Reconstructing MetiTarski Proofs in Isabelle/HOL

End-of-internship Presentation

Cristina Matache

AESTHETIC INTEGRATION

September 22, 2017

Outline

1. MetiTarski — The Problem

2. Approach
   - Translation
   - Proof Generation

3. Summary

MetiTarski

- Automatic theorem prover (ATP).
- Proves universally quantified inequalites involving:
  - polynomials
  - real-valued special functions: *log*, *exp*, *sin*, *cos*, *sqrt* etc.
- Using:
  - resolution
  - a decision procedure for the theory of real closed fields (RCF).
- Special functions are *approximated* by polynomials.

Motivation

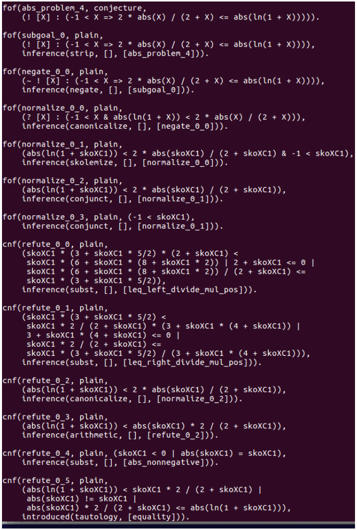
Long-term Goal

Use MetiTarski inside Imandra to solve geometric problems.

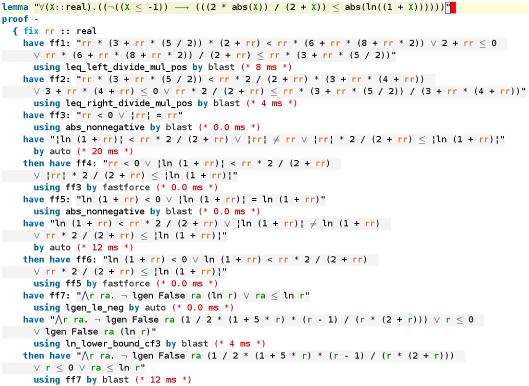
Why translate MetiTarski proofs to Isabelle proofs?

- No formal guarantee of correctness.
- Isabelle is more trustworthy.
- Integrate MetiTarski into Isabelle.

