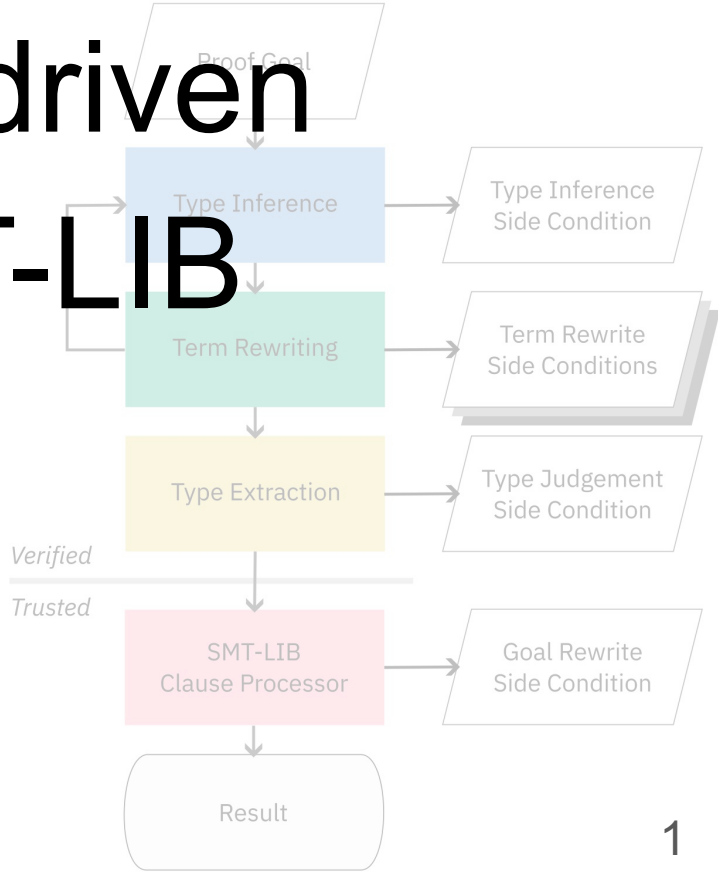
THE UNIVERSITY OF BRITISH COLUMBIA
Computer Science
Faculty of Science

Smtlink 3: Rewrite-driven Compilation to SMT-LIB

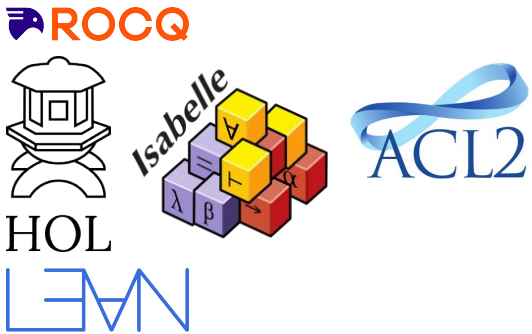
Chris Chen, Mark Greenstreet, and Margo Seltzer
The University of British Columbia
Computer Science



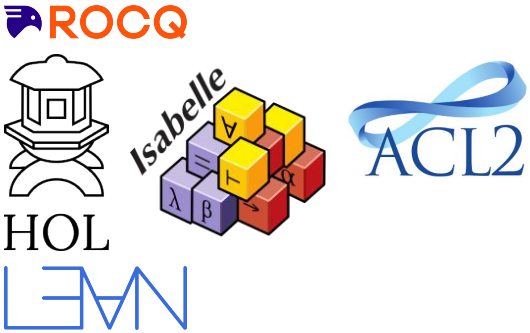
ACL2 2026 Workshop
16 November 2026

Interactive theorem provers

Satisfiability modulo theories (SMT) solvers







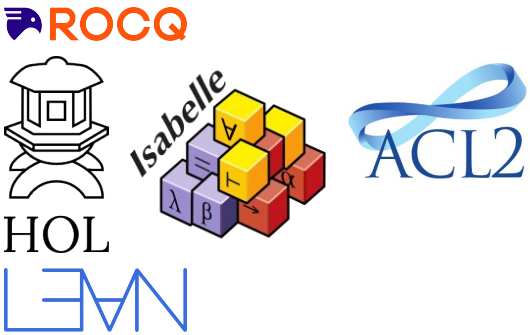


Proof engineering

Undecidable theories

Reflection


ROCQ
HOL
Isabelle
ACL2
Z3
CVC5
Constraint solving
Theory combination
~Decidable theories





Explicit Structure

e.g. structural induction

Implicit Structure

e.g. clause learning

Questions

Is the translation correct?

Is the SMT solver trustworthy?

Questions

Is the translation correct?

Is the SMT solver trustworthy?

Related work: proof reconstruction of external witness

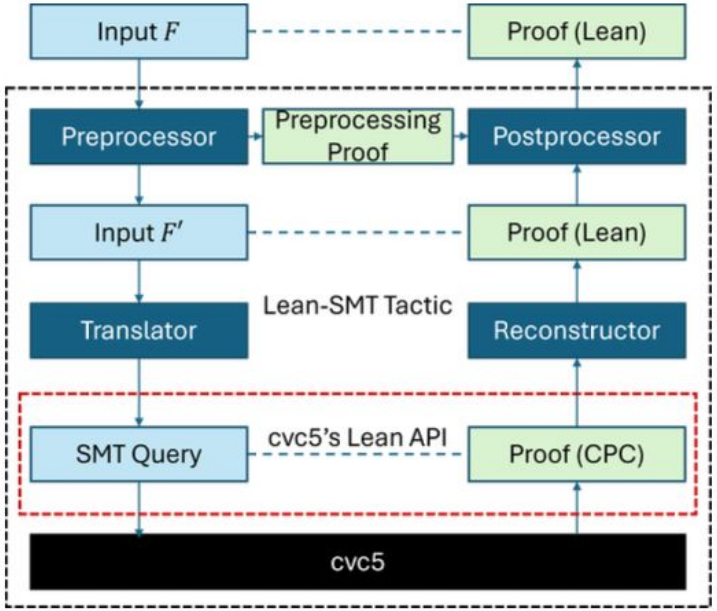


Fig. 1: Architecture of the LEAN-SMT tactic.

