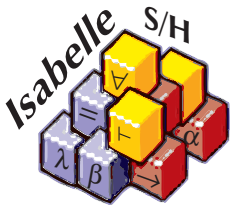

Hammering Away

A User's Guide to Sledgehammer for Isabelle/HOL

Jasmin Christian Blanchette
Institut für Informatik, Technische Universität München

January 30, 2011

Contents

1 Introduction

Sledgehammer is a tool that applies first-order automatic theorem provers (ATPs) and satisfiability-modulo-theories (SMT) solvers on the current goal. The supported ATPs are E [7], SPASS [10], Vampire [6], SInE-E [4], and SNARK [8]. The ATPs are run either locally or remotely via the SystemOnTPTP web service [9]. In addition to the ATPs, the SMT solvers Z3 [11] is used, and you can tell Sledgehammer to try Yices [3] and CVC3 [1] as well.

The problem passed to the automatic provers consists of your current goal together with a heuristic selection of hundreds of facts (theorems) from the current theory context, filtered by relevance. Because jobs are run in the background, you can continue to work on your proof by other means. Provers can be run in parallel. Any reply (which may arrive half a minute later) will appear in the Proof General response buffer.

The result of a successful proof search is some source text that usually (but not always) reconstructs the proof within Isabelle. For ATPs, the reconstructed proof relies on the general-purpose Metis prover [5], which is fully integrated into Isabelle/HOL, with explicit inferences going through the kernel. Thus its results are correct by construction.

In this manual, we will explicitly invoke the **sledgehammer** command. Sledgehammer also provides an automatic mode that can be enabled via the "Auto Sledgehammer" option from the "Isabelle" menu in Proof General. In this mode, Sledgehammer is run on every newly entered theorem. The time limit for Auto Sledgehammer and other automatic tools can be set using the "Auto Tools Time Limit" option.

To run Sledgehammer, you must make sure that the theory *Sledgehammer* is imported—this is rarely a problem in practice since it is part of *Main*. Examples of Sledgehammer use can be found in Isabelle's `src/HOL/Metis_Examples` directory. Comments and bug reports concerning Sledgehammer or this manual should be directed to `blanchette@in.tum.de`.

2 Installation

Sledgehammer is part of Isabelle, so you don't need to install it. However, it relies on third-party automatic theorem provers (ATPs) and SAT solvers. Currently, E, SPASS, and Vampire can be run locally; in addition, E, Vampire, SInE-E, and SNARK are available remotely via SystemOnTPTP [9]. If you want better performance, you should install E and SPASS locally.

There are three main ways to install ATPs on your machine:

- If you installed an official Isabelle package with everything inside, it should already include properly setup executables for E and SPASS, ready to use. [1]
- Alternatively, you can download the Isabelle-aware E and SPASS binary packages from Isabelle's download page. Extract the archives, then add a line to your `~/.isabelle/etc/components` file with the absolute path to E or SPASS. For example, if the `components` does not exist yet and you extracted SPASS to `/usr/local/spass-3.7`, create the `components` file with the single line

  ```
  /usr/local/spass-3.7
  ```

  in it.
- If you prefer to build E or SPASS yourself, or obtained a Vampire executable from somewhere (e.g., `http://www.vprover.org/`), set the environment variable `E_HOME`, `SPASS_HOME`, or `VAMPIRE_HOME` to the directory that contains the `eproof`, `SPASS`, or `vampire` executable. Sledgehammer has been tested with E 1.0 and 1.2, SPASS 3.5 and 3.7, and Vampire 1.0 [2]. Since the ATPs' output formats are neither documented nor stable, other versions of the ATPs might or might not work well with Sledgehammer.

To check whether E and SPASS are installed, follow the example in §3.

