




Sledgehammering without ATPs

Short Paper

Martin Desharnais   

Ludwig-Maximilians-Universität München, Germany

Jasmin Blanchette   

Ludwig-Maximilians-Universität München, Germany

Abstract

We describe an alternative architecture for “hammers,” inspired by Magnushammer, in which proofs are found by the proof assistant’s built-in automation instead of by external automatic theorem provers (ATPs). We implemented this approach in Isabelle’s Sledgehammer and evaluated it. The new ATP-free approach nicely complements the traditional Sledgehammer. The two approaches in combination solve more goals than the traditional ATP-based approach alone.

2012 ACM Subject Classification Computing methodologies → Theorem proving algorithms

Keywords and phrases Interactive theorem proving, proof assistants, proof automation

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material *Dataset:* <https://doi.org/10.5281/zenodo.15727890>

Funding *Martin Desharnais:* Co-funded by the European Union (ERC, Nekoka, 101083038). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Jasmin Blanchette: Same as for Desharnais.

Acknowledgments We thank Benedikt Pilger, whose preliminary experiments motivated us to investigate an ATP-free hammer further. We also thank Mark Summerfield and the anonymous reviewers for suggesting textual improvements.

1 Introduction

Sledgehammer [3,18] is a tool for Isabelle/HOL [17] that solves proof goals automatically using external automatic theorem provers (ATPs). “Hammer” systems such as Sledgehammer but also HOL(y)Hammer [12] and CoqHammer [9] are made up of four or five main components [6]:

1. The *relevance filter* (or “premise selector”) heuristically identifies a subset of the available facts (definitions, lemmas, theorems, etc.) as possibly relevant to the current goal. Typically, hundreds of facts can be selected without overwhelming the ATP. Sledgehammer includes three relevance filters: MePo is based on iterative selection [15], MaSh is based on naive Bayes and k nearest neighbors [5], and MeSh combines MePo and MaSh.
2. The *translation module* constructs an ATP problem from the selected facts and the current goal, converting them from the proof assistant’s logic (e.g., Isabelle’s polymorphic higher-order logic) to the ATP’s logic.
3. The ATP tries to prove the problem. For Sledgehammer, commonly used ATPs include the first-order superposition prover SPASS [7]; the higher-order superposition provers E [21], Vampire [2], and Zipperposition [20]; the higher-order paramodulation prover Leo-III [19]; and the first-order SMT solvers cvc5 [1], veriT [8], and Z3 [10]. Multiple ATPs are tried in parallel.



© Martin Desharnais and Jasmin Blanchette;
licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:8

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

4. On success, the optional *proof minimization module* repeatedly invokes the ATP with subsets of the facts referenced in the ATP proof, in an attempt to reduce the number of dependencies and speed up the next step, proof reconstruction.
5. Finally, the *proof reconstruction module* transforms a proof found by an ATP into the proof assistant’s input language (e.g., Isabelle’s Isar [22]). Typically, the structure of the ATP proof is discarded, and a single Isabelle proof method (also called tactic) is invoked with the facts referenced in the ATP proof.

As an example, Sledgehammer might select 1024 facts and translate them, along with the goal, to E’s monomorphic higher-order logic. Then suppose E finds a proof involving three facts: f_1 , f_2 , and f_3 . Minimization reduces this list to two: f_1 and f_3 . Finally, by trial and error, Sledgehammer determines that the proof method `simp add: f_1 f_3` solves the Isabelle goal, so this is suggested to the user. The user can simply click this proof method invocation to insert it into their proof development, and move on to the next goal.

The ATPs’ strength is that they can find their way through hundreds of facts. Nevertheless, it is not uncommon for Sledgehammer to fail on a goal that can be solved by a single parameterless call to `auto`, Isabelle’s main workhorse. The ATPs, which routinely surprise users with nonobvious proofs, can also be a bottleneck.