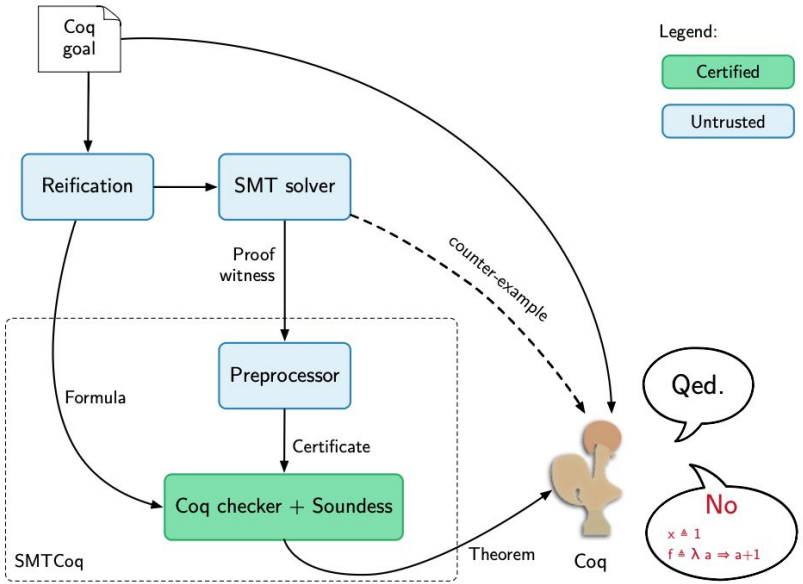
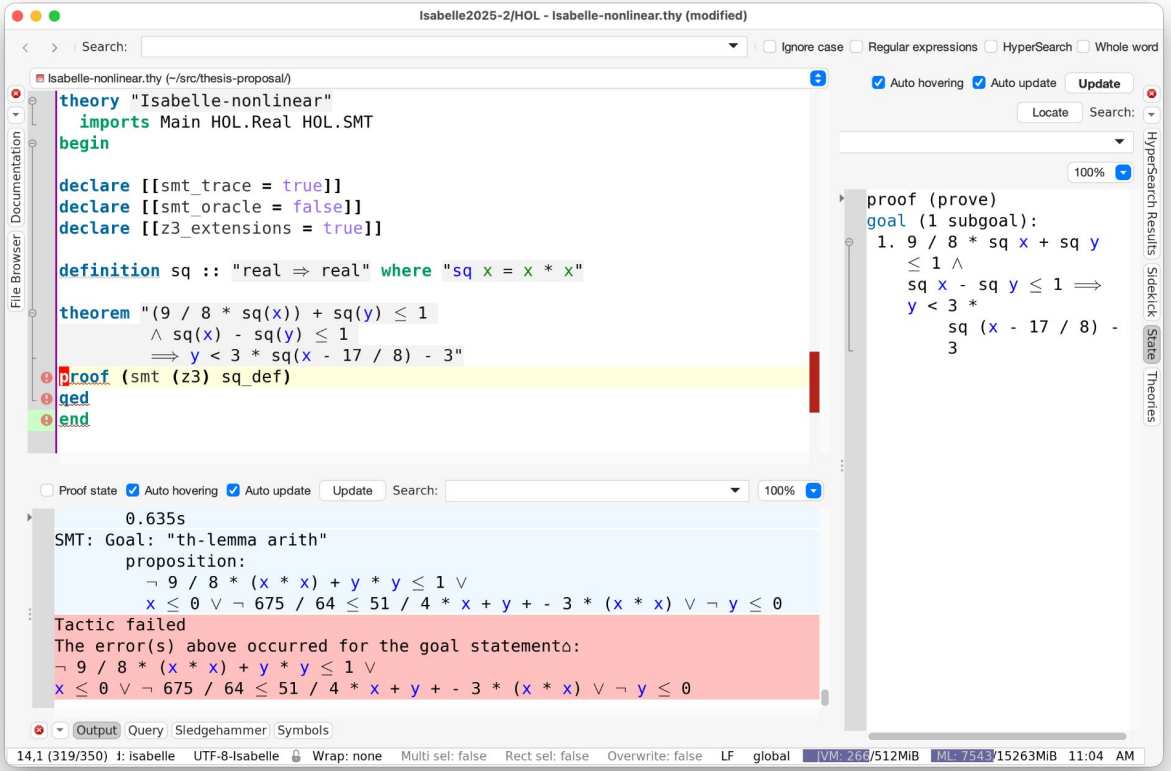
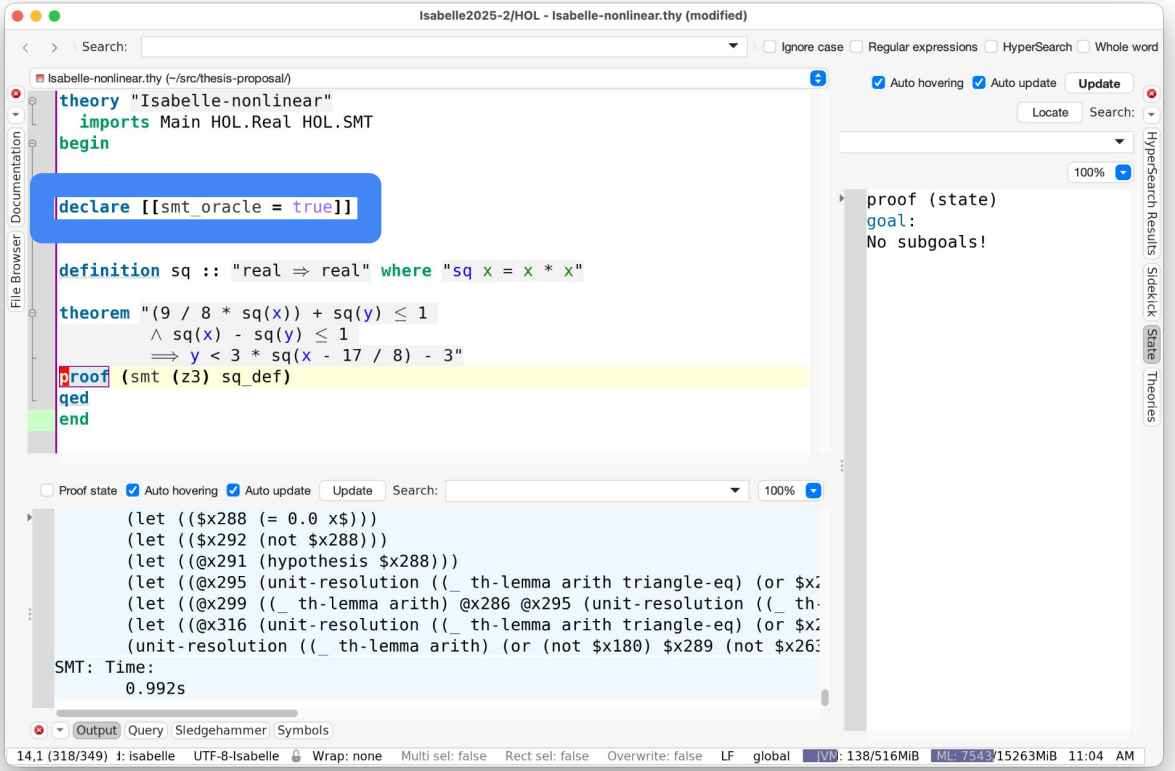


Figure 1: SMTCoq's architecture

Reconstruction is not always feasible



Reconstruction is not always feasible



One big question

Is the translation correct?