The Problem



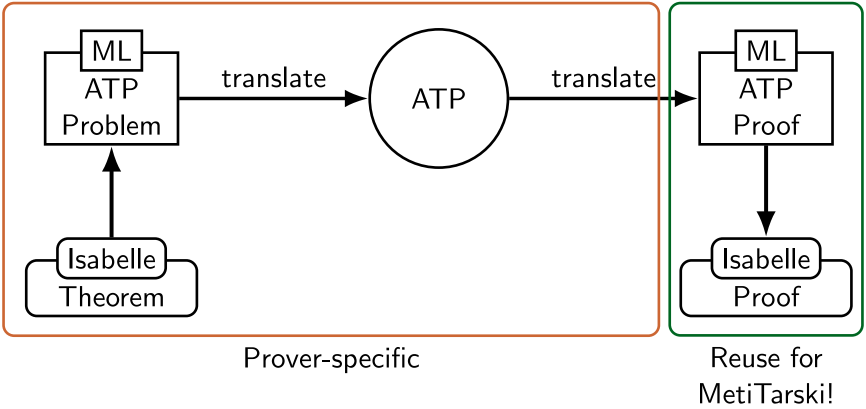
```
fof(abs_problem_4, conjecture,
    (! [X] : (-1 < X => 2 * abs(X) / (2 + X) <= abs(ln(1 + X))))).

fof(subgoal_0, plain,
    (! [X] : (-1 < X => 2 * abs(X) / (2 + X) <= abs(ln(1 + X)))),
    inference(strip, [], [abs_problem_4])).

fof(negate_0_0, plain,
    (~ ! [X] : (-1 < X => 2 * abs(X) / (2 + X) <= abs(ln(1 + X)))),
    inference(negate, [], [subgoal_0])).

fof(normalize_0_0, plain,
    (? [X] : (-1 < X & abs(ln(1 + X)) < 2 * abs(X) / (2 + X))),
    inference(canonicalize, [], [negate_0_0])).

fof(normalize_0_1, plain,
    (abs(ln(1 + skoXC1)) < 2 * abs(skoXC1) / (2 + skoXC1) & -1 < skoXC1),
    inference(skolemize, [], [normalize_0_0])).

fof(normalize_0_2, plain,
    (abs(ln(1 + skoXC1)) < 2 * abs(skoXC1) / (2 + skoXC1)),
    inference(conjunct, [], [normalize_0_1])).

fof(normalize_0_3, plain, (-1 < skoXC1),
    inference(conjunct, [], [normalize_0_1])).

cnf(refute_0_0, plain,
    (skoXC1 * (3 + skoXC1 * 5/2) * (2 + skoXC1) <
     skoXC1 * (6 + skoXC1 * (8 + skoXC1 * 2)) | 2 + skoXC1 <= 0 |
     skoXC1 * (6 + skoXC1 * (8 + skoXC1 * 2)) / (2 + skoXC1) <=
     skoXC1 * (3 + skoXC1 * 5/2)),
    inference(subst, [], [leq_left_divide_mul_pos])).

cnf(refute_0_1, plain,
    (skoXC1 * (3 + skoXC1 * 5/2) <
     skoXC1 * 2 / (2 + skoXC1) * (3 + skoXC1 * (4 + skoXC1)) |
     3 + skoXC1 * (4 + skoXC1) <= 0 |
     skoXC1 * 2 / (2 + skoXC1) <=
     skoXC1 * (3 + skoXC1 * 5/2) / (3 + skoXC1 * (4 + skoXC1))),
    inference(subst, [], [leq_right_divide_mul_pos])).

cnf(refute_0_2, plain,
    (abs(ln(1 + skoXC1)) < 2 * abs(skoXC1) / (2 + skoXC1)),
    inference(canonicalize, [], [normalize_0_2])).

cnf(refute_0_3, plain,
    (abs(ln(1 + skoXC1)) < abs(skoXC1) * 2 / (2 + skoXC1)),
    inference(arithmetic, [], [refute_0_2])).

cnf(refute_0_4, plain, (skoXC1 < 0 | abs(skoXC1) = skoXC1),
    inference(subst, [], [abs_nonnegative])).

cnf(refute_0_5, plain,
    (abs(ln(1 + skoXC1)) < skoXC1 * 2 / (2 + skoXC1) |
     abs(skoXC1) != skoXC1 |
     abs(skoXC1) * 2 / (2 + skoXC1) <= abs(ln(1 + skoXC1))),
    introduced(tautology, [equality])).
```



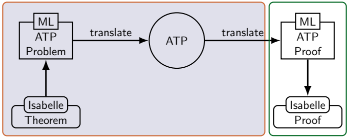
```
lemma "∀(X::real).((¬((X ≤ -1)) ⟶ (((2 * abs(X)) / (2 + X)) ≤ abs(ln((1 + X))))))"
proof -
  { fix rr :: real
    have ff1: "rr * (3 + rr * (5 / 2)) * (2 + rr) < rr * (6 + rr * (8 + rr * 2)) ∨ 2 + rr ≤ 0
      ∨ rr * (6 + rr * (8 + rr * 2)) / (2 + rr) ≤ rr * (3 + rr * (5 / 2))"
      using leq_left_divide_mul_pos by blast (* 8 ms *)
    have ff2: "rr * (3 + rr * (5 / 2)) < rr * 2 / (2 + rr) * (3 + rr * (4 + rr))
      ∨ 3 + rr * (4 + rr) ≤ 0 ∨ rr * 2 / (2 + rr) ≤ rr * (3 + rr * (5 / 2)) / (3 + rr * (4 + rr))"
      using leq_right_divide_mul_pos by blast (* 4 ms *)
    have ff3: "rr < 0 ∨ ¦rr¦ = rr"
      using abs_nonnegative by blast (* 0.0 ms *)
    have "¦ln (1 + rr)¦ < rr * 2 / (2 + rr) ∨ ¦rr¦ ≠ rr ∨ ¦rr¦ * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      by auto (* 20 ms *)
    then have ff4: "rr < 0 ∨ ¦ln (1 + rr)¦ < rr * 2 / (2 + rr)
      ∨ ¦rr¦ * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      using ff3 by fastforce (* 0.0 ms *)
    have ff5: "ln (1 + rr) < 0 ∨ ¦ln (1 + rr)¦ = ln (1 + rr)"
      using abs_nonnegative by blast (* 0.0 ms *)
    have "ln (1 + rr) < rr * 2 / (2 + rr) ∨ ¦ln (1 + rr)¦ ≠ ln (1 + rr)
      ∨ rr * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      by auto (* 12 ms *)
    then have ff6: "ln (1 + rr) < 0 ∨ ln (1 + rr) < rr * 2 / (2 + rr)
      ∨ rr * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      using ff5 by fastforce (* 0.0 ms *)
    have ff7: "⋀r ra. ¬ lgen False ra (ln r) ∨ ra ≤ ln r"
      using lgen_le_neg by auto (* 0.0 ms *)
    have "⋀r ra. ¬ lgen False ra (1 / 2 * (1 + 5 * r) * (r - 1) / (r * (2 + r))) ∨ r ≤ 0
      ∨ lgen False ra (ln r)"
      using ln_lower_bound_cf3 by blast (* 4 ms *)
    then have "⋀r ra. ¬ lgen False ra (1 / 2 * (1 + 5 * r) * (r - 1) / (r * (2 + r)))
      ∨ r ≤ 0 ∨ ra ≤ ln r"
      using ff7 by blast (* 12 ms *)
```