This suggests an alternative, ATP-free architecture for hammers:

1. The relevance filter works as before, except that it selects at most a few dozen facts, because proof methods tend to scale less well than ATPs.
2. A proof method is invoked. The selected facts f_1, \dots, f_n are either introduced into the goal as premises (e.g., `using $f_1 \dots f_n$ by auto`) or are made available to the proof method via parameters (e.g., `by (simp add: $f_1 \dots f_n$)`), depending on the method.
3. On success, the proof minimization module attempts to reduce the number of facts needed for the proof to speed up the proof method.

We implemented this approach in Sledgehammer, with proof methods complementing the ATPs for proof search. The idea is not completely new. Magnushammer [16], whose relevance filter relies on powerful transformers, has a similar architecture and obtained remarkable results. However, the tool is not readily available, and it requires considerably heavier computational resources than Sledgehammer’s faster and simpler relevance filters.

With our evaluation (Section 4), we try to answer the following research questions: *How successful is an ATP-free hammer that relies on fast and simple relevance filtering? How fast are proof methods? How successful are ATP and ATP-free hammers when used in combination?* But before we try to answer these questions, we review the implementation in Sledgehammer (Section 2) and study an example interaction with the newly extended tool (Section 3).

2 Implementation

Sledgehammer has a generic notion of “prover” with two instances: traditional ATPs [14] and SMT solvers [3]. We have now extended the tool with a third instance: proof methods. The supported proof methods are those that Sledgehammer already used for reconstruction and that do not rely on external tools: `algebra`, `argo`, `auto`, `blast`, `fastforce`, `force`, `linarith`, `meson`, `metis`, `order`, `presburger`, `satx`, and `simp`. Notably, `simp` performs simplification using equations as oriented rewrite rules, `auto` extends `simp` with general-purpose classical reasoning, and `blast` is a tableau prover [4].

Thus, our ATP-free hammer is not a brand new hammer; rather, it is integrated in an existing hammer that is already part of the users' workflow. Our objective is to increase the users' productivity without requiring them to change their habits.

This tool will be part of the next Isabelle release, which is expected at the end of 2025. We used the results of our empirical evaluation to craft a new default portfolio that runs traditional ATPs, SMT solvers, and proof methods in parallel.

3 Example

The `Khovanskii_Theorem` entry of the *Archive of Formal Proofs* formalizes a theorem in additive combinatorics due to Askold Khovanskii [13]. Among the proof steps is the following:

```
have map2 (+) (map2 (+) xs ys) zs = map2 (+) xs (map2 (+) ys zs)
  for xs ys zs :: ('a :: ab_semigroup_add) list
```

In Isabelle, the **have** command states a new goal within a local proof context, and **for** specifies universally quantified variables. The goal states that the operation `map2 (+)` is associative for any lists `xs`, `ys`, and `zs` of elements that form an additive abelian semigroup. The function `map2 :: ('a ⇒ 'b ⇒ 'c) ⇒ 'a list ⇒ 'b list ⇒ 'c list` maps a binary function on the respective elements of two lists. The binary operation `+` is from the additive abelian semigroup structure.

The first thing an Isabelle user might do to prove this goal is to invoke the **try0** command to launch several standard proof methods in parallel. This leads to a timeout with no proof found. Next, the user might try Sledgehammer. When invoked as an ATP hammer, Sledgehammer also times out with no proof found. When invoked as an ATP-free hammer, Sledgehammer finds and outputs the following proof:

```
by (simp add: case_prodI2 prod.case_distrib zip_assoc case_prod_app
    map_zip_map map_zip_map2 ab_semigroup_add_class.add_ac(1))
```

4 Evaluation

For our evaluation, we used Mirabelle [11, Section 4] to evaluate the ATP and ATP-free hammers on goals originating from different sessions of the Isabelle distribution and the *Archive of Formal Proofs*. Our objective was to include goals from various areas of mathematics and computer science representing diverse formalization styles. Within a session, the goals were selected at regular intervals, alleviating the issue that consecutive goals tend to be similar. We used the repository revisions 0b46bf0a434f of Isabelle (2025-02-25) and 9dac8e411570 of the *Archive of Formal Proofs* (2025-02-28). The evaluation was run on a server with an Intel Xeon Silver 4114 CPU and 187 GiB of RAM. The raw evaluation data is available online.¹

We first ran a representative external prover with a timeout of 10 seconds, 32 facts, and varying relevance filter configurations on 1000 goals (four sessions with 250 goals each). The aim was to identify the relevance filter that works best for a small number of facts. We found out that with MePo, MaSh, and MeSh, the success rate was respectively 30.1 %, 24.4 %, and 32.4 %. Based on these results, we decided to use MeSh for the remaining experiments. We selected 5000 goals (20 sessions with 250 goals each).

