# Certified Interpolant Generation for EUF

*Andrew Reynolds*
*Cesare Tinelli*
*University of Iowa*

*Liana Hadarean*
*New York University*

*July 14, 2011*

# Overview

- Interpolation
  - Importance of certification
- Certified Interpolants via Proof Checking
  - Proof checker LFSC
  - Interpolant Generating Calculi
  - Generation of Interpolants via Type Inference
- Interpolation Calculus for EUF
- Results
- Conclusions and Future Work

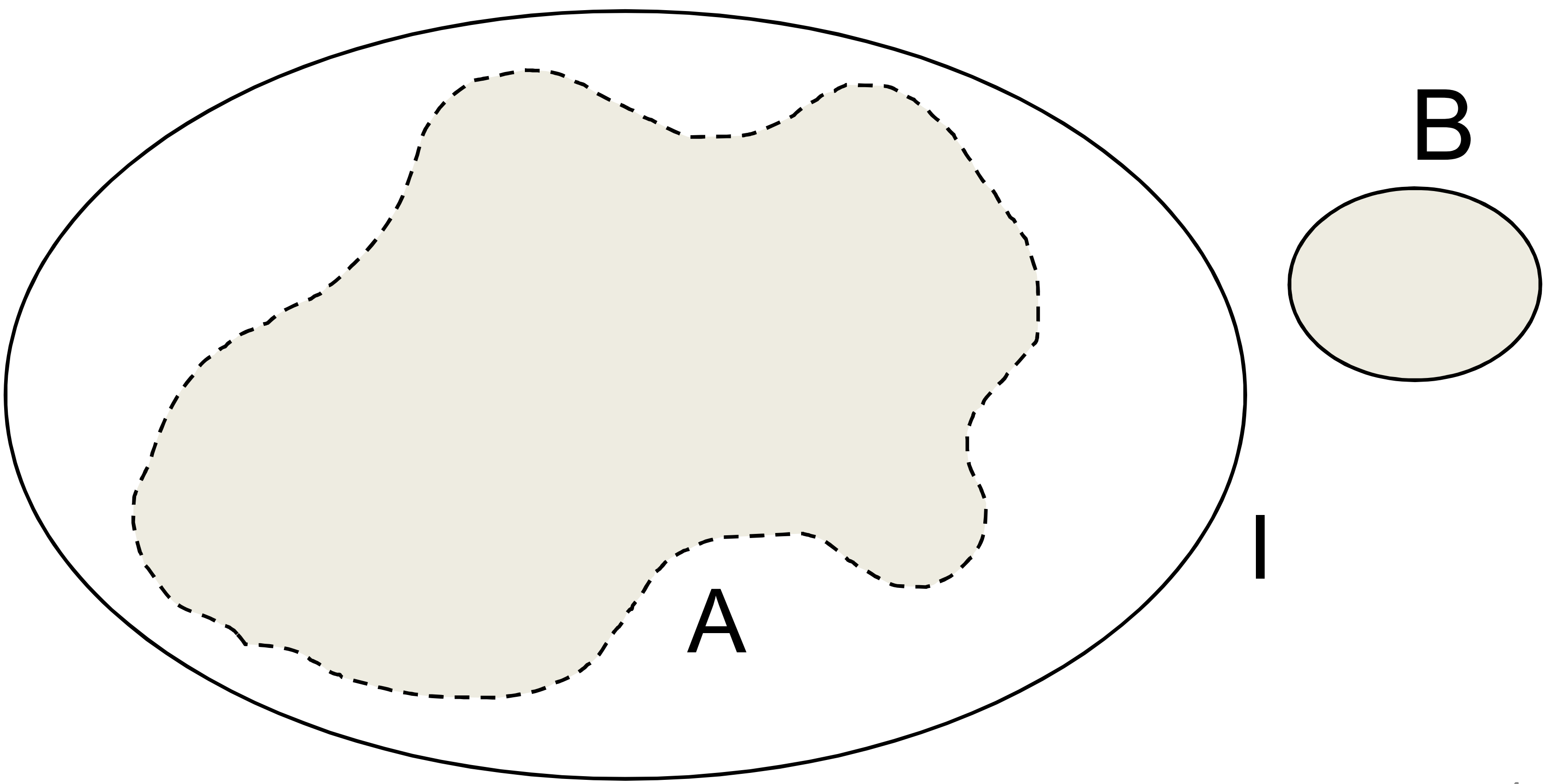
# Interpolation

- For theory T, a T-interpolant I for (A,B)