Remote ATP invocation via the SystemOnTPTP web service requires Perl with the World Wide Web Library (`libwww-perl`) installed. If you must use a proxy server to access the Internet, set the `http_proxy` environment variable to the proxy, either in the environment in which Isabelle is launched or in your `~/.isabelle/etc/settings` file. Here are a few examples:

1. Vampire's license prevents us from doing the same for this otherwise wonderful tool.
2. Following the rewrite of Vampire, the counter for version numbers was reset to 0; hence the new Vampire 1.0 is more recent than Vampire 11.5.

```
http_proxy=http://proxy.example.org
http_proxy=http://proxy.example.org:8080
http_proxy=http://joeblow:pAsSwRd@proxy.example.org
```

3 First Steps

To illustrate Sledgehammer in context, let us start a theory file and attempt to prove a simple lemma:

theory *Scratch*
imports *Main*
begin

lemma "$[a] = [b] \longleftrightarrow a = b$"
sledgehammer

Instead of issuing the **sledgehammer** command, you can also find Sledgehammer in the "Commands" submenu of the "Isabelle" menu in Proof General or press the Emacs key sequence C-c C-a C-s. Either way, Sledgehammer produces the following output after a few seconds:

Sledgehammer: "e" for subgoal 1:
$([a] = [b]) = (a = b)$
Try this command: ***by*** *(metis hd.simps).*
To minimize the number of lemmas, try this:
sledgehammer *minimize [prover = e] (hd.simps).*

Sledgehammer: "spass" for subgoal 1:
$([a] = [b]) = (a = b)$
Try this command: ***by*** *(metis insert_Nil last_ConsL).*
To minimize the number of lemmas, try this:
sledgehammer *minimize [prover = spass] (insert_Nil last_ConsL).*

Sledgehammer: "vampire" for subgoal 1:
$([a] = [b]) = (a = b)$
Try this command: ***by*** *(metis eq_commute last_snoc)*
To minimize the number of lemmas, try this:
sledgehammer *minimize [prover = vampire] (eq_commute last_snoc).*

Sledgehammer: "remote_sine_e" for subgoal 1:
$([a] = [b]) = (a = b)$
Try this command: ***by*** *(metis hd.simps)*

To minimize the number of lemmas, try this:
sledgehammer *minimize [prover = remote_sine_e] (hd.simps). Sledgehammer: "remote_z3" for subgoal 1:*
$([a] = [b]) = (a = b)$
Try this command: ***by*** *(metis hd.simps)*
To minimize the number of lemmas, try this:
sledgehammer *minimize [prover = remote_sine_e] (hd.simps).*

Sledgehammer ran E, SPASS, Vampire, SInE-E, and Z3 in parallel. Depending on which provers are installed and how many processor cores are available, some of the provers might be missing or present with a *remote_* prefix.

For each successful prover, Sledgehammer gives a one-liner proof that uses the *metis* or *smt* method. You can click the proof to insert it into the theory text. You can click the "**sledgehammer** *minimize*" command if you want to look for a shorter (and probably faster) proof. But here the proof found by E looks perfect, so click it to finish the proof.

You can ask Sledgehammer for an Isar text proof by passing the *isar_proof* option:

sledgehammer [*isar_proof*]

When Isar proof construction is successful, it can yield proofs that are more readable and also faster than the *metis* one-liners. This feature is experimental and is only available for ATPs.

4 Hints

For best results, first simplify your problem by calling *auto* or at least *safe* followed by *simp_all*. None of the ATPs contain arithmetic decision procedures. They are not especially good at heavy rewriting, but because they regard equations as undirected, they often prove theorems that require the reverse orientation of a *simp* rule. Higher-order problems can be tackled, but the success rate is better for first-order problems. Hence, you may get better results if you first simplify the problem to remove higher-order features.

Note that problems can be easy for *auto* and difficult for ATPs, but the reverse is also true, so don't be discouraged if your first attempts fail. Because the system refers to all theorems known to Isabelle, it is particularly suitable when your goal has a short proof from lemmas that you don't know about.

5 Command Syntax

Sledgehammer can be invoked at any point when there is an open goal by entering the **sledgehammer** command in the theory file. Its general syntax is as follows:

sledgehammer *subcommand*? *options*? *facts_override*? *num*?

For convenience, Sledgehammer is also available in the "Commands" submenu of the "Isabelle" menu in Proof General or by pressing the Emacs key sequence C-c C-a C-s. This is equivalent to entering the **sledgehammer** command with no arguments in the theory text.