¹ <https://doi.org/10.5281/zenodo.15727890>

■ **Table 1** Success rate of several proof methods. The last column gives the success rate of the union of all goals solved when considering 0, 1, 2, 4, 8, 16, 32, and 64 facts.

Proof method	Success rate (%) w.r.t. number of facts								Union
	0	1	2	4	8	16	32	64	
algebra	0.6	0.4	0.2	0.1	0.0	0.0	0.0	0.0	0.7
argo	4.4	4.8	5.0	5.1	5.7	6.2	6.4	6.4	6.5
auto	24.8	25.5	26.2	26.9	28.5	27.1	20.8	7.9	33.9
blast	11.1	11.8	12.4	12.9	13.5	13.7	13.3	12.4	17.4
fastforce	26.1	26.9	27.8	28.4	29.4	28.8	20.8	9.5	36.8
force	26.0	26.7	27.4	27.9	28.4	26.9	21.1	9.8	35.5
linarith	4.0	4.3	4.4	4.6	4.9	5.2	5.3	5.3	5.6
meson	5.9	6.6	7.1	7.7	9.5	10.7	10.6	8.3	14.3
metis	9.3	10.3	11.2	12.6	15.9	18.9	21.6	20.6	28.1
order	0.5	0.5	0.6	0.6	0.6	0.7	0.7	0.7	0.7
presburger	6.3	6.7	7.1	7.7	8.4	8.7	8.0	5.4	11.3
satx	2.3	2.5	2.6	2.7	2.9	3.1	3.3	3.3	3.3
simp	17.7	18.5	19.1	19.7	20.4	18.7	12.2	4.0	27.2
All of the above	28.5	29.7	30.7	32.1	35.1	38.8	39.0	32.8	46.8

Success Rate We evaluated the ATP hammer (i.e., Sledgehammer’s default portfolio) with a timeout of 30 seconds: The success rate was 72.1 %. (This is similar to the value found by Desharnais et al. [11].) In the present evaluation, we ran several proof methods with a timeout of two seconds and a number of facts varying from 0 to 64: The success rates are in Table 1. The union of all proof methods with 0 facts had a success rate of 28.5 %; this scenario approximates the **try0** command. The union of all proof methods with any number of facts (i.e., the ATP-free hammer) had a success rate of 46.8 %.

Together, the ATP hammer and our approximation of **try0** had a success rate of 74.1 %: an improvement of 2.0 percentage points over the ATP hammer alone. Together, the ATP and ATP-free hammers had a success rate of 74.6 %: an improvement of 2.5 percentage points over the ATP hammer and of 0.5 percentage points over the ATP hammer and **try0** together. A 2.5 or 0.5 percentage points increase of the success rate might not sound like much, but as Blanchette et al. remarked in a previous paper [7, Section 7]:

When analyzing enhancements to automatic provers, it is important to remember what difference a modest-looking gain of a few percentage points can make to users. The benchmarks were chosen to be representative of typical Isabelle goals and include many that are either too easy or too hard to effectively evaluate automatic provers. Indeed, some of the most essential tools in Isabelle, such the arithmetic decision procedures, score well below 10% when applied indiscriminately to the entire Judgment Day suite.

Running Time How quickly do the proof methods find the proofs? Table 2 answers this question for some proof methods, focusing on the ones that have a success rate of more than 10 % according to the last column of Table 1. The median, or 50th percentile, is shown, as well as other percentiles. For example, the number 31 in the first row indicates that 90 % of the goals proved by **auto** without any fact were proved within 31 milliseconds. For comparison, the last row presents the running times for the ATP hammer.