This is orthogonal

Is the SMT solver trustworthy?

The overarching Smtlink story

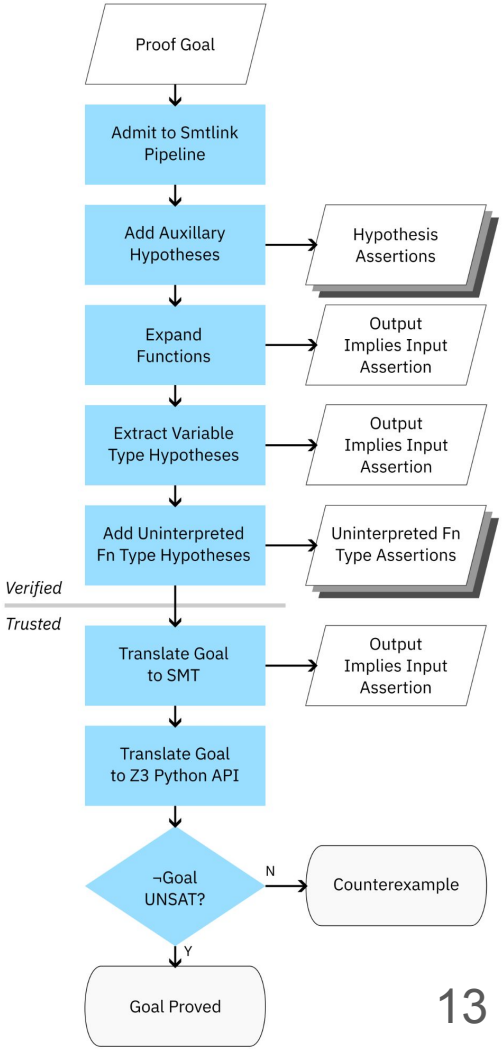
We can use ACL2's metaprogramming and metareasoning facilities to turn the theorem prover into a validity-preserving term compiler?

Smtlink 2.0

Multi-phase ACL2 to Z3 compiler

Certifying front-end

Partially-certifying back-end

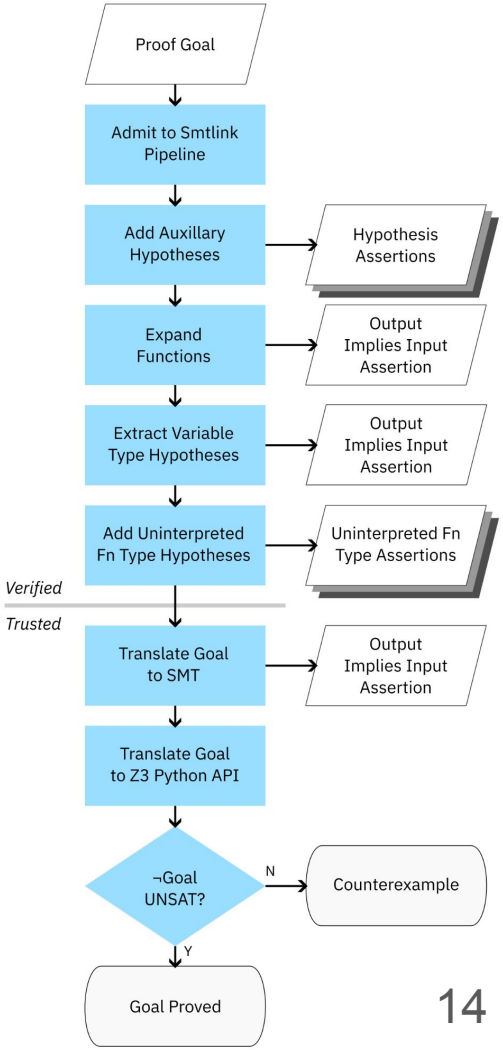


Smtlink 2.0

Required explicit type hypotheses

Hardcoded translation in the trusted clause processor

Python & Z3 semantics



Smtlink 2.0 + Type Inference

Type Inference Using Meta-extract for Smtlink and Beyond (ACL2 2020)
Peng, Greenstreet, doi: 10.4204/EPTCS.327.7

Typed terms

Defprod of pseudo-terms

Verified type inference and translation

Clause processors apply meta-extracted theorems

Smtlink 2.0 + Type Inference

We completed the type inference proofs 👍🏽

Translation was still an open problem 🤔

First idea:

Platonic ACL2 theories normalize.

Translating to SMT-LIB can be seen as normalization.

Given a normalizing translation theory, can we use the ACL2 rewriter in place of a custom translator?

Second idea:

Type reasoning is already done in the ACL2 rewriter.

Can we use the ACL2 rewriter to generate type judgements, in place of a custom type inference procedure?

