Are Translations between Proof Assistants Possible or Even Desirable at All?

Jasmin Christian Blanchette

Inria Nancy – Grand Est, France
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Scenario 1
Flyspeck in HOL Light, Isabelle/HOL, and Coq

Scenario 2
Analysis in HOL Light and Isabelle/HOL

"Larry": In the past, people have published a number of papers on the automatic translation of proofs from one system to another. As far as I know, **this translated material is almost never used**. Considering the proofs I have translated, the reason is obvious: **the proofs really need to be made native to the new system**. In the case of "John"'s material, everything he does is confined to the special case of R^n. He is forced to identify C with R^2 and he is frequently forced to translate explicitly between R^1 and R. Automatically-translated proofs would suffer all the same drawbacks. But instead, **many of the translated results are applicable to general topological spaces, or to metric spaces, etc.**

Scenario 2
Analysis in HOL Light and Isabelle/HOL

"Tobias": You are right: **automatic translation** between even slightly different logics **is hardly used,** mostly because the **resulting proofs are not optimal in the target system**, eg no structured proofs and no type classes.

Scenario 2
Analysis in HOL Light and Isabelle/HOL

"Larry": I have always maintained that one should never attempt to formalise proofs that one didn't understand, and yet here on many occasions I have done exactly that. **I've ported many proofs without grasping the underlying concepts** and reasoning. No doubt this made things much more difficult, especially as regards knowing what is likely to be true and what not, yet it was possible. The key was to look in the HOL Light proof for clues about the proof's milestones. Sometimes **explicit subgoals were visible in the proof text** and on other occasions they could be reconstructed, especially when named theorems were instantiated with specific expressions, when clearly the theorem's preconditions would need to be proved. Then one could create a proof structure, and in many cases, **fill in the gaps using sledgehammer**.

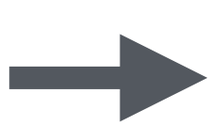
Scenario 2
Analysis in HOL Light and Isabelle/HOL

"Larry": It seems that I've ported approximately **17,000 lines of HOL Light proofs**. I did this **without ever launching HOL Light** (in fact I don't know how to do this), almost never looking at a mathematical textbook or paper, and very often not understanding the result being proved or the properties involved. And indeed, **often with a television on, sometimes with a glass of wine at my side**.

It's kind of amusing, but **is it science**? What exactly are the conclusions?

