for interactive theorem proving. In K. Schneider and J. Brandt, editors, *Theorem Proving in Higher Order Logics, TPHOLs 2007*, volume 4732 of *Lect. Notes in Comp. Sci.*, pages 232–245. Springer-Verlag, 2007.
8. A. Riazanov and A. Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15:91–110, 2002.
9. S. Schulz. E - a brainiac theorem prover. *AI Commun.*, 15(2-3):111–126, 2002.
10. G. Sutcliffe. SystemOnTPTP. In D. McAllester, editor, *Automated Deduction, CADE-17*, volume 1831 of *Lect. Notes in Comp. Sci.*, pages 406–410. Springer-Verlag, 2000.
11. G. Sutcliffe. The 4th IJCAR Automated Theorem Proving System Competition — CASC-J4. *AI Commun.*, 22:59–72, 2009.
12. G. Sutcliffe, J. Zimmer, and S. Schulz. TSTP Data-Exchange Formats for Automated Theorem Proving Tools. In V. Sorge and W. Zhang, editors, *Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems*, pages 201–215. IOS Press, 2004.
13. J. Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Automated Reasoning*, 37(1-2):21–43, 2006.
14. C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski. Spass version 3.5. In R. A. Schmidt, editor, *Automated Deduction, CADE-22*, Lect. Notes in Comp. Sci., pages 140–145. Springer-Verlag, 2009.
15. M. Wenzel. Isar — a generic interpretative approach to readable formal proof documents. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors, *Theorem Proving in Higher Order Logics, TPHOLs'99*, volume 1690 of *Lect. Notes in Comp. Sci.*, pages 167–183. Springer-Verlag, 1999.
16. L. Wos, G. Robinson, and D. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *J. ACM*, 12:536–541, 1965.