Table 2 shows that proof methods can find a proof very quickly, but that the running time of some proof methods (e.g., **auto**, **force**) rapidly increases with the number of facts.

■ **Table 2** Running time to find a proof of several proof methods.

Prover(s)	Facts	Running time (ms) at percentiles					
		25	50	75	90	95	99
auto	0	2	4	12	31	56	224
auto	4	5	12	29	69	136	382
auto	16	58	124	251	532	814	1 532
auto	64	716	1 088	1 478	1 718	1 854	1 981
blast	0	0	1	2	5	11	86
blast	4	1	1	3	7	14	107
blast	16	3	6	10	29	87	546
blast	64	42	85	262	624	1 078	1 789
fastforce	0	1	5	21	70	146	566
fastforce	4	5	15	42	116	234	668
fastforce	16	48	147	384	816	1 181	1 813
fastforce	64	313	986	1 421	1 810	1 887	2 066
force	0	1	5	19	63	136	906
force	4	4	13	35	113	234	918
force	16	39	103	250	551	856	1 688
force	64	563	943	1 407	1 805	1 926	2 056
meson	0	0	4	8	14	26	74
meson	4	4	8	12	22	35	174
meson	16	20	30	45	96	216	984
meson	64	98	182	352	886	1 228	1 794
metis	0	8	11	19	32	70	474
metis	4	12	17	25	47	114	486
metis	16	36	49	80	182	372	1 318
metis	64	224	404	816	1 454	1 635	1 900
simp	0	1	4	11	29	51	135
simp	4	2	6	17	41	71	225
simp	16	9	30	106	298	488	1 100
simp	64	50	204	605	1 365	1 776	1 990
ATP hammer		464	653	1 044	1 820	2 339	3 966

Portfolios So far, we have seen that a virtual portfolio consisting of all 13 proof methods with all eight number-of-fact configurations had a success rate of 46.8% (Table 1). Such a virtual portfolio is not very realistic, because it would require $13 \times 8 \times 2 = 208$ seconds with our per-configuration timeout of two seconds. A more realistic alternative is to consider the greedy sequence of length n . The sequence is obtained by iteratively (1) taking first some best prover configuration for the goals of interest, and (2) removing all goals proved by this configuration. These two steps are repeated n times, yielding n configurations. The result is not necessarily optimal, but it can be computed efficiently.

Table 3a presents the greedy sequence of length 32 based on our experiments. This portfolio could be for a ATP-free hammer that aims to maximize the success rate. Table 3b presents an alternative greedy sequence of the same length that first evaluates all proof methods without any fact. This portfolio could be for a ATP-free hammer that aims to subsume the **try0** command. Given 64 seconds—or 8 seconds and 8 threads—the first portfolio would have had a success rate of 46.7% and the second of 46.4%.

■ **Table 3** Greedy sequences of proof methods. The column SRI contains the success rate improvement (% points) over the previous row and CSR contains the cumulative success rate (%).

(a) When maximizing success rate

	Proof method	Facts	SRI	CSR
1	fastforce	8	29.40	29.4
2	metis	64	7.62	37.0
3	auto	32	2.56	39.6
4	metis	16	1.52	41.1
5	force	0	1.22	42.3
6	simp	32	1.00	43.3
7	fastforce	16	0.72	44.0
8	metis	32	0.52	44.6
9	fastforce	2	0.24	44.8
10	linarith	16	0.20	45.0
11	metis	8	0.20	45.2
12	simp	16	0.20	45.4
13	simp	64	0.18	45.6
14	blast	32	0.12	45.7
15	algebra	0	0.10	45.8
16	force	32	0.10	45.9
17	argo	16	0.08	46.0
18	auto	4	0.08	46.1
19	auto	64	0.08	46.1
20	presburger	32	0.08	46.2
21	meson	16	0.06	46.3
22	blast	8	0.04	46.3
23	blast	64	0.04	46.4
24	fastforce	32	0.04	46.4
25	force	8	0.04	46.4
26	linarith	64	0.04	46.5
27	meson	64	0.04	46.5
28	simp	1	0.04	46.6
29	simp	8	0.04	46.6
30	algebra	2	0.02	46.6
31	blast	0	0.02	46.6
32	blast	4	0.02	46.7