I thought of focusing on the attached proof, **which apparently is a very important result (honestly I have no idea)** and at 688 lines, it is quite a monstrosity.

```
let OUTSIDE_COMPACT_IN_OPEN = prove
(`!s t:real^N->bool.
      compact s /\ open t /\ s SUBSET t /\ ~(t = {})
      ==> ~(outside s INTER t = {})`,
 REPEAT GEN_TAC THEN STRIP_TAC THEN
 FIRST_ASSUM(MP_TAC o MATCH_MP OUTSIDE_BOUNDED_NONEMPTY o
      MATCH_MP COMPACT_IMP_BOUNDED) THEN
 FIRST_X_ASSUM(MP_TAC o GEN_REWRITE_RULE I [GSYM MEMBER_NOT_EMPTY]) THEN
 REWRITE_TAC[GSYM MEMBER_NOT_EMPTY; LEFT_IMP_EXISTS_THM; IN_INTER] THEN
 X_GEN_TAC `b:real^N` THEN DISCH_TAC THEN
 X_GEN_TAC `a:real^N` THEN DISCH_TAC THEN
 ASM_CASES_TAC `(a:real^N) IN t` THENL [ASM_MESON_TAC[]; ALL_TAC] THEN
 MP_TAC(ISPECL [`linepath(a:real^N,b)`; `(:real^N) DIFF t`]
      EXISTS_PATH_SUBPATH_TO_FRONTIER) THEN
 REWRITE_TAC[PATH_LINEPATH; PATHSTART_LINEPATH; PATHFINISH_LINEPATH] THEN
 ASM_REWRITE_TAC[IN_DIFF; IN_UNIV; LEFT_IMP_EXISTS_THM] THEN
 X_GEN_TAC `g:real^1->real^N` THEN REWRITE_TAC[FRONTIER_COMPLEMENT] THEN
 REWRITE_TAC[PATH_IMAGE_LINEPATH; INTERIOR_DIFF; INTERIOR_UNIV] THEN
 ABBREV_TAC `c:real^N = pathfinish g` THEN STRIP_TAC THEN
 SUBGOAL_THEN `frontier t SUBSET (:real^N) DIFF s` MP_TAC THENL
  [ONCE_REWRITE_TAC[GSYM FRONTIER_COMPLEMENT] THEN
   REWRITE_TAC[frontier] THEN
   ASM_SIMP_TAC[CLOSURE_CLOSED; GSYM OPEN_CLOSED] THEN ASM SET_TAC[];
   REWRITE_TAC[SUBSET; IN_DIFF; IN_UNIV]] THEN
 DISCH_THEN(MP_TAC o SPEC `c:real^N`) THEN ASM_REWRITE_TAC[] THEN
 DISCH_TAC THEN MP_TAC(ISPEC `(:real^N) DIFF s` OPEN_CONTAINS_CBALL) THEN
 ASM_SIMP_TAC[GSYM closed; COMPACT_IMP_CLOSED; IN_DIFF; IN_UNIV] THEN
 DISCH_THEN(MP_TAC o SPEC `c:real^N`) THEN ASM_REWRITE_TAC[] THEN
 DISCH_THEN(X_CHOOSE_THEN `e:real` STRIP_ASSUME_TAC) THEN
 MP_TAC(ISPECL [`c:real^N`; `t:real^N->bool`]
      CLOSURE_APPROACHABLE) THEN
 RULE_ASSUM_TAC(REWRITE_RULE[frontier; IN_DIFF]) THEN
 ASM_REWRITE_TAC[] THEN
 DISCH_THEN(MP_TAC o SPEC `e:real`) THEN ASM_REWRITE_TAC[] THEN
 MATCH_MP_TAC MONO_EXISTS THEN X_GEN_TAC `d:real^N` THEN
 STRIP_TAC THEN ASM_REWRITE_TAC[] THEN
 MATCH_MP_TAC OUTSIDE_SAME_COMPONENT THEN
 EXISTS_TAC `a:real^N` THEN ASM_REWRITE_TAC[] THEN
 REWRITE_TAC[connected_component] THEN
 EXISTS_TAC `path_image(g) UNION segment[c:real^N,d]` THEN
 REWRITE_TAC[IN_UNION; ENDS_IN_SEGMENT] THEN CONJ_TAC THENL
  [MATCH_MP_TAC CONNECTED_UNION THEN
   ASM_SIMP_TAC[CONNECTED_SEGMENT; GSYM MEMBER_NOT_EMPTY;
                CONNECTED_PATH_IMAGE] THEN
   EXISTS_TAC `c:real^N` THEN REWRITE_TAC[ENDS_IN_SEGMENT; IN_INTER] THEN
   ASM_MESON_TAC[PATHFINISH_IN_PATH_IMAGE; SUBSET];
   CONJ_TAC THENL [ALL_TAC; ASM_MESON_TAC[PATHSTART_IN_PATH_IMAGE]] THEN
   REWRITE_TAC[UNION_SUBSET] THEN CONJ_TAC THENL
    [FIRST_X_ASSUM(MATCH_MP_TAC o MATCH_MP (SET_RULE
      `~(c IN s)
       ==> (t DELETE c) SUBSET (UNIV DIFF s)
           ==> t SUBSET (UNIV DIFF s)`)) THEN