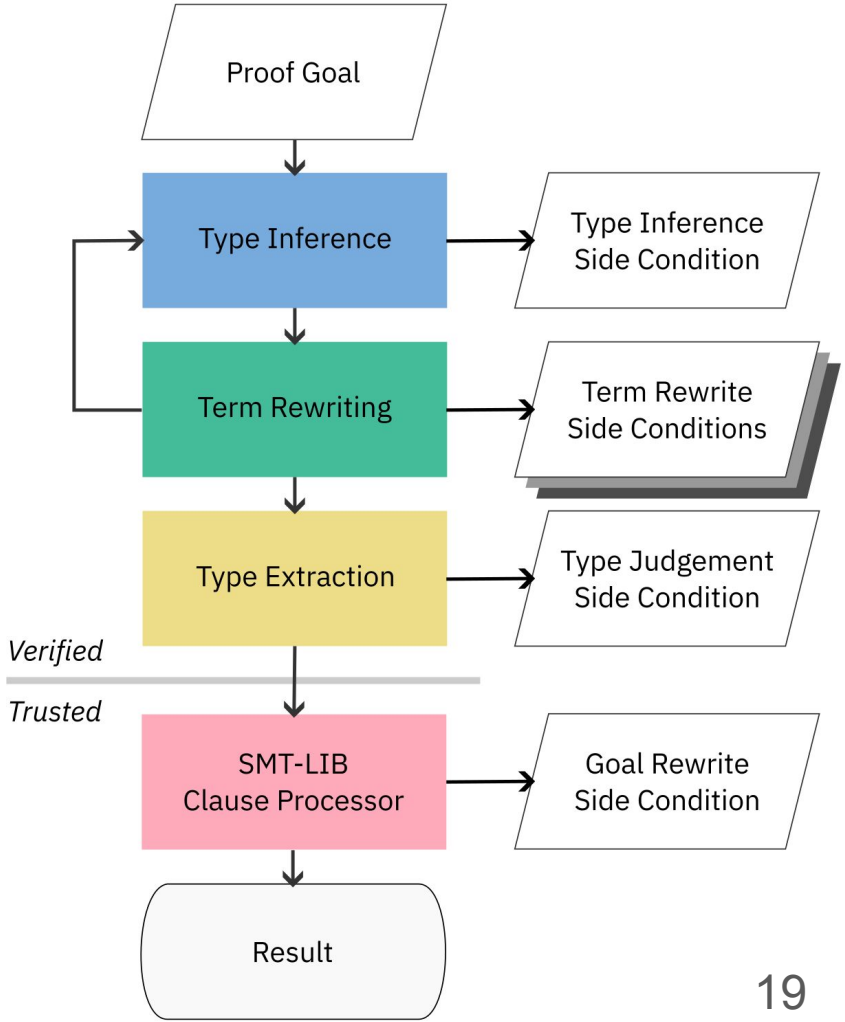
Smtlink 3

Verified type inference

Verified term rewriting

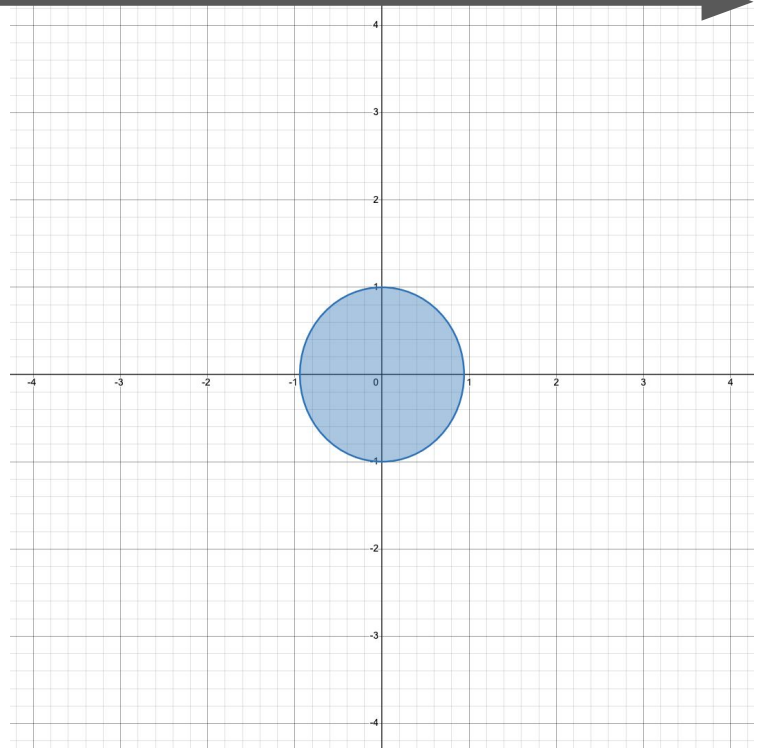
Targets SMT-LIB

Smaller trusted code base (84%)