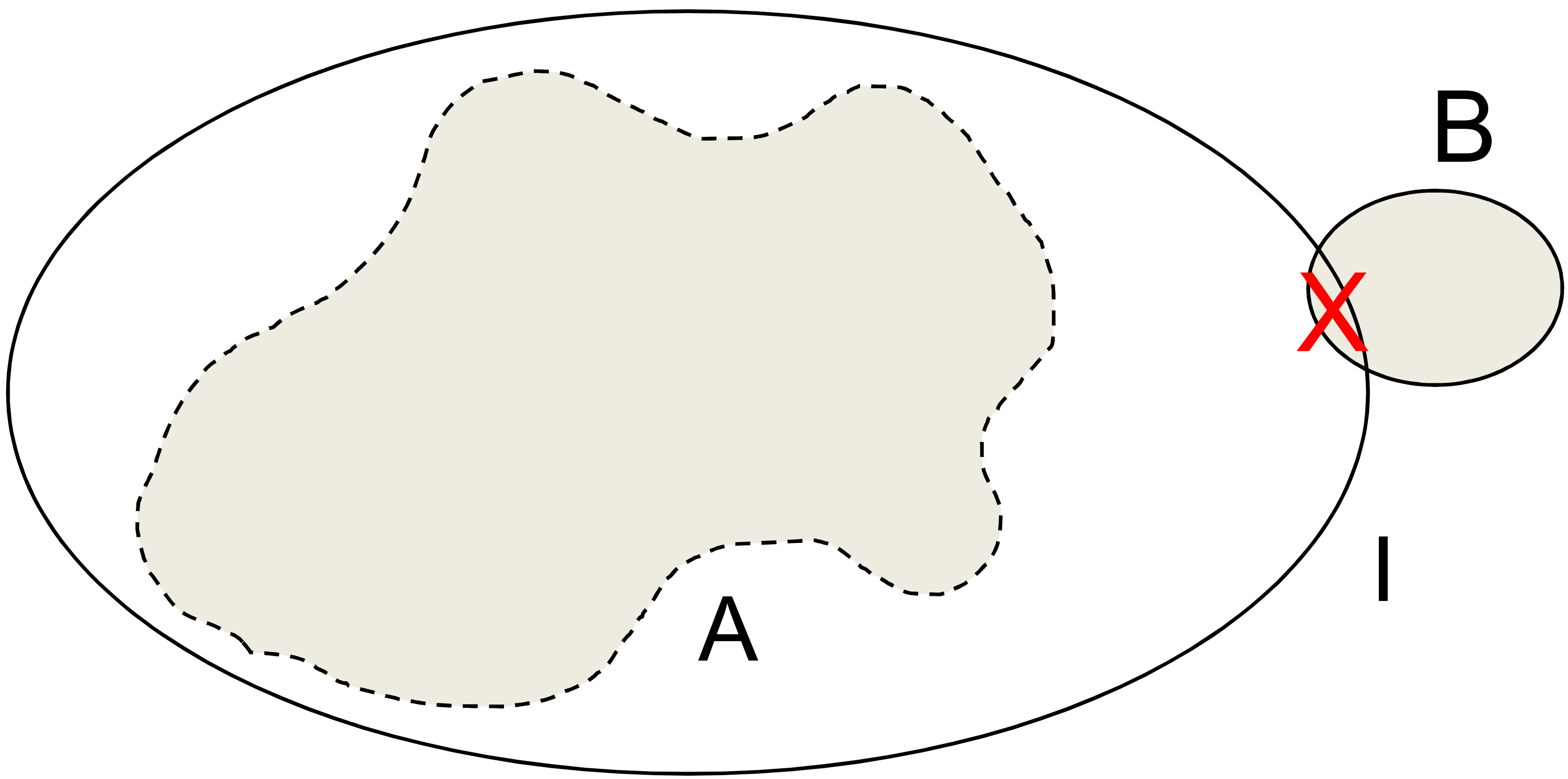
$$(1)\ A \models_T I$$
$$(2)\ B, I \models_T \perp$$
$$(3)\ L(I) \subseteq L(A) \cap L(B)$$