     FIRST_X_ASSUM(MATCH_MP_TAC o MATCH_MP (REWRITE_RULE[IMP_CONJ]
      SUBSET_TRANS)) THEN
     SIMP_TAC[SET_RULE `UNIV DIFF s SUBSET UNIV DIFF t <=> t SUBSET s`] THEN
     ASM_MESON_TAC[SUBSET_TRANS; CLOSURE_SUBSET];
     FIRST_X_ASSUM(MATCH_MP_TAC o MATCH_MP (REWRITE_RULE[IMP_CONJ_ALT]
      SUBSET_TRANS)) THEN
     REWRITE_TAC[SEGMENT_CONVEX_HULL] THEN MATCH_MP_TAC HULL_MINIMAL THEN
     ASM_SIMP_TAC[CONVEX_CBALL; INSERT_SUBSET; REAL_LT_IMP_LE;
                  EMPTY_SUBSET; CENTRE_IN_CBALL] THEN
     REWRITE_TAC[IN_CBALL] THEN
     ASM_MESON_TAC[DIST_SYM; REAL_LT_IMP_LE]]]);;
```



```
lemma outside_compact_in_open:
  fixes s :: "'a :: {real_normed_vector,perfect_space} set"
  assumes s: "compact s" and t: "open t" and "s ⊆ t" "t ≠ {}"
    shows "outside s ∩ t ≠ {}"
proof -
 have "outside s ≠ {}"
  by (simp add: compact_imp_bounded outside_bounded_nonempty s)
 with assms obtain a b where a: "a ∈ outside s" and b: "b ∈ t" by auto
 show ?thesis
 proof (cases "a ∈ t")
  case True with a show ?thesis by blast
 next
  case False
   have front: "frontier t ⊆ - s"
    using `s ⊆ t` frontier_disjoint_eq t by auto
   { fix γ
    assume "path γ" and pimg_sbs: "path_image γ - {pathfinish γ} ⊆ interior (- t)"
       and pf: "pathfinish γ ∈ frontier t" and ps: "pathstart γ = a"
    def c ≡ "pathfinish γ"
    have "c ∈ -s" unfolding c_def using front pf by blast
    moreover have "open (-s)" using s compact_imp_closed by blast
    ultimately obtain ε::real where "ε > 0" and ε: "cball c ε ⊆ -s"
     using open_contains_cball[of "-s"] s by blast
    then obtain d where "d ∈ t" and d: "dist d c < ε"
     using closure_approachable [of c t] pf unfolding c_def
     by (metis Diff_iff frontier_def)
    then have "d ∈ -s" using ε
     using dist_commute by (metis contra_subsetD mem_cball not_le not_less_iff_gr_or_eq)
    have pimg_sbs_cos: "path_image γ ⊆ -s"
     using pimg_sbs apply (auto simp: path_image_def)
     apply (drule subsetD)
     using `c ∈ - s` `s ⊆ t` interior_subset apply (auto simp: c_def)
     done
    have "closed_segment c d ≤ cball c ε"
     apply (simp add: segment_convex_hull)
     apply (rule hull_minimal)
     using  `ε > 0` d apply (auto simp: dist_commute)
     done
    with ε have "closed_segment c d ≤ -s" by blast
    moreover have con_gcd: "connected (path_image γ ∪ closed_segment c d)"
     by (rule connected_Un) (auto simp: c_def `path γ` connected_path_image)
    ultimately have "connected_component (- s) a d"
     unfolding connected_component_def using pimg_sbs_cos ps by blast
    then have "outside s ∩ t ≠ {}"
     using outside_same_component [OF _ a]  by (metis IntI `d ∈ t` empty_iff)
   } note * = this
   have pal: "pathstart (linepath a b) ∈ closure (- t)"
    by (auto simp: False closure_def)
   show ?thesis
    by (rule exists_path_subpath_to_frontier [OF path_linepath pal _ *]) (auto simp: b)
 qed
qed
```