In the general syntax, the *subcommand* may be any of the following:

- ***run* (the default):** Runs Sledgehammer on subgoal number *num* (1 by default), with the given options and facts.
- ***minimize*:** Attempts to minimize the provided facts (specified in the *facts_override* argument) to obtain a simpler proof involving fewer facts. The options and goal number are as for *run*.
- ***messages*:** Redisplays recent messages issued by Sledgehammer. This allows you to examine results that might have been lost due to Sledgehammer's asynchronous nature. The *num* argument specifies a limit on the number of messages to display (5 by default).
- ***available_provers*:** Prints the list of installed provers. See §2 and §6.1 for more information on how to install automatic provers.
- ***running_provers*:** Prints information about currently running automatic provers, including elapsed runtime and remaining time until timeout.
- ***kill_provers*:** Terminates all running automatic provers.
- ***refresh_tptp*:** Refreshes the list of remote ATPs available at SystemOnTPTP [9].

Sledgehammer's behavior can be influenced by various *options*, which can be specified in brackets after the **sledgehammer** command. The *options* are a list of key–value pairs of the form "$[k_1 = v_1, \ldots, k_n = v_n]$". For Boolean options, "= *true*" is optional. For example:

sledgehammer [*isar_proof*, *timeout* = 120 *s*]

Default values can be set using **sledgehammer_params**:

sledgehammer_params *options*

The supported options are described in §6.

The *facts_override* argument lets you alter the set of facts that go through the relevance filter. It may be of the form "(*facts*)", where *facts* is a space-separated list of Isabelle facts (theorems, local assumptions, etc.), in which case the relevance filter is bypassed and the given facts are used. It may also be of the form "(*add*: $facts_1$)", "(*del*: $facts_2$)", or "(*add*: $facts_1$ *del*: $facts_2$)", where the relevance filter is instructed to proceed as usual except that it should consider $facts_1$ highly-relevant and $facts_2$ fully irrelevant.

You can instruct Sledgehammer to run automatically on newly entered theorems by enabling the "Auto Sledgehammer" option from the "Isabelle" menu in Proof General. For automatic runs, only the first prover set using *provers* (§6.1) is considered, *verbose* (§6.4) and *debug* (§6.4) are disabled, fewer facts are passed to the prover, and *timeout* (§6.1) is superseded by the "Auto Tools Time Limit" in Proof General's "Isabelle" menu. Sledgehammer's output is also more concise.

6 Option Reference

Sledgehammer's options are categorized as follows: mode of operation (§6.1), problem encoding (§6.2), relevance filter (§6.3), output format (§6.4), and authentication (§6.5).

The descriptions below refer to the following syntactic quantities:

- ⟨***string***⟩: A string.
- ⟨***bool***⟩: *true* or *false*.
- ⟨***bool_or_smart***⟩: *true*, *false*, or *smart*.
- ⟨***int***⟩: An integer.
- ⟨***float_pair***⟩: A pair of floating-point numbers (e.g., 0.6 0.95).
- ⟨***int_or_smart***⟩: An integer or *smart*.
- ⟨***float_or_none***⟩: An integer (e.g., 60) or floating-point number (e.g., 0.5) expressing a number of seconds, or the keyword *none* (∞ seconds).

Default values are indicated in square brackets. Boolean options have a negated counterpart (e.g., *blocking* vs. *non_blocking*). When setting Boolean options, "= *true*" may be omitted.

6.1 Mode of Operation

provers = ⟨***string***⟩

Specifies the automatic provers to use as a space-separated list (e.g., "*e spass*"). The following provers are supported:

- ***e*:** E is an ATP developed by Stephan Schulz [7]. To use E, set the environment variable `E_HOME` to the directory that contains the `eproof` executable, or install the prebuilt E package from Isabelle's download page. See §2 for details.
- ***spass*:** SPASS is an ATP developed by Christoph Weidenbach et al. [10]. To use SPASS, set the environment variable `SPASS_HOME` to the directory that contains the `SPASS` executable, or install the prebuilt SPASS package from Isabelle's download page. Sledgehammer requires version 3.5 or above. See §2 for details.
- ***vampire*:** Vampire is an ATP developed by Andrei Voronkov and his colleagues [6]. To use Vampire, set the environment variable `VAMPIRE_HOME` to the directory that contains the `vampire` executable. Sledgehammer has been tested with versions 11, 0.6, and 1.0.
- ***z3*:** Z3 is an SMT solver developed at Microsoft Research [11]. To use Z3, set the environment variable `Z3_SOLVER` to the complete path of the executable, including the file name. Sledgehammer has been tested with 2.7 to 2.15.
- ***yices*:** Yices is an SMT solver developed at SRI [3]. To use Yices, set the environment variable `YICES_SOLVER` to the complete path of the executable, including the file name. Sledgehammer has been tested with version 1.0.
- ***cvc3*:** CVC3 is an SMT solver developed by Clark Barrett, Cesare Tinelli, and their colleagues [1]. To use CVC3, set the environment variable `CVC3_SOLVER` to the complete path of the executable, including the file name. Sledgehammer has been tested with version 2.2.
- ***remote_e*:** The remote version of E runs on Geoff Sutcliffe's Miami servers [9].

- ***remote_vampire*:** The remote version of Vampire runs on Geoff Sutcliffe's Miami servers. Version 9 is used.
- ***remote_sine_e*:** SInE-E is a metaprover developed by Kryštof Hoder [4] based on E. The remote version of SInE runs on Geoff Sutcliffe's Miami servers.
- ***remote_snark*:** SNARK is a prover developed by Stickel et al. [8]. The remote version of SNARK runs on Geoff Sutcliffe's Miami servers.
- ***remote_z3*:** The remote version of Z3 runs on servers at the TU München (or wherever `REMOTE_SMT_URL` is set to point).
- ***remote_cvc3*:** The remote version of CVC3 runs on servers at the TU München (or wherever `REMOTE_SMT_URL` is set to point).

By default, Sledgehammer will run E, SPASS, Vampire, SInE-E, and Z3 (or whatever the SMT module's *smt_solver* configuration option is set to) in parallel—either locally or remotely, depending on the number of processor cores available. For historical reasons, the default value of this option can be overridden using the option "Sledgehammer: Provers" from the "Isabelle" menu in Proof General.

It is a good idea to run several provers in parallel, although it could slow down your machine. Running E, SPASS, Vampire, and SInE-E together for 5 seconds yields a better success rate than running the most effective of these (Vampire) for 120 seconds [2].

prover $= \langle$***string***$\rangle$

Alias for *provers*.

atps $= \langle$***string***$\rangle$

Legacy alias for *provers*.

atp $= \langle$***string***$\rangle$

Legacy alias for *provers*.

timeout $= \langle$***float_or_none***$\rangle$ **[30]**

Specifies the maximum number of seconds that the automatic provers should spend searching for a proof. For historical reasons, the default value of this option can be overridden using the option "Sledgehammer: Time Limit" from the "Isabelle" menu in Proof General.

blocking $[= \langle$***bool***$\rangle]$ **[*false*]** **(neg.: *non_blocking*)**

Specifies whether the **sledgehammer** command should operate synchronously. The asynchronous (non-blocking) mode lets the user start

proving the putative theorem manually while Sledgehammer looks for a proof, but it can also be more confusing.

overlord $[= \langle bool \rangle]$ **[*false*]** **(neg.: *no_overlord*)**

Specifies whether Sledgehammer should put its temporary files in `$ISABELLE_HOME_USER`, which is useful for debugging Sledgehammer but also unsafe if several instances of the tool are run simultaneously. The files are identified by the prefix `prob_`; you may safely remove them after Sledgehammer has run.

See also *debug* (§6.4).

6.2 Problem Encoding

explicit_apply $[= \langle bool \rangle]$ **[*false*]** **(neg.: *implicit_apply*)**

Specifies whether function application should be encoded as an explicit "apply" operator in ATP problems. If the option is set to *false*, each function will be directly applied to as many arguments as possible. Enabling this option can sometimes help discover higher-order proofs that otherwise would not be found.

full_types $[= \langle bool \rangle]$ **[*false*]** **(neg.: *partial_types*)**

Specifies whether full-type information is encoded in ATP problems. Enabling this option can prevent the discovery of type-incorrect proofs, but it also tends to slow down the ATPs significantly. For historical reasons, the default value of this option can be overridden using the option "Sledgehammer: Full Types" from the "Isabelle" menu in Proof General.

