

# Extending Sledgehammer with Induction Provers

Jasmin Blanchette

Inria & LORIA, Nancy

MPI für Informatik, Saarbrücken

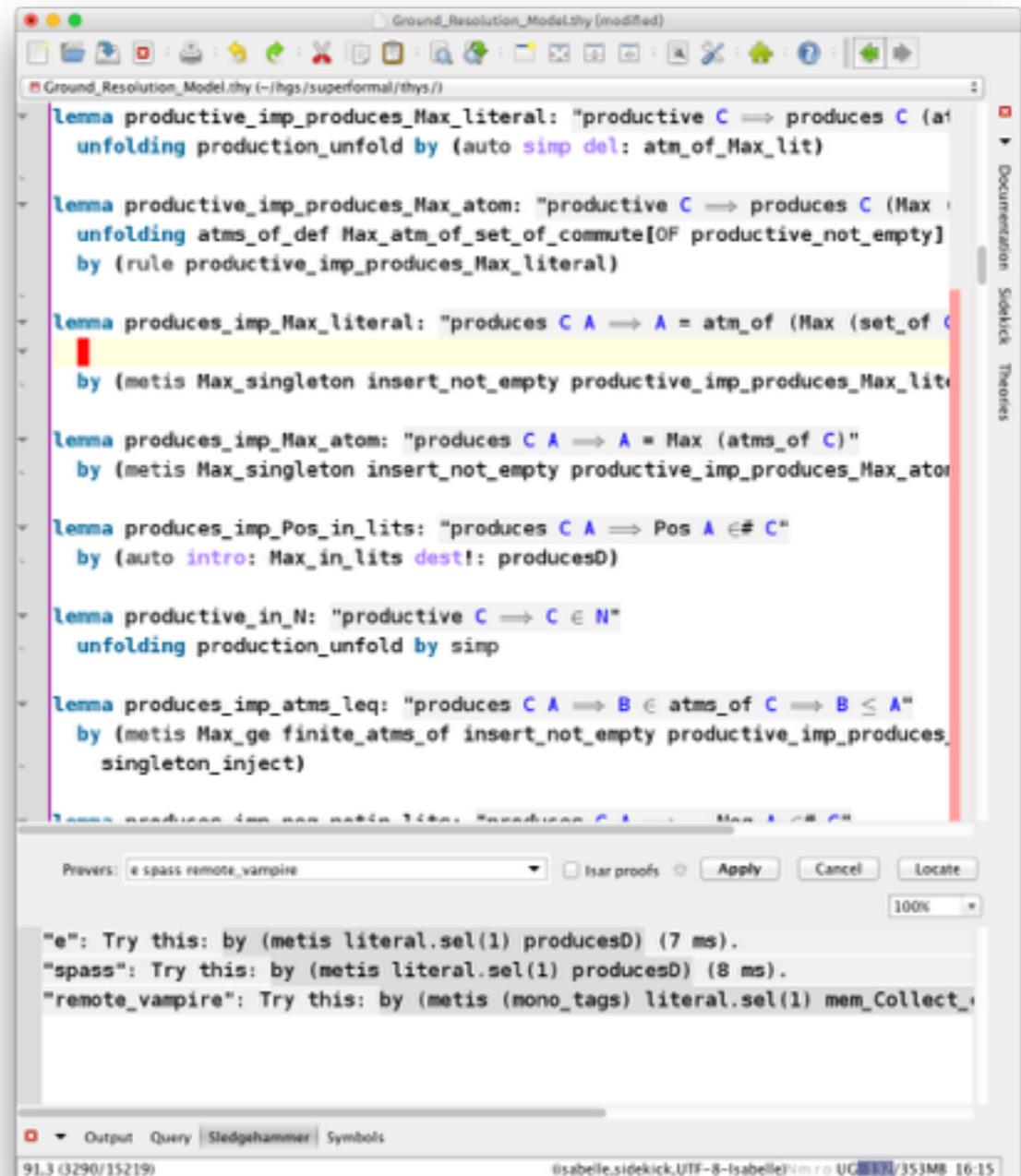
# Isabelle

- general purpose proof assistant
- HOL: higher-order logic
- small kernel
- safe definitional principles
- high proof automation
- modern user interface