Scenario 2
Analysis in HOL Light and Isabelle/HOL

"Larry": **In HOL Light, and I'm willing to bet in Coq, proofs are almost never modified**. I have been working with a snapshot of HOL Light that is almost 2 years old. Yesterday I grabbed the latest snapshot. Although there are quite a few differences, virtually all of them consist of entire proofs added or deleted. Only a few proofs were modified, and only in trivial ways. Every formalisation decision is therefore a commitment. **With us it is very different.**

There were quite a few changes to HOL Light, my point is that these were all at the level of whole proofs. **Proofs were added and occasionally moved, but never modified**.

It's obvious that structured proofs should be easier to maintain, but we've never sought empirical justification.

Scenario 3
My KBO in Isabelle/HOL

"Heiko": As I have been accepted for the Graduate School of Computer Sciences Preparatory Phase, I would like to ask you **whether you have topics for Research Immersion Labs** to offer at the group for the Automation of Logic.

Scenario 3
My KBO in Isabelle/HOL

"Jasmin": I can supervise work on (1) **formalization**, (2) proof assistant implementation, and (3) automatic theorem provers (calculi and implementation). So you have plenty of languages to choose from (Isabelle/HOL, Standard ML, C, …).

Isabelle/Isar Proofs from ATP Proofs in Sledgehammer

Jasmin Christian Blanchette

Inria Nancy – Grand Est, France
Max-Planck-Institut für Informatik, Saarbrücken, Germany

Does there exist a function f from reals to reals such that for all x and y, $f(x + y^2) - f(x) \geq y$?

```
let lemma = prove
(`!f:real->real. ~(!x y. f(x + y * y) - f(x) >= y)`,
 REWRITE_TAC[real_ge] THEN REPEAT STRIP_TAC THEN
 SUBGOAL_THEN `!n x y. &n * y <= f(x + &n * y * y) - f(x)` MP_TAC THENL
  [MATCH_MP_TAC num_INDUCTION THEN SIMP_TAC[REAL_MUL_LZERO; REAL_ADD_RID] THEN
   REWRITE_TAC[REAL_SUB_REFL; REAL_LE_REFL; GSYM REAL_OF_NUM_SUC] THEN
   GEN_TAC THEN REPEAT(MATCH_MP_TAC MONO_FORALL THEN GEN_TAC) THEN
   FIRST_X_ASSUM(MP_TAC o SPECL [`x + &n * y * y`; `y:real`]) THEN
   SIMP_TAC[REAL_ADD_ASSOC; REAL_ADD_RDISTRIB; REAL_MUL_LID] THEN
   REAL_ARITH_TAC;
   X_CHOOSE_TAC `m:num` (SPEC `f(&1) - f(&0):real` REAL_ARCH_SIMPLE) THEN
   DISCH_THEN(MP_TAC o SPECL [`SUC m EXP 2`; `&0`; `inv(&(SUC m))`]) THEN
   REWRITE_TAC[REAL_ADD_LID; GSYM REAL_OF_NUM_SUC; GSYM REAL_OF_NUM_POW] THEN
   REWRITE_TAC[REAL_FIELD `(&m + &1) pow 2 * inv(&m + &1) = &m + &1`;
     REAL_FIELD `(&m + &1) pow 2 * inv(&m + &1) * inv(&m + &1) = &1`] THEN
   ASM_REAL_ARITH_TAC]);;
```

"John"

Does there exist a function f from reals to reals such that for all x and y, $f(x + y^2) - f(x) \geq y$?

[1] $f(x + y^2) - f(x) \geq y$ for any x and y (given)

[2] $f(x + n y^2) - f(x) \geq n y$ for any x, y, and natural number n
(by an easy induction using [1] for the step case)

[3] $f(1) - f(0) \geq m + 1$ for any natural number m
(set $n = (m + 1)^2$, $x = 0$, $y = 1/(m + 1)$ in [2])

[4] Contradiction of [3] and the Archimedean property of the reals

"John"

```
lemma
  shows "¬ (∃f :: real ⇒ real. ∀x y. f (x + y * y) - f x ≥ y)"
proof
  assume "∃f :: real ⇒ real. ∀x y. f (x + y * y) - f x ≥ y"
  then obtain f :: "real ⇒ real" where f: "∧x y. f (x + y * y) - f x ≥ y"
    by blast
  have nf: "∧(n :: nat) x y. f (x + real n * y * y) - f x ≥ real n * y"
  proof -
    fix n x y
    show "f (x + real n * y * y) - f x ≥ real n * y"
    proof (induct n)
      case 0 thus ?case by simp
    next
      case (Suc n) show ?case
      proof simp
        have "∃r. y ≤ f (y * y + (x + y * (y * real n))) - r ∧ y * real n ≤ r - f x"
        by (metis Suc.hyps add.commute f mult.commute)
      then have "y + y * real n ≤ f (y * y + (x + y * (y * real n))) - f x"
        by linarith
      then show "(1 + real n) * y ≤ f (x + (1 + real n) * y * y) - f x"
        by (simp add: add.left_commute distrib_left mult.commute)
      qed
    qed
  qed
  have min: "∧m. f 1 - f 0 ≥ real m + 1"
  proof -
    fix m
    show "f 1 - f 0 ≥ real m + 1"
    proof -
      have "∧r ra rb. (r :: real) / ra * rb = r * (rb / ra)"
        by simp
      then have "real (m + 1) * (real (m + 1) / real (m + 1)) ≤
          f (real (m + 1) * (real (m + 1) / (real (m + 1) * real (m + 1)))) - f 0"
        using nf[where n = "(m + 1) * (m + 1)" and x = 0 and y = "1 / (m + 1)"]
        by (metis (no_types) add.left_neutral divide_divide_eq_left mult.right_neutral of_nat_mult
          times_divide_eq_right)
      then have "real (m + 1) ≤ f 1 - f 0"
        by simp
      then show ?thesis
        by simp
    qed
  qed
  then show False
    by (metis add.commute add_le_imp_le_diff add_le_same_cancel2 add_mono diff_add_cancel
      ex_le_of_nat not_one_le_zero)
qed
```

