Hammering Away

A User's Guide to Sledgehammer for Isabelle/HOL

Jasmin Christian Blanchette
Institut für Informatik, Technische Universität München

June 21, 2010

Contents

1 Introduction

Sledgehammer is a tool that applies first-order automatic theorem provers (ATPs) on the current goal. The supported ATPs are E [4], SPASS [6], and Vampire [3], which can be run locally or remotely via the SystemOnTPTP web service [5].

The problem passed to ATPs consists of the current goal together with a heuristic selection of facts (theorems) from the current theory context, filtered by relevance. The result of a successful ATP proof search is some source text that usually (but not always) reconstructs the proof within Isabelle, without requiring the ATPs again. The reconstructed proof relies on the general-purpose Metis prover [2], which is fully integrated into Isabelle/HOL, with explicit inferences going through the kernel. Thus its results are correct by construction.

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). Currently, E, SPASS, and Vampire are supported. All of these are available remotely via SystemOnTPTP [5], but if you want better performance you will need to install at least E and SPASS locally.

There are three main ways to install E and SPASS:

- If you installed an official Isabelle package with everything inside, it should already include properly setup executables for E and SPASS, ready to use.
- Otherwise, 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 `~/.isabelle/etc/components` does not exist yet and you extracted SPASS to `/usr/local/spass-3.7`, create the file `~/.isabelle/etc/components` with the single line

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

- If you prefer to build E or SPASS yourself, feel free to do so and set the environment variable `E_HOME` or `SPASS_HOME` to the directory that contains the `eproof` or `SPASS` executable, respectively.

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

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**

After a few seconds, Sledgehammer produces the following output:

> *Sledgehammer: ATP "e" for subgoal 1:*
> $([a] = [b]) = (a = b)$
> *Try this command:* **by** *(metis hd.simps).*
> *To minimize the number of lemmas, try this command:*
> ***sledgehammer*** *minimize [atp = e] (hd.simps).*
>
> *Sledgehammer: ATP "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 command:*
> ***sledgehammer*** *minimize [atp = spass] (insert_Nil last_ConsL).*
>
> *Sledgehammer: ATP "remote_vampire" for subgoal 1:*
> $([a] = [b]) = (a = b)$
> *Try this command:* **by** *(metis One_nat_def_raw empty_replicate insert_Nil last_ConsL replicate_Suc).*
> *To minimize the number of lemmas, try this command:*
> ***sledgehammer*** *minimize [atp = remote_vampire] (One_nat_def_raw empty_replicate insert_Nil last_ConsL replicate_Suc).*

Sledgehammer ran E, SPASS, and the remote version of Vampire in parallel. If E and SPASS are not installed (§2), you will see references to their remote American cousins *remote_e* and *remote_spass* instead of *e* and *spass*.

Based on each ATP proof, Sledgehammer gives a one-liner proof that uses the *metis* method. You can click them and insert them into the theory text. You can click the "**sledgehammer** *minimize*" command if you want to look for a shorter (and 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.

4 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_atps*:** Prints the list of installed ATPs. See §2 and §5.1 for more information on how to install ATPs.
- ***running_atps*:** Prints information about currently running ATPs, including elapsed runtime and remaining time until timeout.
- ***kill_atps*:** Terminates all running ATPs.
- ***refresh_tptp*:** Refreshes the list of remote ATPs available at SystemOnTPTP [5].

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, "$= \mathit{true}$" 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 §5.

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.

5 Option Reference

Sledgehammer's options are categorized as follows: mode of operation (§5.1), problem encoding (§5.2), output format (§5.3), and timeouts (§5.4).

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.
- ⟨***time***⟩: An integer followed by *min* (minutes), *s* (seconds), or *ms* (milliseconds), or the keyword *none* (∞ years).

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

5.1 Mode of Operation

atps = ⟨***string***⟩

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

- ***e*:** E is an ATP developed by Stephan Schulz [4]. 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. [6]. 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. See §2 for details.
- ***spass_tptp*:** Same as the above, except that Sledgehammer communicates with SPASS using the TPTP syntax rather than the native DFG syntax. This ATP is provided for experimental purposes.
- ***vampire*:** Vampire is an ATP developed by Andrei Voronkov and his colleagues [3]. To use Vampire, set the environment variable `VAMPIRE_HOME` to the directory that contains the `vampire` executable.
- ***remote_e*:** The remote version of E executes on Geoff Sutcliffe's Miami servers [5].