# Isabelle

- general purpose proof assistant
- HOL: higher-order logic
- small kernel
- safe definitional principles
- high proof automation
- modern user interface



```
Ground_Resolution_Model.thy (modified)
Ground_Resolution_Model.thy (-) [hgs/superformal/this/]
lemma productive_imp_produces_Max_literal: "productive C  $\implies$  produces C (at
  unfolding production_unfold by (auto simp del: atm_of_Max_lit)

lemma productive_imp_produces_Max_atom: "productive C  $\implies$  produces C (Max
  unfolding atms_of_def Max_atm_of_set_of_commute[OF productive_not_empty]
  by (rule productive_imp_produces_Max_literal)

lemma produces_imp_Max_literal: "produces C A  $\implies$  A = atm_of (Max (set_of C
  by (metis Max_singleton insert_not_empty productive_imp_produces_Max_lit

lemma produces_imp_Max_atom: "produces C A  $\implies$  A = Max (atms_of C)"
  by (metis Max_singleton insert_not_empty productive_imp_produces_Max_atom

lemma produces_imp_Pos_in_lits: "produces C A  $\implies$  Pos A  $\in$  C"
  by (auto intro: Max_in_lits dest!: producesD)

lemma productive_in_N: "productive C  $\implies$  C  $\in$  N"
  unfolding production_unfold by simp

lemma produces_imp_atms_leq: "produces C A  $\implies$  B  $\in$  atms_of C  $\implies$  B  $\leq$  A"
  by (metis Max_ge finite_atms_of insert_not_empty productive_imp_produces
    singleton_inject)

lemma produces_imp_pos_not_in_lits: "produces C A  $\implies$  Pos A  $\in$  C"

Provers: e spass remote_vampire  Isar proofs  Apply Cancel Locate
100%

"e": Try this: by (metis literal.sel(1) producesD) (7 ms).
"spass": Try this: by (metis literal.sel(1) producesD) (8 ms).
"remote_vampire": Try this: by (metis (mono_tags) literal.sel(1) mem_Collect_

Output Query Sledgehammer Symbols
91.3 (3290/15219) @isabelle.sidekick,UTF-8-Isabelle/Unix/UC/353MB 16:15
```

# Automatic Proofs and Counterexamples

- **Sledgehammer** finds proofs using external ATPs  
CVC4, E, LEO-II, Satallax, SPASS, Vampire, veriT, Z3, ...
- **Nitpick** finds counterexamples using external model finder  
Kodkod (Alloy's backend)
- **Quickcheck** finds counterexamples using random testing,  
exhaustive testing, narrowing

powerful automation  $\implies$  productive users



Larry Paulson

Sledgehammer has found some incredible proofs. I would estimate the improvement in productivity as a factor of **at least three, maybe five.**



Gerwin Klein

**Nitpick (sehr nützlich)** hat mich nach ungefähr der Hälfte überzeugt, dass meine Idee mit dem Merge nicht funktioniert.

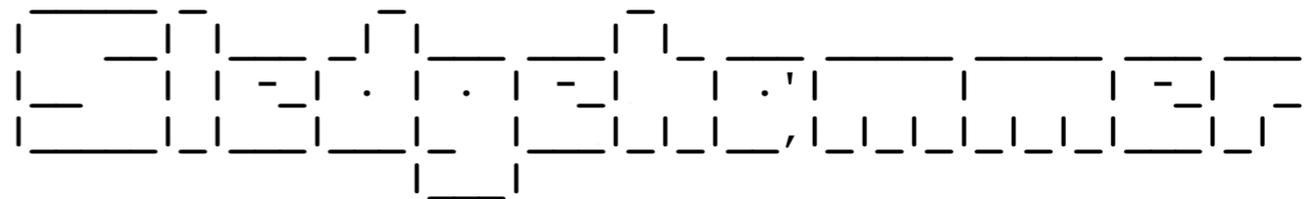


Tobias Nipkow

Last night I got stuck on a goal I was sure was a theorem. After 5–10 minutes I gave Nitpick a try, and **within a few secs it had found a splendid counterexample!** Unfortunately I now have to revise my formalization :-)

Developing proofs without Sledgehammer is like **walking** as opposed to **running**.