intermediate properties

manual

generated automatically

Scratch.thy (modified)

Scratch.thy (~/)

```
lemma "A + B = (A #∪ B) + (A #∩ B)"
```

Proof state | Auto update | Update | Search: | 100%

```
proof (prove)
goal (1 subgoal):
 1. A + B = A #∪ B + A #∩ B
```

Output | Query | Sledgehammer | Symbols

Documentation | Sidekick | State | Theories

Scratch.thy (modified)

Scratch.thy (~/)

```
lemma "A + B = (A #∪ B) + (A #∩ B)"
sledgehammer
```

Proof state | Auto update | Update | Search: | 100%

```
Sledgehammering...
```

Output | Query | Sledgehammer | Symbols

Documentation | Sidekick | State | Theories

Scratch.thy (modified)

Scratch.thy (~/)

```
lemma "A + B = (A #∪ B) + (A #∩ B)"
sledgehammer
```

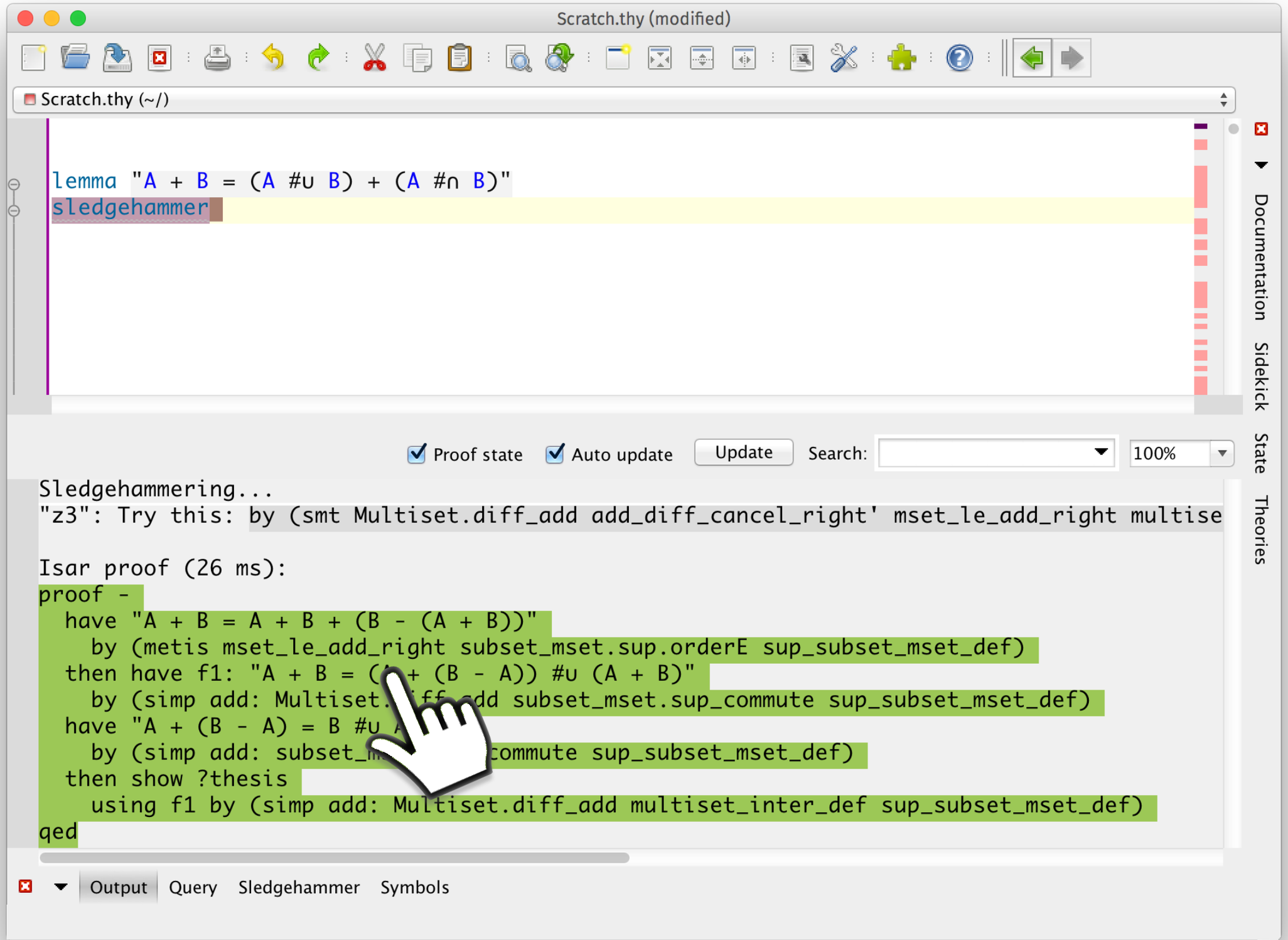
Proof state | Auto update | Update | Search: | 100%