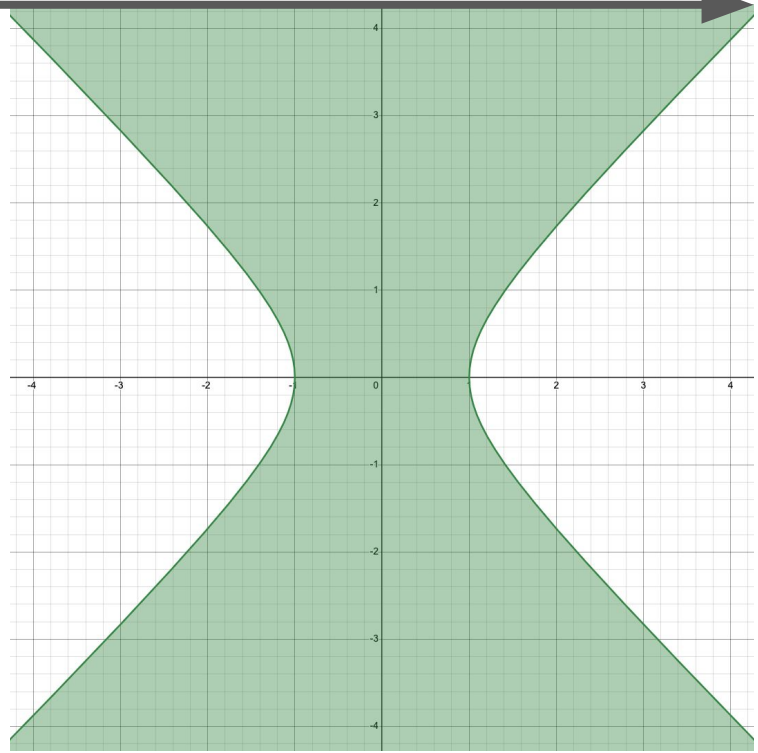
Example

$$x^2 - y^2 \leq 1$$



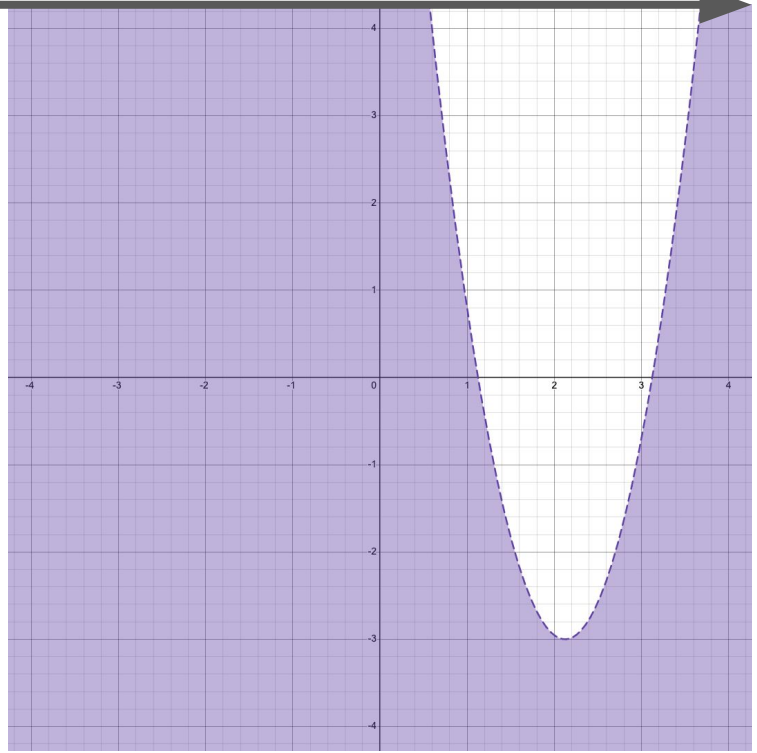
Example

$$\frac{9}{8}x^2 + y^2 \leq 1$$



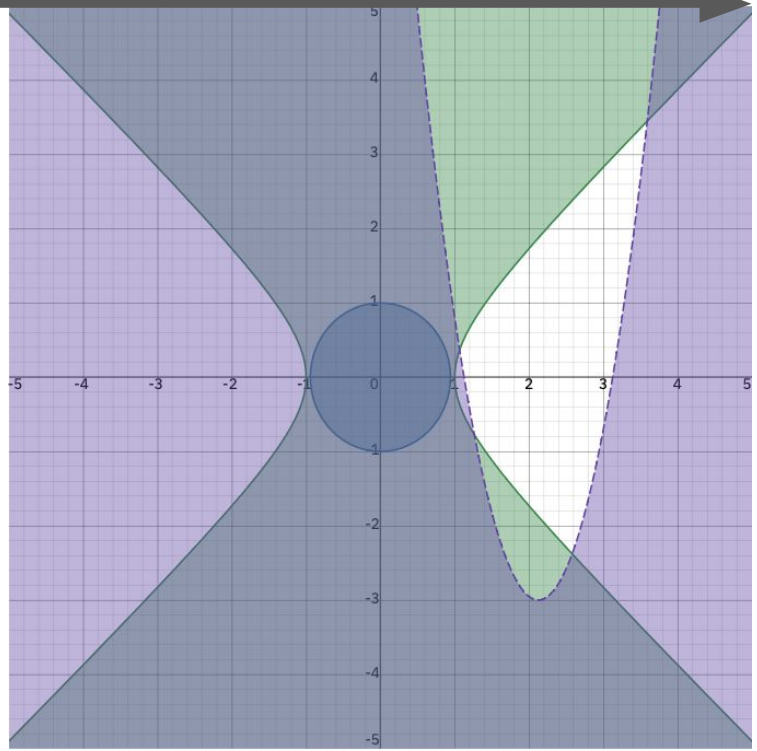
Example

$$y < 3\left(x - \frac{17}{8}\right)^2 - 3$$

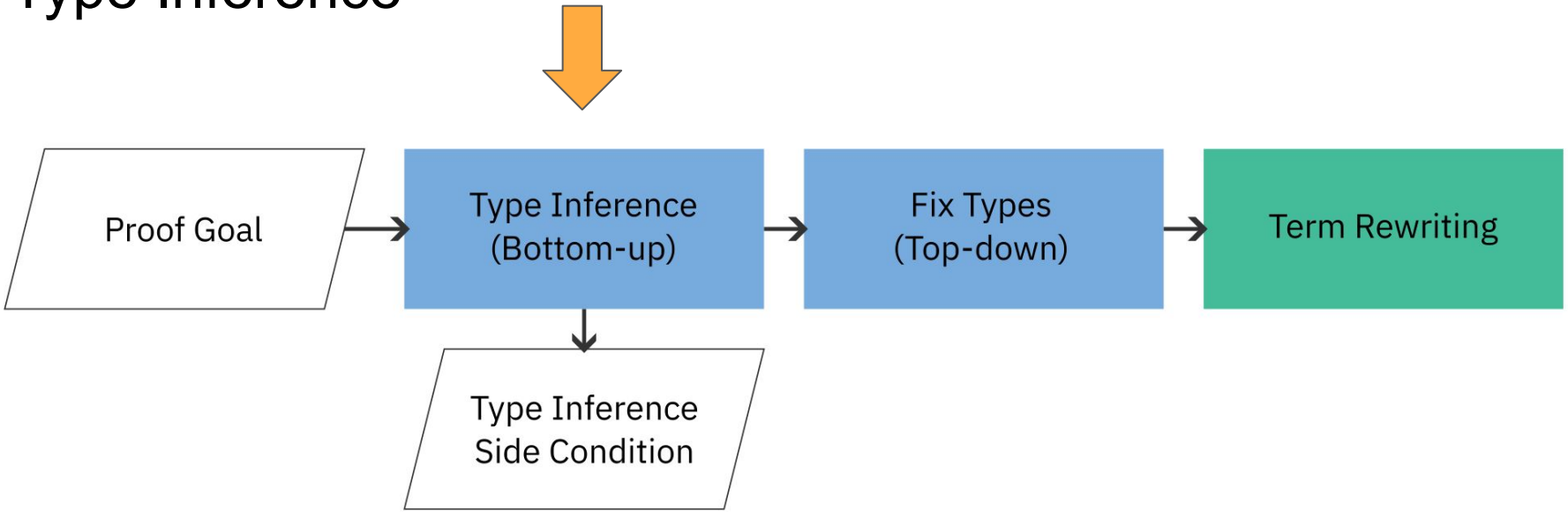


Example

```
(implies
 (and (real/rationalp x)
      (real/rationalp y)
      (<= (+ (* (/ 9 8) x x)
             (* y y))
          1)
      (<= (x^2-y^2 x y) 1))
 (< y
    (- (* 3
          (- x (/ 17 8))
          (- x (/ 17 8)))
       3)))
```