(b) When subsuming try0

	Proof method	Facts	SRI	CSR
1	fastforce	0	26.10	26.1
2	metis	0	1.06	27.2
3	auto	0	0.68	27.8
4	force	0	0.32	28.2
5	algebra	0	0.14	28.3
6	linarith	0	0.14	28.4
7	argo	0	0.06	28.5
8	blast	0	0.04	28.5
9	meson	0	0.00	28.5
10	order	0	0.00	28.5
11	presburger	0	0.00	28.5
12	satx	0	0.00	28.5
13	simp	0	0.00	28.5
14	metis	64	8.90	37.4
15	fastforce	16	4.06	41.5
16	metis	32	1.42	42.9
17	simp	32	1.12	44.0
18	auto	32	0.52	44.6
19	fastforce	8	0.36	44.9
20	metis	16	0.32	45.2
21	simp	16	0.20	45.4
22	simp	64	0.18	45.6
23	linarith	64	0.12	45.7
24	metis	8	0.12	45.9
25	blast	32	0.10	46.0
26	force	32	0.10	46.1
27	presburger	32	0.08	46.1
28	auto	64	0.06	46.2
29	meson	16	0.06	46.3
30	blast	8	0.04	46.3
31	blast	64	0.04	46.3
32	fastforce	4	0.04	46.4

5 Conclusion

We described a hammer architecture in which the external ATPs are replaced by the proof assistant’s built-in proof methods. We presented an extension of Sledgehammer that implements this idea. The relevance filter is then directly connected to Isabelle’s built-in proof automation. In this way, goals that, given the right facts, are easy for the built-in automation but too difficult for external ATPs can now be solved. Our empirical evaluation found that the newly extended Sledgehammer has a higher success rate than its predecessor based only on ATPs.

Future work includes the classification of facts. Currently, we treat all facts given to a proof method in the same way, but some methods (e.g., `auto`) work better when the provided facts are classified (e.g., into simplification, introduction, and elimination rules). By integrating fact classification into Sledgehammer, we hope to improve the success rate of proof methods.