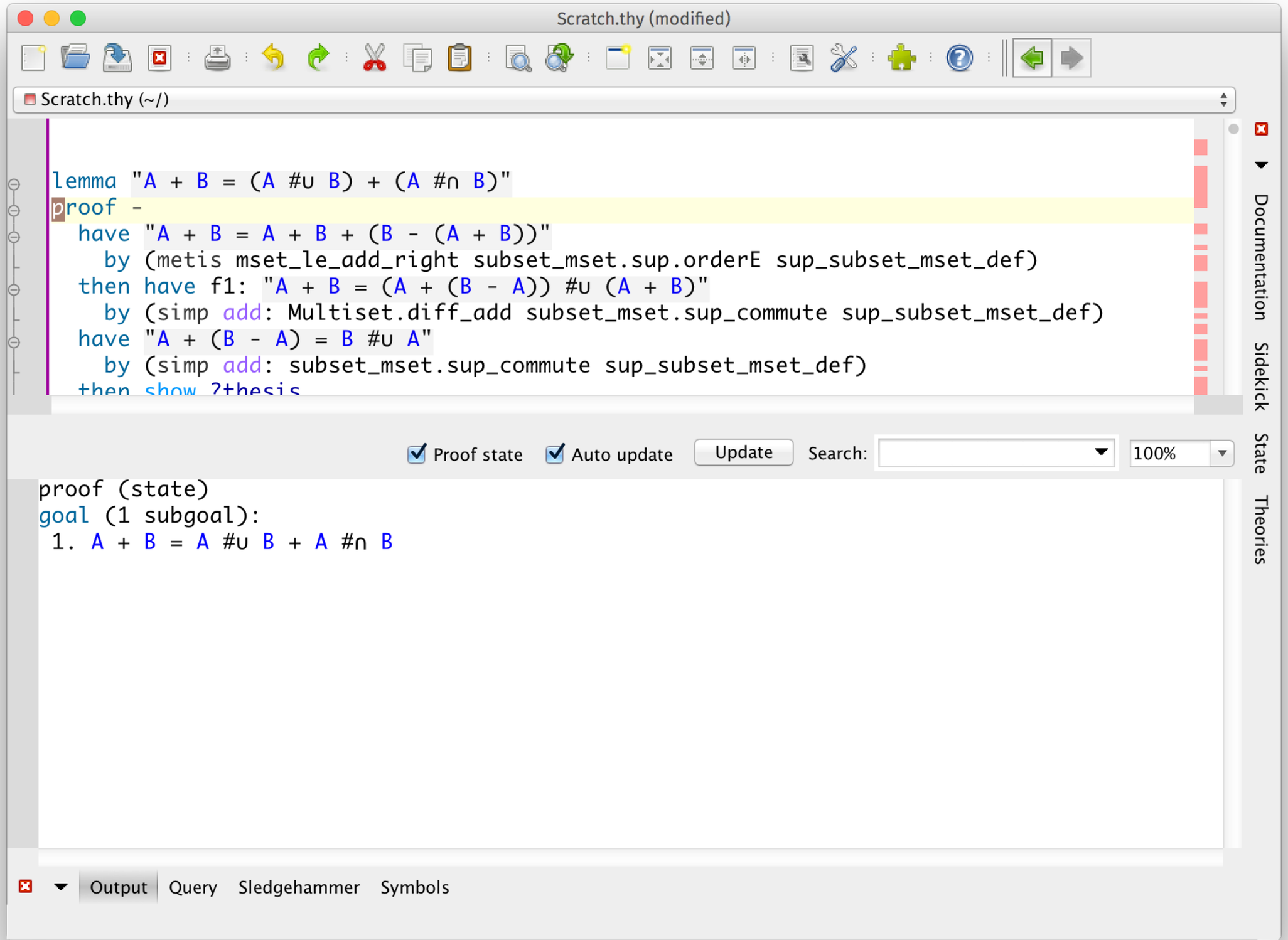
```
Sledgehammering...
"z3": Try this: by (smt Multiset.diff_add add_diff_cancel_right' mset_le_add_right multise

Isar proof (26 ms):
proof -
  have "A + B = A + B + (B - (A + B))"
    by (metis mset_le_add_right subset_mset.sup.orderE sup_subset_mset_def)
  then have f1: "A + B = (A + (B - A)) #∪ (A + B)"
    by (simp add: Multiset.diff_add subset_mset.sup_commute sup_subset_mset_def)
  have "A + (B - A) = B #∪ A"
    by (simp add: subset_mset.sup_commute sup_subset_mset_def)
  then show ?thesis
    using f1 by (simp add: Multiset.diff_add multiset_inter_def sup_subset_mset_def)
qed
```