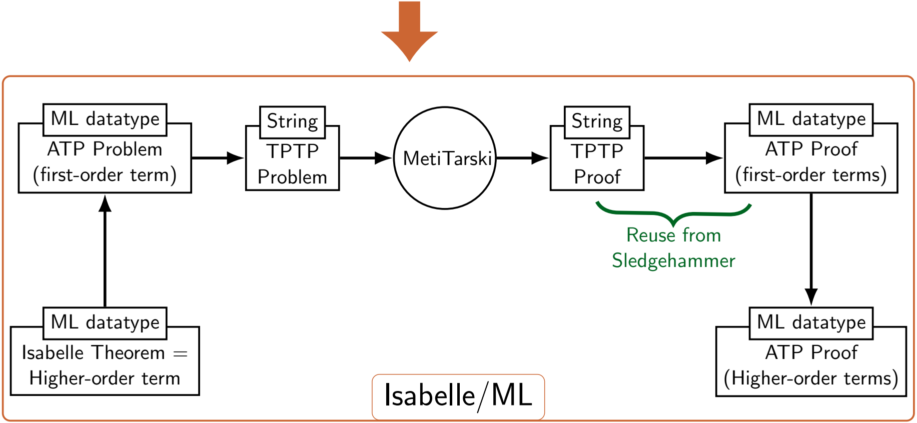
Sledgehammer

- Automatic proof tool in Isabelle.
- Sledgehammer operation:

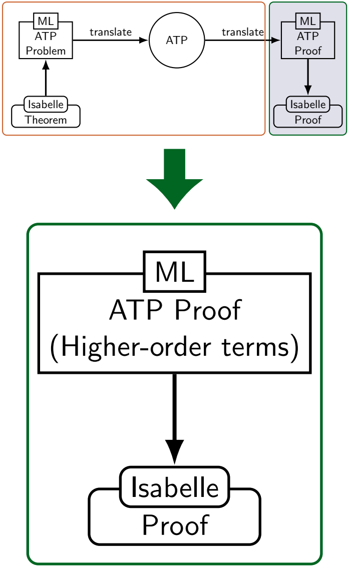


Translations





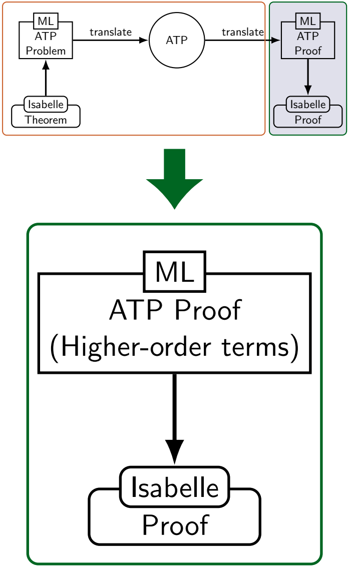
Generating Isabelle Proofs



Existing functionality:

- Proof redirection;
- Proof minimization;
- Proof preplay;
- Type annotations.

