universität freiburg

Sledgehammer

Mathias Fleury

Tutorial Vienna 2023 (Part II)

Since the last part of the tutorial, Coq has announced a new name Rocq[1].

[1] not to be confused with the programming language Roc

Outline

1. Overview

2. Filtering

3. Encoding

4. Reconstruct the Proof

5. Conclusion

Overview

How does Sledgehammer work?

1. Fact filtering
2. Encode the problems
3. Get a proof from an automatic theorem prover
4. Reconstruct it

Filtering

MePo

Name: Meng-Paulson

Relevant facts based on the symbols

Example of Problems

Taken from MaSh paper.

$$\texttt{used}\,[] \subseteq \texttt{used}\,\mathrm{evs}$$

with

$$\texttt{used}\,[] = \bigcup_B \mathrm{parts}(\mathrm{initState}\,B)$$

$$X \in \mathrm{parts}(\mathrm{initState}\,B) \Rightarrow X \in \mathrm{used}\,\mathrm{evs}$$

$$(\forall x.\ x \in A \Rightarrow x \in B) \Rightarrow A \subseteq B$$

$$b \in \bigcup_{x \in B} B\,x \leftrightarrow \exists x \in A.\ b \in B\,x$$

But: “The first two lemmas are ranked 6807th and 6808th”.

MaSh

Name: Machine learning for Sledgehammer.

Learn when facts are useful in Sledgehammer proofs.

Features a theorem:

- types (ignoring everything deep)
- theory it comes from
- kind of rule
- presence of existential quantifiers or λ-abstraction

MeSh

Actually MaSh not good enough, so
MeSh taking average weight = MaSh (weight: .8) + MePo (weight: .2)

Experiments

Mirabelle: tool for calling Sledgehammer on all goals

Warning:

- do not forget to delete the MaSh state (file `mash_state`)
- do not run tests in parallel

I have reviewed papers where I think this happened, but it is very hard to know.

Encoding

What is not translated?

For superposition solvers:

- arithmetic
- datatype
- HO (except for vampire)
- types (for some old TPTP solvers)

Arithmetic

Arithmetic is hard to support for superposition provers, even if there are attempts[2] (hierarchical superposition, vampire with SMT, ...)

Translation of natural numbers with new constraints $n \geq 0$ and translation like $a - b = (\texttt{if}\ a < b\ \texttt{then}\ 0\ \texttt{else}\ a - b)$. Flag: `smt_nat_as_int`.

[2] sorry if I forgot you favorite attempt

Even More Arithmetic

Flag: `z3_extension`
Generates division (with different definition for $a/0$)

Even More Arithmetic

Flag: `z3_extension`
Generates division (with different definition for $a/0$)

Current wisdom: not enough arithmetic goals in Isabelle for it to be useful.

Even More Arithmetic

Flag: `z3_extension`
Generates division (with different definition for $a/0$)

Current wisdom: not enough arithmetic goals in Isabelle for it to be useful.

But: different trade-off for bitvectors, as you can translate into the built-in version

Datatypes

Generation for the pre-standard definition of the SMTLib.

Supported by cvc5 only (?)

Not activated by default

Higher-Order

Lifting (default for SMT): add equation `c x1 ... xn = t`

Higher-Order

Lifting (default for SMT): add equation `c x1 ... xn = t`

Curry Combinators: I, K, S, B, C with axiomatization of the combinators

Reconstruct the Proof

Proof Formats

TPTP (I) E

Proof Formats

TPTP (I)	E
TPTP (II)	Vampire (arguments of skolemization in the opposite direction)

Proof Formats

TPTP (I)	E
TPTP (II)	Vampire (arguments of skolemization in the opposite direction)
TPTP (III)	Satallax with backwards steps in the middle; Leo-II can call E and return all facts

Proof Formats

TPTP (I)	E
TPTP (II)	Vampire (arguments of skolemization in the opposite direction)
TPTP (III)	Satallax with backwards steps in the middle; Leo-II can call E and return all facts
TPTP (IV)	Splitting

Proof Formats

TPTP (I)	E
TPTP (II)	Vampire (arguments of skolemization in the opposite direction)
TPTP (III)	Satallax with backwards steps in the middle; Leo-II can call E and return all facts
TPTP (IV)	Splitting
Alethe	veriT

Proof Formats

TPTP (I)	E
TPTP (II)	Vampire (arguments of skolemization in the opposite direction)
TPTP (III)	Satallax with backwards steps in the middle; Leo-II can call E and return all facts
TPTP (IV)	Splitting
Alethe	veriT
Z3	Z3 (unmaintained)

Isar

UNSAT core : can a built-in tactic reconstruct the proof?

Isar : take the proof generated by the prover, massage it, redirect it (by introducing $\vee$ in some cases), compress it [JAR'15, Blanchette et al, `https://smolka.st/papers/isar_jar.pdf`]

Problems:

1. Subproofs are not really supported
2. Alethe tries to keep equivalences, which is bad for the redirection algorithm

Conclusion

Full Tests

*Seventeen Provers Under the Hammer*ITP'22: `https://drops.dagstuhl.de/storage/00lipics/lipics-vol237-itp2022/LIPIcs.ITP.2022.8/LIPIcs.ITP.2022.8.pdf`

There is a lot of duplicate effort between Sledgehammer and the back-ends (monomorphization, translation HO to FO, fact selection, ...).

Sledgehammer is extremely useful during development.

My dream: overfitting MaSh and solvers on the theories I am currently working on.