Output | Query | Sledgehammer | Symbols

Documentation | Sidekick | State | Theories


Scratch.thy (modified)
Scratch.thy (~/)
lemma "A + B = (A #∪ B) + (A #∩ B)"
sledgehammer
Proof state
Auto update
Update
Search:
100%
Sledgehammering...
"z3": Try this: by (smt Multiset.diff_add add_diff_cancel_right' mset_le_add_right multise
Isar proof (26 ms):
proof -
have "A + B = A + B + (B - (A + B))"
by (metis mset_le_add_right subset_mset.sup.orderE sup_subset_mset_def)
then have f1: "A + B = (A + (B - A)) #∪ (A + B)"
by (simp add: Multiset.diff_add subset_mset.sup_commute sup_subset_mset_def)
have "A + (B - A) = B #∪
by (simp add: subset_m commute sup_subset_mset_def)
then show ?thesis
using f1 by (simp add: Multiset.diff_add multiset_inter_def sup_subset_mset_def)
qed
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
State
Theories


Scratch.thy (modified)
Scratch.thy (~/)
lemma "A + B = (A #∪ B) + (A #∩ B)"
proof -
have "A + B = A + B + (B - (A + B))"
by (metis mset_le_add_right subset_mset.sup.orderE sup_subset_mset_def)
then have f1: "A + B = (A + (B - A)) #∪ (A + B)"
by (simp add: Multiset.diff_add subset_mset.sup_commute sup_subset_mset_def)
have "A + (B - A) = B #∪ A"
by (simp add: subset_mset.sup_commute sup_subset_mset_def)
then show ?thesis
Proof state
Auto update
Update
Search:
100%
proof (state)
goal (1 subgoal):
1. A + B = A #∪ B + A #∩ B
Output
Query
Sledgehammer
Symbols
Documentation
Sidekick
State
Theories

Next:

Proof with holes

Higher-order reasoning (induction etc.)