References

- 1 Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. *cvc5: A versatile and industrial-strength SMT solver*. In Dana Fisman and Grigore Rosu, editors, *TACAS 2022, Part I*, volume 13243 of *LNCS*, pages 415–442. Springer, 2022. URL: https://doi.org/10.1007/978-3-030-99524-9_24.
- 2 Ahmed Bhayat and Martin Suda. A higher-order Vampire (short paper). In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *IJCAR 2024, Part I*, volume 14739 of *LNCS*, pages 75–85. Springer, 2024. URL: https://doi.org/10.1007/978-3-031-63498-7_5.
- 3 Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending Sledgehammer with SMT solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013. URL: <https://doi.org/10.1007/s10817-013-9278-5>.
- 4 Jasmin Christian Blanchette, Lukas Bulwahn, and Tobias Nipkow. Automatic proof and disproof in Isabelle/HOL. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *FroCoS 2011*, volume 6989 of *LNCS*, pages 12–27. Springer, 2011. URL: https://doi.org/10.1007/978-3-642-24364-6_2.
- 5 Jasmin Christian Blanchette, David Greenaway, Cezary Kaliszyk, Daniel Kühlwein, and Josef Urban. A learning-based fact selector for Isabelle/HOL. *Journal of Automated Reasoning*, 57(3):219–244, 2016. URL: <http://dx.doi.org/10.1007/s10817-016-9362-8>.
- 6 Jasmin Christian Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *Journal of Formalized Reasoning*, 9(1):101–148, 2016. URL: <https://doi.org/10.6092/issn.1972-5787/4593>.
- 7 Jasmin Christian Blanchette, Andrei Popescu, Daniel Wand, and Christoph Weidenbach. More SPASS with Isabelle: Superposition with hard sorts and configurable simplification. In Lennart Beringer and Amy Felty, editors, *ITP 2012*, volume 7406 of *LNCS*, pages 345–360. Springer, 2012. URL: https://doi.org/10.1007/978-3-642-32347-8_24.
- 8 Thomas Bouton, Diego Caminha Barbosa de Oliveira, David Déharbe, and Pascal Fontaine. veriT: An open, trustable and efficient SMT-solver. In Renate A. Schmidt, editor, *CADE-22*, volume 5663 of *LNCS*, pages 151–156. Springer, 2009. URL: https://doi.org/10.1007/978-3-642-02959-2_12.
- 9 Łukasz Czajka and Cezary Kaliszyk. Hammer for Coq: Automation for dependent type theory. *Journal of Automated Reasoning*, 61(1-4):423–453, 2018. URL: <https://doi.org/10.1007/s10817-018-9458-4>.
- 10 Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008. URL: https://doi.org/10.1007/978-3-540-78800-3_24.
- 11 Martin Desharnais, Petar Vukmirović, Jasmin Blanchette, and Makarius Wenzel. Seventeen provers under the hammer. In June Andronick and Leonardo de Moura, editors, *ITP 2022*, volume 237 of *LIPICs*, pages 8:1–8:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://doi.org/10.4230/LIPICs.ITP.2022.8>.
- 12 Cezary Kaliszyk and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light. *Mathematics in Computer Science*, 9(1):5–22, 2015. URL: <https://doi.org/10.1007/s11786-014-0182-0>.
- 13 Angeliki Koutsoukou-Argyraki and Lawrence C. Paulson. Khovanskii’s theorem. *Archive of Formal Proofs*, 2022. URL: https://isa-afp.org/entries/Khovanskii_Theorem.html.
- 14 Jia Meng and Lawrence C. Paulson. Translating higher-order clauses to first-order clauses. *Journal of Automated Reasoning*, 40(1):35–60, 2008. URL: <https://doi.org/10.1007/s10817-007-9085-y>.

- 15 Jia Meng and Lawrence C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *Journal of Applied Logic*, 7(1):41–57, 2009. URL: <https://doi.org/10.1016/j.jal.2007.07.004>.
- 16 Maciej Mikula, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Albert Q. Jiang, Jin Peng Zhou, Christian Szegedy, Lukasz Kucinski, Piotr Milos, and Yuhuai Wu. Magnus-hammer: A transformer-based approach to premise selection. In *ICLR 2024*. OpenReview.net, 2024. URL: <https://openreview.net/forum?id=oYjPk8mqAV>.
- 17 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. URL: <https://doi.org/10.1007/3-540-45949-9>.
- 18 Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *IWIL-2010*, volume 2 of *EPiC*, pages 1–11. EasyChair, 2012. URL: <https://doi.org/10.29007/36dt>.
- 19 Alexander Steen and Christoph Benzmüller. The higher-order prover Leo-III. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *IJCAR 2018*, volume 10900 of *LNCS*, pages 108–116. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-94205-6_8.
- 20 Petar Vukmirović, Alexander Bentkamp, Jasmin Blanchette, Simon Cruanes, Visa Nummelin, and Sophie Tourret. Making higher-order superposition work. In André Platzter and Geoff Sutcliffe, editors, *CADE-28*, volume 12699 of *LNCS*, pages 415–432. Springer, 2021. URL: <https://doi.org/10.1007/s10817-021-09613-z>.
- 21 Petar Vukmirović, Jasmin Blanchette, and Stephan Schulz. Extending a high-performance prover to higher-order logic. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *TACAS 2023, Part II*, volume 13994 of *LNCS*, pages 111–129. Springer, 2023. URL: https://doi.org/10.1007/978-3-031-30820-8_10.
- 22 Makarius Wenzel. Isabelle/Isar—a generic framework for human-readable proof documents. In Roman Matuszewski and Anna Zalewska, editors, *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*, volume 10(23) of *Studies in Logic, Grammar, and Rhetoric*. University of Białystok, 2007.