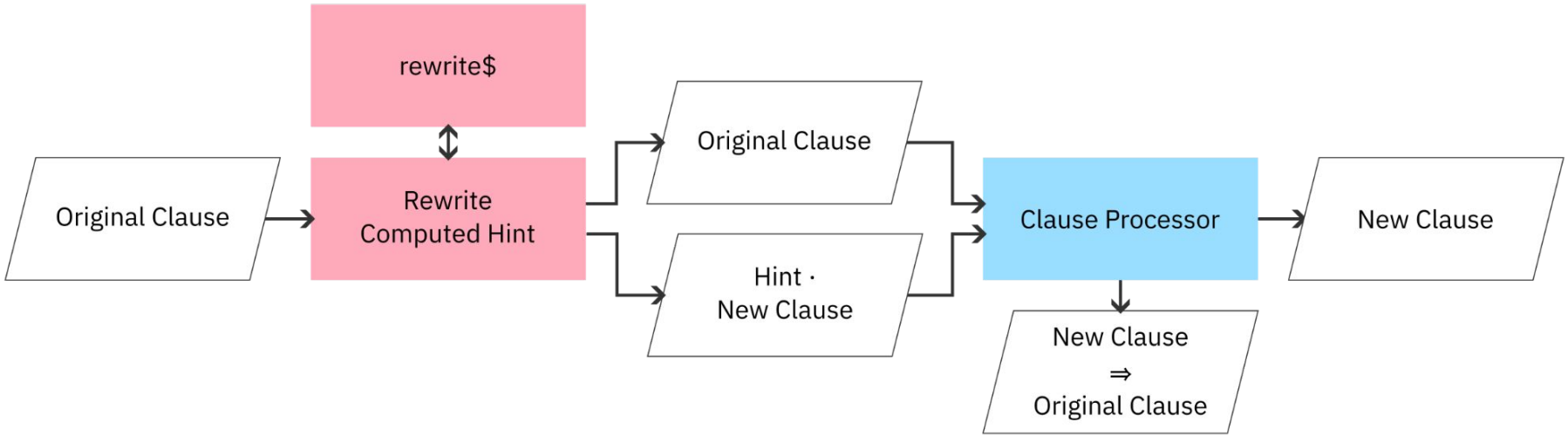
Type Inference



Type Inference Schema

```
(implies
 x.path-cond
 (and (ti-equal (hide (type-recognizer-1 x))
                (type-recognizer-1 x))
      (ti-equal (hide (type-recognizer-2 x))
                (type-recognizer-2 x))
      ...))
```

Smtlink 3 Rewrite Computed Hint



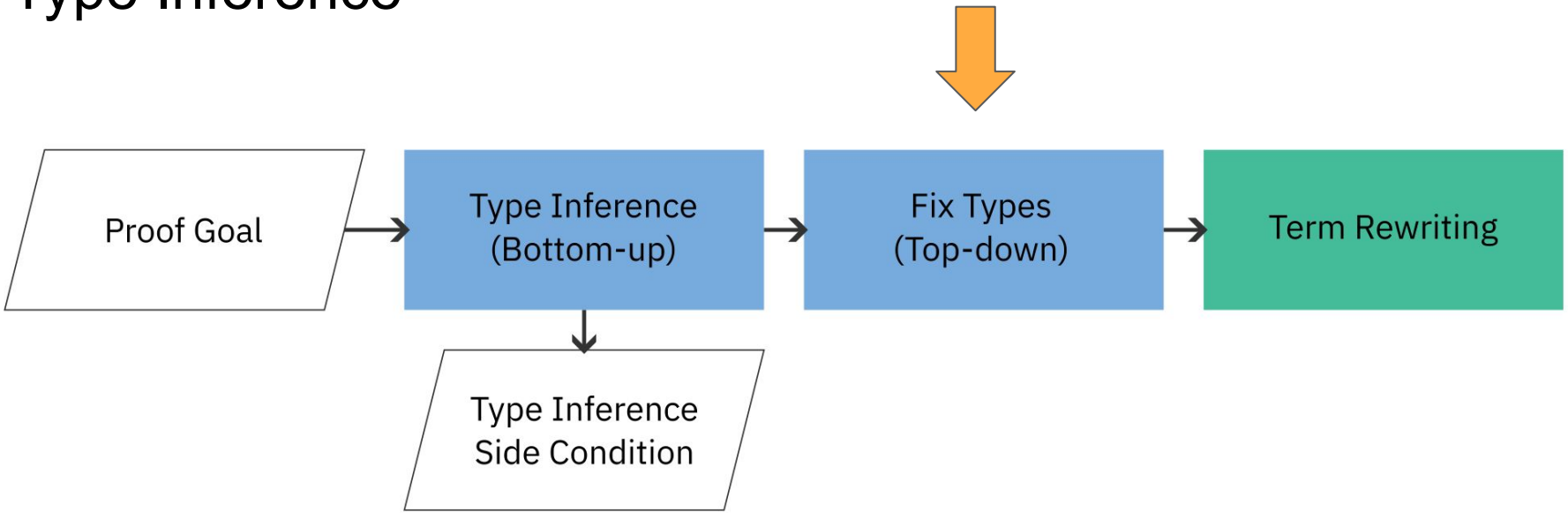
Type Inference of '3

```
((smt::path-cond)
 (smt::judgements (integerp smt::x)
                  (rationalp smt::x))
 (smt::smt-judgements)
 (smt::guts :quote 3))
```

Type Correctness Condition of '3

```
(if (if 't
        (if ((lambda (smt::x)
               (if (integerp smt::x)
                   (if (rationalp smt::x) 't 'nil)
                 'nil))
             '3)
            't
          'nil)
      't)
    (if (if 't
            (if ((lambda (smt::x) 't) '3) 't 'nil)
          't)
        't
      'nil)...)
```

Type Inference



Narrows scope for term rewriting

Fixing the Type of '3

```
((smt::path-cond)
 (smt::judgements (integerp smt::x)
                  (rationalp smt::x))
 (smt::smt-judgements (rationalp smt::x))
 (smt::guts :quote 3))
```