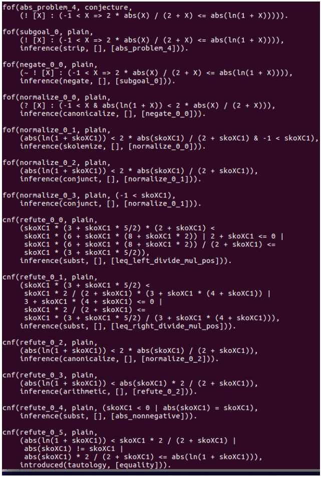
Generating Isabelle Proofs



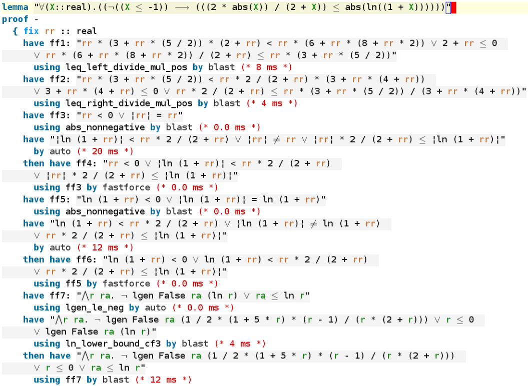
MetiTarski requirements:

- Proof methods for:
  - algebraic simplification;
  - the decision procedure.
- Correctness of special function bounds;

What we have so far



```
fof(abs_problem_4, conjecture,
    (! [X] : (-1 < X => 2 * abs(X) / (2 + X) <= abs(ln(1 + X))))).

fof(subgoal_0, plain,
    (! [X] : (-1 < X => 2 * abs(X) / (2 + X) <= abs(ln(1 + X)))),
    inference(strip, [], [abs_problem_4])).

fof(negate_0_0, plain,
    (~ ! [X] : (-1 < X => 2 * abs(X) / (2 + X) <= abs(ln(1 + X)))),
    inference(negate, [], [subgoal_0])).

fof(normalize_0_0, plain,
    (? [X] : (-1 < X & abs(ln(1 + X)) < 2 * abs(X) / (2 + X))),
    inference(canonicalize, [], [negate_0_0])).

fof(normalize_0_1, plain,
    (abs(ln(1 + skoXC1)) < 2 * abs(skoXC1) / (2 + skoXC1) & -1 < skoXC1),
    inference(skolemize, [], [normalize_0_0])).

fof(normalize_0_2, plain,
    (abs(ln(1 + skoXC1)) < 2 * abs(skoXC1) / (2 + skoXC1)),
    inference(conjunct, [], [normalize_0_1])).

fof(normalize_0_3, plain, (-1 < skoXC1),
    inference(conjunct, [], [normalize_0_1])).

cnf(refute_0_0, plain,
    (skoXC1 * (3 + skoXC1 * 5/2) * (2 + skoXC1) <
     skoXC1 * (6 + skoXC1 * (8 + skoXC1 * 2)) | 2 + skoXC1 <= 0 |
     skoXC1 * (6 + skoXC1 * (8 + skoXC1 * 2)) / (2 + skoXC1) <=
     skoXC1 * (3 + skoXC1 * 5/2)),
    inference(subst, [], [leq_left_divide_mul_pos])).

cnf(refute_0_1, plain,
    (skoXC1 * (3 + skoXC1 * 5/2) <
     skoXC1 * 2 / (2 + skoXC1) * (3 + skoXC1 * (4 + skoXC1)) |
     3 + skoXC1 * (4 + skoXC1) <= 0 |
     skoXC1 * 2 / (2 + skoXC1) <=
     skoXC1 * (3 + skoXC1 * 5/2) / (3 + skoXC1 * (4 + skoXC1))),
    inference(subst, [], [leq_right_divide_mul_pos])).

cnf(refute_0_2, plain,
    (abs(ln(1 + skoXC1)) < 2 * abs(skoXC1) / (2 + skoXC1)),
    inference(canonicalize, [], [normalize_0_2])).

cnf(refute_0_3, plain,
    (abs(ln(1 + skoXC1)) < abs(skoXC1) * 2 / (2 + skoXC1)),
    inference(arithmetic, [], [refute_0_2])).

cnf(refute_0_4, plain, (skoXC1 < 0 | abs(skoXC1) = skoXC1),
    inference(subst, [], [abs_nonnegative])).

cnf(refute_0_5, plain,
    (abs(ln(1 + skoXC1)) < skoXC1 * 2 / (2 + skoXC1) |
     abs(skoXC1) != skoXC1 |
     abs(skoXC1) * 2 / (2 + skoXC1) <= abs(ln(1 + skoXC1))),
    introduced(tautology, [equality])).
```