- ***remote_spass*:** The remote version of SPASS executes on Geoff Sutcliffe's Miami servers.
- ***remote_vampire*:** The remote version of Vampire executes on Geoff Sutcliffe's Miami servers. Version 9 is used.

By default, Sledgehammer will run E, SPASS, and Vampire in parallel. For E and SPASS, it will use any locally installed version if available, falling back on the remote versions if necessary. For historical reasons, the default value of this option can be overridden using the option "Sledgehammer: ATPs" from the "Isabelle" menu in Proof General.

It is a good idea to run several ATPs in parallel, although it could slow down your machine. Tobias Nipkow observed that running E, SPASS, and Vampire together for 5 seconds yields the same success rate as running the most effective of these (Vampire) for 120 seconds [1].

atp = ⟨***string***⟩

Alias for *atps*.

overlord [= ⟨***bool***⟩] **[*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* (§5.3).

5.2 Problem Encoding

explicit_apply [= ⟨***bool***⟩] **[*false*]** (**neg.:** ***implicit_apply***)

Specifies whether function application should be encoded as an explicit "apply" operator. 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 [= ⟨***bool***⟩] **[*false*]** (**neg.:** ***partial_types***)

Specifies whether full-type information is exported. 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: ATPs" from the "Isabelle" menu in Proof General.

relevance_threshold = ⟨***int***⟩ **[*50*]**

Specifies the threshold above which facts are considered relevant by the relevance filter. The option ranges from 0 to 100, where 0 means that all theorems are relevant.

relevance_convergence = ⟨***int***⟩ **[*320*]**

Specifies the convergence quotient, multiplied by 100, used by the relevance filter. This quotient is used by the relevance filter to scale down the relevance of facts at each iteration of the filter.

theory_relevant [= ⟨***bool_or_smart***⟩] **[*smart*]** **(neg.: *theory_irrelevant*)**

Specifies whether the theory from which a fact comes should be taken into consideration by the relevance filter. If the option is set to *smart*, it is taken to be *true* for SPASS and *false* for E and Vampire, because empirical results suggest that these are the best settings.

defs_relevant [= ⟨***bool***⟩] **[*false*]** **(neg.: *defs_irrelevant*)**

Specifies whether the definition of constants occurring in the formula to prove should be considered particularly relevant. Enabling this option tends to lead to larger problems and typically slows down the ATPs.

respect_no_atp [= ⟨***bool***⟩] **[*true*]** **(neg.: *ignore_no_atp*)**

Specifies whether Sledgehammer should honor the *no_atp* attributes. The *no_atp* attributes marks theorems that tend to confuse ATPs, typically because they can lead to unsound ATP proofs [1]. It is normally a good idea to leave this option enabled, unless you are debugging Sledgehammer.

5.3 Output Format

verbose [= ⟨***bool***⟩] **[*false*]** **(neg.: *quiet*)**

Specifies whether the **sledgehammer** command should explain what it does.

debug [= ⟨***bool***⟩] **[*false*]** **(neg.: *no_debug*)**

Specifies whether Nitpick should display additional debugging information beyond what *verbose* already displays. Enabling *debug* also enables *verbose* behind the scenes.

See also *overlord* (§5.1).

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.

5.4 Timeouts

timeout $= \langle$***time***$\rangle$ **[60 *s*]**

Specifies the maximum amount of time that the ATPs should spend looking 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.

minimize_timeout $= \langle$***time***$\rangle$ **[5 *s*]**

Specifies the maximum amount of time that the ATPs should spend looking for a proof for **sledgehammer** *minimize.*

References

[1] 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.

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

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

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

[5] G. Sutcliffe. System description: SystemOnTPTP. In J. G. Carbonell and J. Siekmann, editors, *Automated Deduction — CADE-17 International Conference*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 406–410. Springer-Verlag, 2000.

[6] 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`.