Fixing the Type of '3

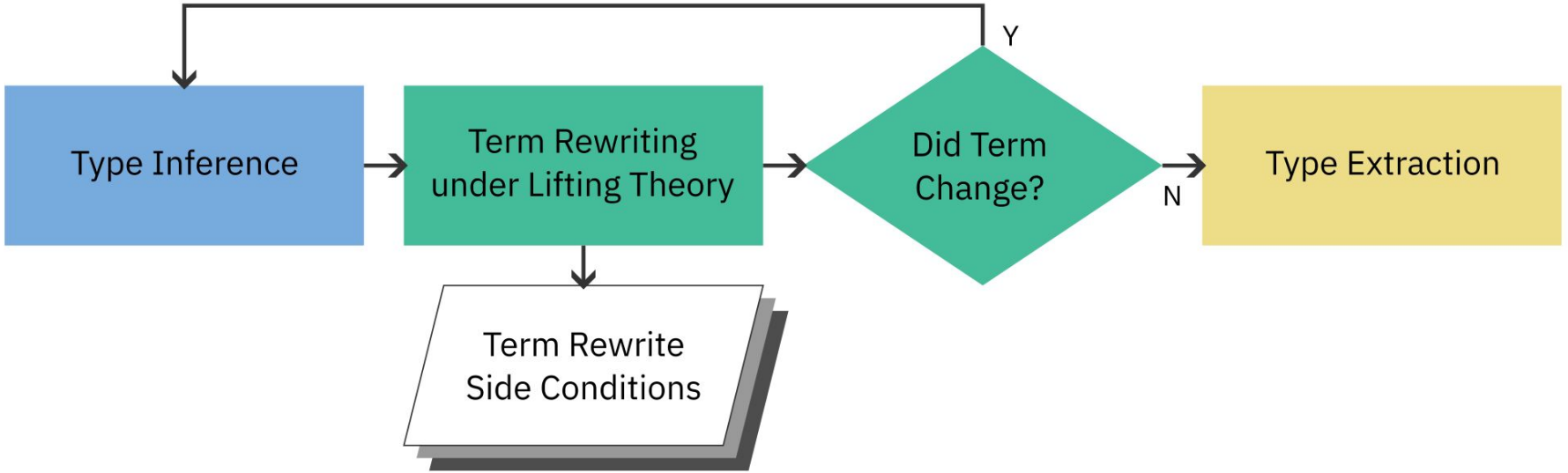
```
((smt::path-cond)
 (smt::judgements (integerp smt::x)
                  (rationalp smt::x))
 (smt::smt-judgements (rationalp smt::x))
 (smt::guts :quote 3))
```

Invariant: oset subset

Updated Type Correctness Condition of '3

```
(if (if 't
        (if ((lambda (smt::x)
               (if (integerp smt::x)
                   (if (rationalp smt::x) 't 'nil)
                 'nil))
             '3)
            't
          'nil)
      't)
    (if (if 't
            (if ((lambda (smt::x)
                   (if (rationalp smt::x) 't 'nil))
                 '3)
                't...)...)...)...)
```

Smtlink 3 Term Rewriting



A Clique of Constrained Functions for each SMT Theory

```
(encapsulate
    (((rational-+ * *) => *)
     ((rational-- *) => *)
     ((rational-* * *) => *)
     ((rational-/ *) => *)
     ((rational-< * *) => *))

  (local
    (defun rational-+ (x y)
      (binary-+ x y)))
...
```

A Clique of Constrained Functions for each SMT Theory

```
(defthm return-of-rational-+
  (implies (and (rationalp x)
                (rationalp y))
           (booleanp (rational-+ x y))))

(defthmd specialize-binary-rational-operators
  (implies (and (rationalp x)
                (rationalp y))
           (and (equal (binary-+ x y)
                       (rational-+ x y))
                (equal (binary-* x y)
                       (rational-* x y))
                ...)))
...
```

A Clique of Constrained Functions for each SMT Theory

```
(defthm return-of-rational-+
  (implies (and (rationalp x)
                (rationalp y))
           (booleanp (rational-+ x y))))

(defthmd specialize-binary-rational-operators
  (implies (and (rationalp x)
                (rationalp y))
           (and (equal (binary-+ x y)
                       (rational-+ x y))
                (equal (binary-* x y)
                       (rational-* x y))
                ...)))
...
```

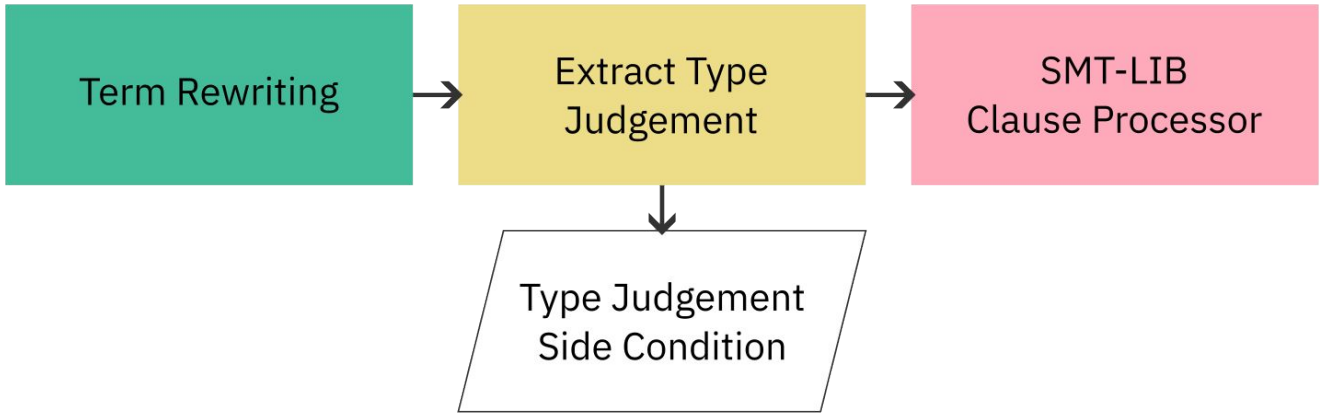
Smtlink Term Rewriting Theory Hint

```
(defthm poly-ineq-example
  (implies (and (real/rationalp x)
                (real/rationalp y)
                (<= (+ (* (/ 9 8) x x) (* y y)) 1)
                (<= (x^2-y^2 x y) 1))
           (< y
              (- (* 3 (- x (/ 17 8)) (- x (/ 17 8)))
                 3)))
  :hints (("Goal''"
       :smtlink
       (:translation-theory
        (union-theories
         (theory 'minimal-theory)
         '(smt::specialize-binary-rational-operators))))))
```

Smtlink Term Rewriting Result

```
Subgoal 2.6.2.6.2.6'
(implies
  (and (rationalp x)
       (rationalp y)
       (<= (smt::rational-+ (smt::rational-* y y)
                            (smt::rational-* 9/8
                              (smt::rational-* x x)))
           1)
       (<= (smt::rational-+ (smt::rational-* x x)
                            (smt::rational--
                              (smt::rational-* y y)))
           1))
  ...)
```