- In some cases, may be efficiently generated from pf
- Applications
  - Model Checking
  - Predicate Abstraction
  - ...
- In some, correctness of interpolant is critical

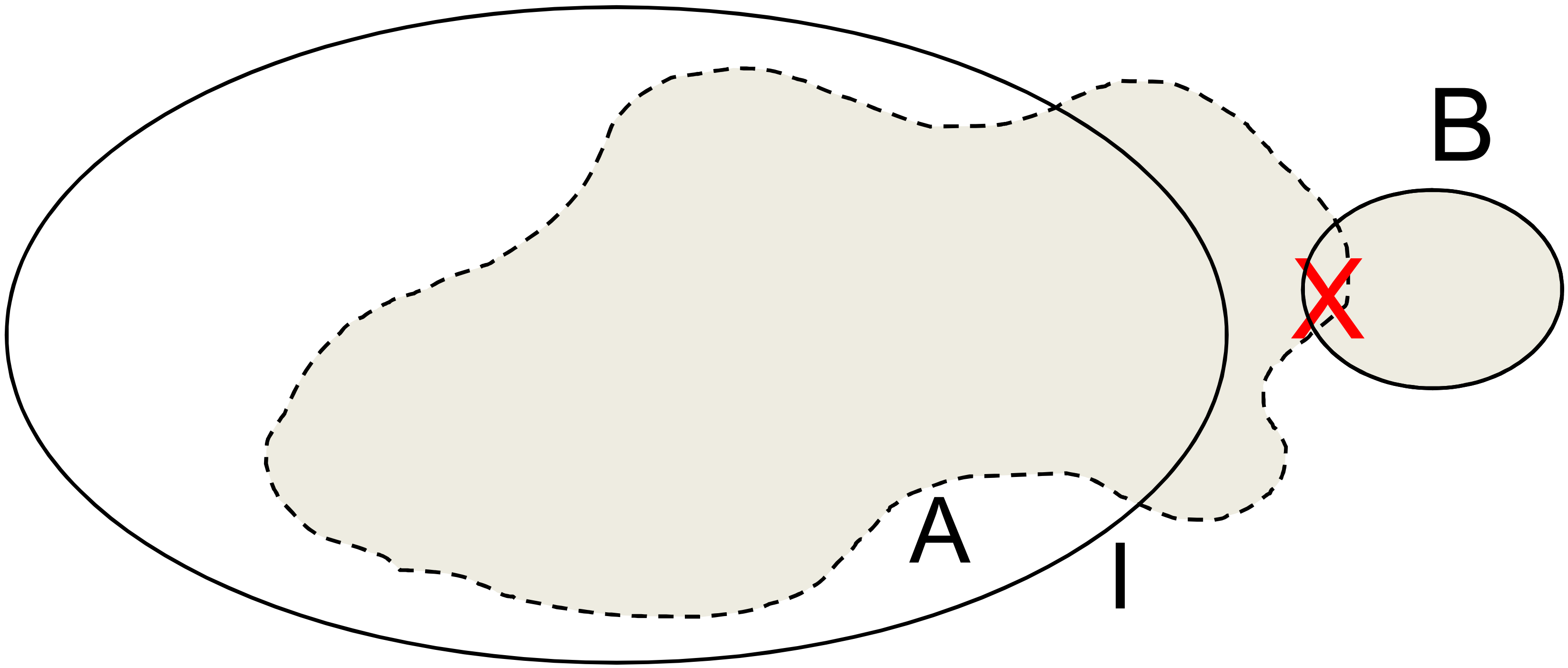
- Interpolant over-approximates reachable states



- Alleged Interpolants that violate $B, I \models_T \perp$ lead to spurious error states