(\*This homework was presented to you by:



\*)

# Sledgehammer-Generated Proofs

```
have f1:  $\forall n. \text{rat\_of\_int } \lfloor \text{rat\_of\_nat } n \rfloor = \text{rat\_of\_nat } n$   
  using of_int_of_nat_eq by simp  
have f2:  $\forall n. \text{real\_of\_int } \lfloor \text{rat\_of\_nat } n \rfloor = \text{real } n$   
  using of_int_of_nat_eq real_eq_of_nat by auto  
have f3:  $\forall i \text{ ia}. \lfloor \text{rat\_of\_int } i / \text{rat\_of\_int } \text{ia} \rfloor = \lfloor \text{real\_of\_int } i / \text{real\_of\_int } \text{ia} \rfloor$   
  using div_is_floor_divide_rat div_is_floor_divide_real by simp  
have f4:  $0 < \lfloor \text{rat\_of\_nat } p \rfloor$   
  using p by simp  
have  $\lfloor S \rfloor \leq s$   
  using less_floor_le_iff by auto  
hence  $\lfloor \text{rat\_of\_int } \lfloor S \rfloor / \text{rat\_of\_nat } p \rfloor \leq \lfloor \text{rat\_of\_int } s / \text{rat\_of\_nat } p \rfloor$   
  using f1 f3 f4 by (metis div_is_floor_divide_real zdiv_mono1)  
hence  $\lfloor S / \text{real } p \rfloor \leq \lfloor \text{rat\_of\_int } s / \text{rat\_of\_nat } p \rfloor$   
  using f1 f2 f3 f4 by (metis div_is_floor_divide_real floor_div_pos_int)  
thus  $S / \text{real } p \leq \text{real\_of\_int } (s \text{ div int } p) + 1$   
  using f1 f3  
  by (metis div_is_floor_divide_real floor_le_iff floor_of_nat less_eq_real_def)
```

# Sledgehammer-Generated Proofs

```
lemma powsum_ub:  $i \leq n \implies x^i \leq x_0^n$ 
proof (induct n)
  case 0 show case
    by (metis (hide_lams, mono_tags) 0.prem1 eq_iff le_0_eq power_0 powsum_00)
next
  case (Suc n) show case
  proof –
    { assume aa1:  $\text{Suc } n \neq i$ 
      have ff1:  $x_0^{\text{Suc } n} \leq x_0^{\text{Suc } n} \wedge \text{Suc } n \neq i$ 
        using aa1 by fastforce
      have ff2:  $\exists a. x_0^n + a \leq x_0^{\text{Suc } n} \wedge \text{Suc } n \neq i$ 
        using ff1 powsum2 by auto
      have  $x^i \leq x_0^{\text{Suc } n}$ 
        using ff2 by (metis Suc.hyps Suc.prem1 add_lub le_SucE less_eq_def) }
    thus  $x^i \leq x_0^{\text{Suc } n}$ 
      using less_eq_def powsum_split_var2 by auto
  qed
qed
```

# Wish List for Prover Authors

- detailed output similar to input syntax
- record which induction **principle** and **instance** were used
- respect the **Russian doll** principle
- well-documented input
- rich input

# Datatypes

- Polymorphism:

```
datatype 'a list = Nil | Cons 'a ('a list)
```

- Mutual recursion:

```
datatype 'a tree = Node 'a ('a trees)
and 'a trees = TNil | TCons ('a tree) ('a trees)
```

- Nested recursion:

```
datatype 'a tree = Node 'a ('a tree list)
                                     fset?
                                     llist?
```

# Codatatypes

- Polymorphism:

```
codatatype 'a llist = LNil | LCons 'a ('a llist)
```

- Mutual recursion:

```
codatatype 'a ltree = LNode 'a ('a ltrees)
```

```
and 'a ltrees = LNil | LTCons ('a ltree) ('a ltrees)
```

- Nested recursion:

```
codatatype 'a ltree = LNode 'a ('a ltree llist)
```

fset?

list?

Non-Free Datatypes?

Rule (Co)induction?

Intergration in a HO ATP?

Work with Arbitrary  
Induction Principles?

# Syntax for Benchmarks

- Should be **natural**, not an encoding
- Ideally, extension of something existing
- TIS syntax!
- Theories?
- Important to have a parser/AST
- Checker

Thank you!