Type Extraction

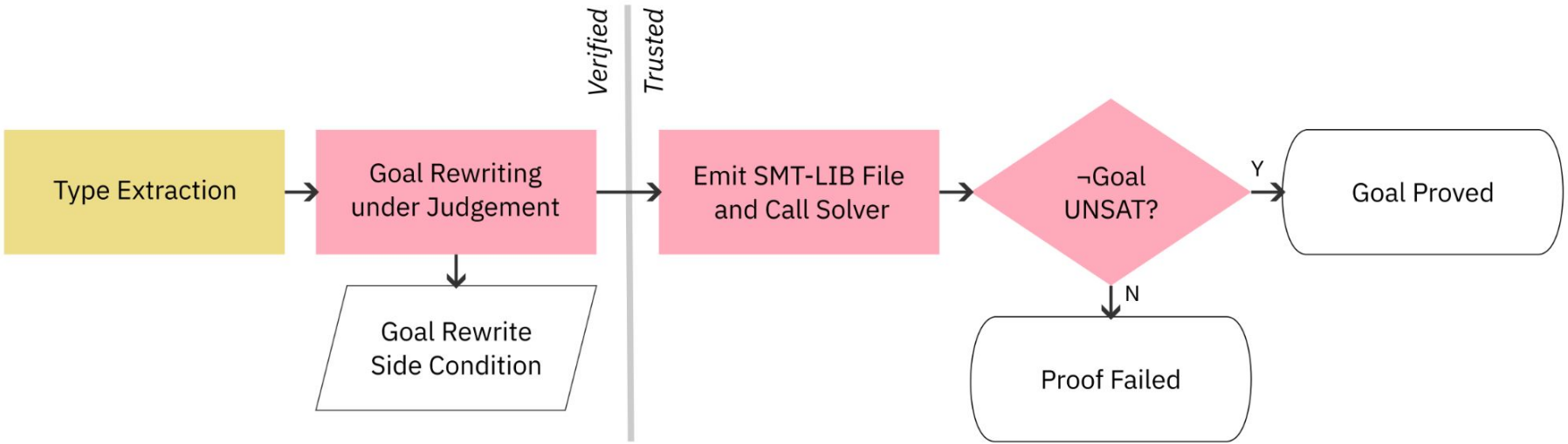


Clause-level type judgements

Variable and Function Judgements at Clause-level

```
((smt::vars ((smt::name . x)
             (smt::judgement . rationalp))
            ((smt::name . y)
             (smt::judgement . rationalp)))
 (smt::fns ((smt::name . smt::rational-*)
            (smt::domain rationalp rationalp)
            (smt::range . rationalp))
           ((smt::name . smt::rational-+)
            (smt::domain rationalp rationalp)
            (smt::range . rationalp))
           ...))
```

Smtlink 3 SMT-LIB Clause Processor

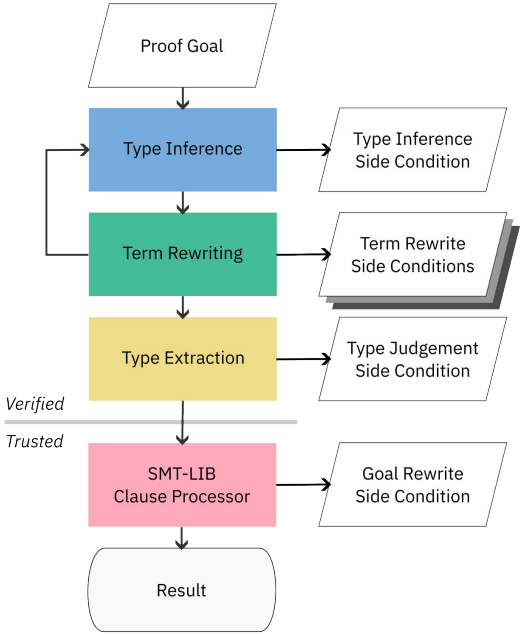


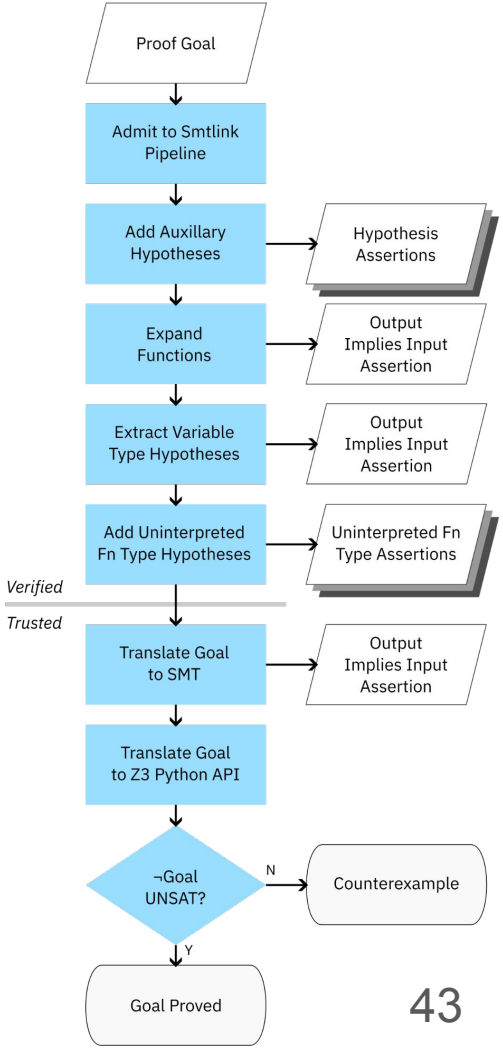
Smtlink 3 SMT-LIB Clause Processor

```
((|set-logic| AUFNIRA)
 ...
 (|declare-const| X |Real|)
 (|declare-const| Y |Real|)
 (|assert| (|not| (|ite| (< 1 (+ (* Y Y) (* (/ 9 8) (* X X))))
                         |true|
                         (|ite| (< 1 (+ (* X X) (- (* Y Y))))
                                |true|
                                (< Y
                                   (+ (/ 675 64)
                                      (+ (* 3 (* (/ (- 17) 8) X))
                                         (+ (* 3 (* (/ (- 17) 8) X))
                                            (* 3 (* X X))))))))))))
 (|check-sat|))
```

Other changes from Smtlink 2

Waterfall handles expansion, hypothesis instantiation





Smtlink 3 Supports

Real theory

Real+integer mixed theory

Alist to arrays with extensionality

User hypotheses

Future Work

Improved hint interface

SMT-LIB book

FTY support

More theories

Counterexample translation

Future Work

I will be looking for Future Work in about a year.

More Questions?

Our tutorial is Wednesday at 13:15