6.3 Relevance Filter

relevance_thresholds $= \langle float\_pair \rangle$ **[0.45 0.85]**

Specifies the thresholds above which facts are considered relevant by the relevance filter. The first threshold is used for the first iteration of the relevance filter and the second threshold is used for the last iteration (if it is reached). The effective threshold is quadratically interpolated for the other iterations. Each threshold ranges from 0 to 1, where 0 means that all theorems are relevant and 1 only theorems that refer to previously seen constants.

max_relevant $[= \langle$***bool_or_smart***$\rangle]$ **[*smart*]** **(neg.: *int_or_smart*)**

Specifies the maximum number of facts that may be returned by the relevance filter. If the option is set to *smart*, it is set to a value that was empirically found to be appropriate for the prover. A typical value would be 300.

6.4 Output Format

verbose $[= \langle$***bool***$\rangle]$ **[*false*]** **(neg.: *quiet*)**

Specifies whether the **sledgehammer** command should explain what it does. This option is implicitly disabled for automatic runs.

debug $[= \langle$***bool***$\rangle]$ **[*false*]** **(neg.: *no_debug*)**

Specifies whether Sledgehammer should display additional debugging information beyond what *verbose* already displays. Enabling *debug* also enables *verbose* and *blocking* (§6.1) behind the scenes. The *debug* option is implicitly disabled for automatic runs.

See also *overlord* (§6.1).

isar_proof $[= \langle$***bool***$\rangle]$ **[*false*]** **(neg.: *no_isar_proof*)**

Specifies whether Isar proofs should be output in addition to one-liner *metis* proofs. Isar proof construction is still experimental and often fails; however, they are usually faster and sometimes more robust than *metis* proofs.

isar_shrink_factor $= \langle$***int***$\rangle$ **[1]**

Specifies the granularity of the Isar proof. A value of n indicates that each Isar proof step should correspond to a group of up to n consecutive proof steps in the ATP proof.

6.5 Authentication

expect $= \langle$***string***$\rangle$

Specifies the expected outcome, which must be one of the following:

- ***some*****:** Sledgehammer found a (potentially unsound) proof.
- ***none*****:** Sledgehammer found no proof.
- ***unknown*****:** Sledgehammer encountered some problem.

Sledgehammer emits an error (if *blocking* is enabled) or a warning (otherwise) if the actual outcome differs from the expected outcome. This option is useful for regression testing.

See also *blocking* (§6.1).

References

[1] C. Barrett and C. Tinelli. CVC3. In W. Damm and H. Hermanns, editors, *CAV*, volume 4590 of *Lecture Notes in Computer Science*, pages 298–302. Springer, 2007.

[2] S. Böhme and T. Nipkow. Sledgehammer: Judgement day. In J. Giesl and R. Hähnle, editors, *Automated Reasoning: IJCAR 2010*, Lecture Notes in Computer Science. Springer-Verlag, 2010.

[3] B. Dutertre and L. de Moura. The Yices SMT solver, 2006.

[4] K. Hoder. Sine (sumo inference engine). `http://www.cs.man.ac.uk/~hoderk/sine/`.

[5] J. Hurd. Metis theorem prover. `http://www.gilith.com/software/metis/`.

[6] A. Riazanov and A. Voronkov. The design and implementation of Vampire. *Journal of AI Communications*, 15(2/3):91–110, 2002.

[7] S. Schulz. E—a brainiac theorem prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.

[8] M. Stickel, R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood. Deductive composition of astronomical software from subroutine libraries. In A. Bundy, editor, *Automated Deduction — CADE-12 International Conference*, LNAI 814, pages 341–355. Springer, 1994.

[9] G. Sutcliffe. System description: SystemOnTPTP. In D. McAllester, editor, *Automated Deduction — CADE-17 International Conference*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 406–410. Springer-Verlag, 2000.

[10] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski. SPASS version 3.5. `http://www.spass-prover.org/publications/spass.pdf`.

[11] Z3: An efficient SMT solver. `http://research.microsoft.com/en-us/um/redmond/projects/z3/`.