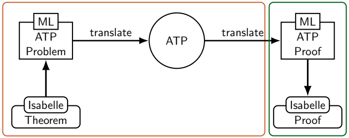


```
lemma "∀(X::real).((¬((X ≤ -1)) ⟶ (((2 * abs(X)) / (2 + X)) ≤ abs(ln((1 + X))))))"
proof -
  { fix rr :: real
    have ff1: "rr * (3 + rr * (5 / 2)) * (2 + rr) < rr * (6 + rr * (8 + rr * 2)) ∨ 2 + rr ≤ 0
      ∨ rr * (6 + rr * (8 + rr * 2)) / (2 + rr) ≤ rr * (3 + rr * (5 / 2))"
      using leq_left_divide_mul_pos by blast (* 8 ms *)
    have ff2: "rr * (3 + rr * (5 / 2)) < rr * 2 / (2 + rr) * (3 + rr * (4 + rr))
      ∨ 3 + rr * (4 + rr) ≤ 0 ∨ rr * 2 / (2 + rr) ≤ rr * (3 + rr * (5 / 2)) / (3 + rr * (4 + rr))"
      using leq_right_divide_mul_pos by blast (* 4 ms *)
    have ff3: "rr < 0 ∨ ¦rr¦ = rr"
      using abs_nonnegative by blast (* 0.0 ms *)
    have "¦ln (1 + rr)¦ < rr * 2 / (2 + rr) ∨ ¦rr¦ ≠ rr ∨ ¦rr¦ * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      by auto (* 20 ms *)
    then have ff4: "rr < 0 ∨ ¦ln (1 + rr)¦ < rr * 2 / (2 + rr)
      ∨ ¦rr¦ * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      using ff3 by fastforce (* 0.0 ms *)
    have ff5: "ln (1 + rr) < 0 ∨ ¦ln (1 + rr)¦ = ln (1 + rr)"
      using abs_nonnegative by blast (* 0.0 ms *)
    have "ln (1 + rr) < rr * 2 / (2 + rr) ∨ ¦ln (1 + rr)¦ ≠ ln (1 + rr)
      ∨ rr * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      by auto (* 12 ms *)
    then have ff6: "ln (1 + rr) < 0 ∨ ln (1 + rr) < rr * 2 / (2 + rr)
      ∨ rr * 2 / (2 + rr) ≤ ¦ln (1 + rr)¦"
      using ff5 by fastforce (* 0.0 ms *)
    have ff7: "⋀r ra. ¬ lgen False ra (ln r) ∨ ra ≤ ln r"
      using lgen_le_neg by auto (* 0.0 ms *)
    have "⋀r ra. ¬ lgen False ra (1 / 2 * (1 + 5 * r) * (r - 1) / (r * (2 + r))) ∨ r ≤ 0
      ∨ lgen False ra (ln r)"
      using ln_lower_bound_cf3 by blast (* 4 ms *)
    then have "⋀r ra. ¬ lgen False ra (1 / 2 * (1 + 5 * r) * (r - 1) / (r * (2 + r)))
      ∨ r ≤ 0 ∨ ra ≤ ln r"
      using ff7 by blast (* 12 ms *)
```

But some proof methods are still missing!

Summary

- Translate MetiTarski proofs to Isabelle.
- Reuse part of the Sledgehammer code.
- Implement the prover-specific part.



- Provide appropriate proof methods.

Still to do

- Finish proof method for algebraic simplification.
- Use the formalisation of the decision procedure.
- Prove more bounds.
- Thoroughly test the proof reconstruction.