- Alleged Interpolants that violate $A \models_T I$ may lead to unsoundness




*Safe instead of Unsafe*

# Certified Interpolation

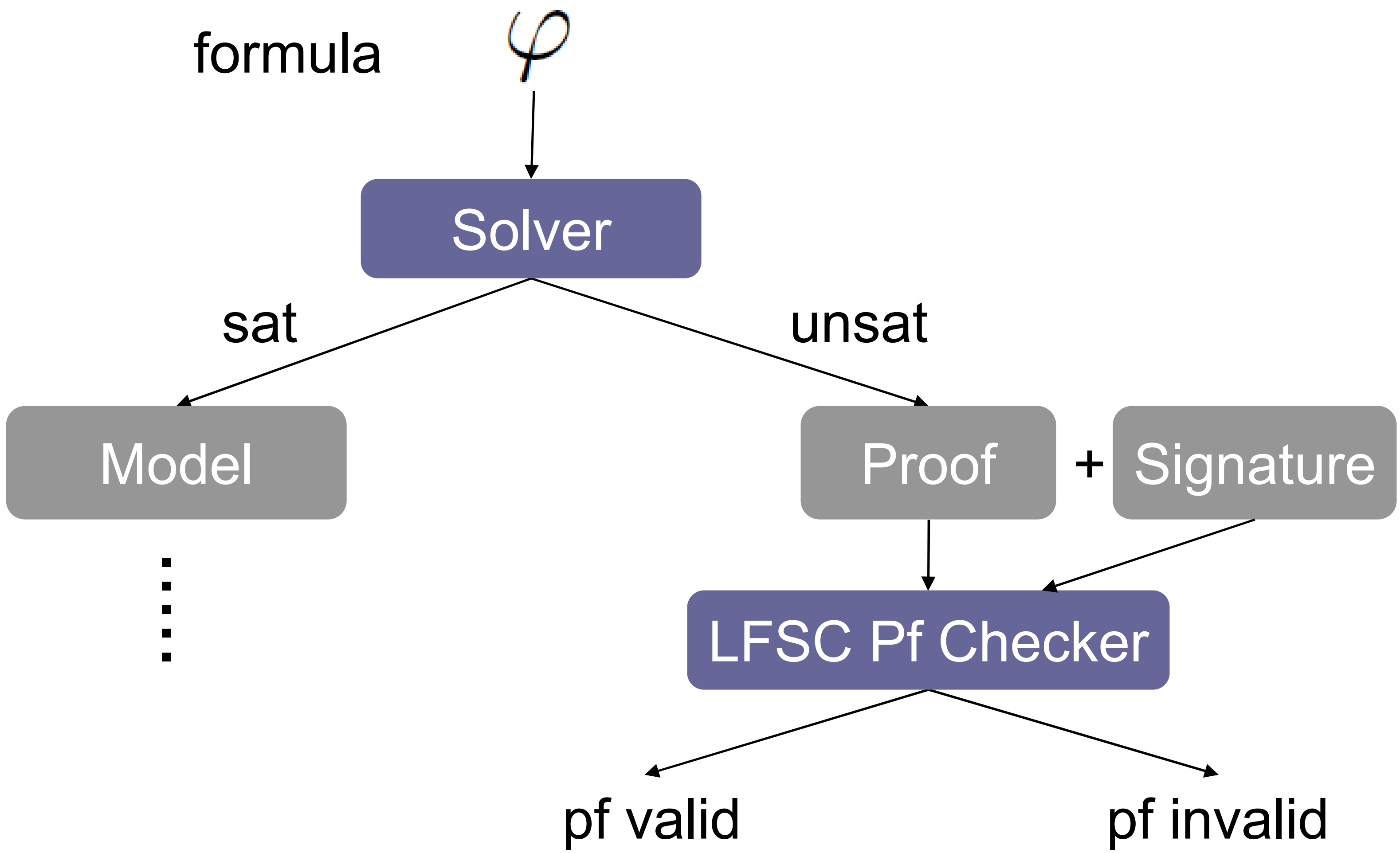
- Clearly, correctness of interpolant is important
- SMT solvers produce interpolants
  - None do so in a verified way

- Goal: Certify interpolants via proof checker
  - Certification via Interpolating Calculi
  - Alternatively, may *generate* interpolants
    - Certified Correct by Construction

# Proof Checking in LFSC

- LFSC: Meta-logical Framework (Stump ’08)
  - Proof + user-defined Signature
- Based on Edinburgh Logical Framework (LF)
- Extends LF with
  - Computational Side Conditions
  - Support for Integer, Rational arithmetic
- Proofs as Terms
  - Proof checking amounts to type checking

# Proof Checking

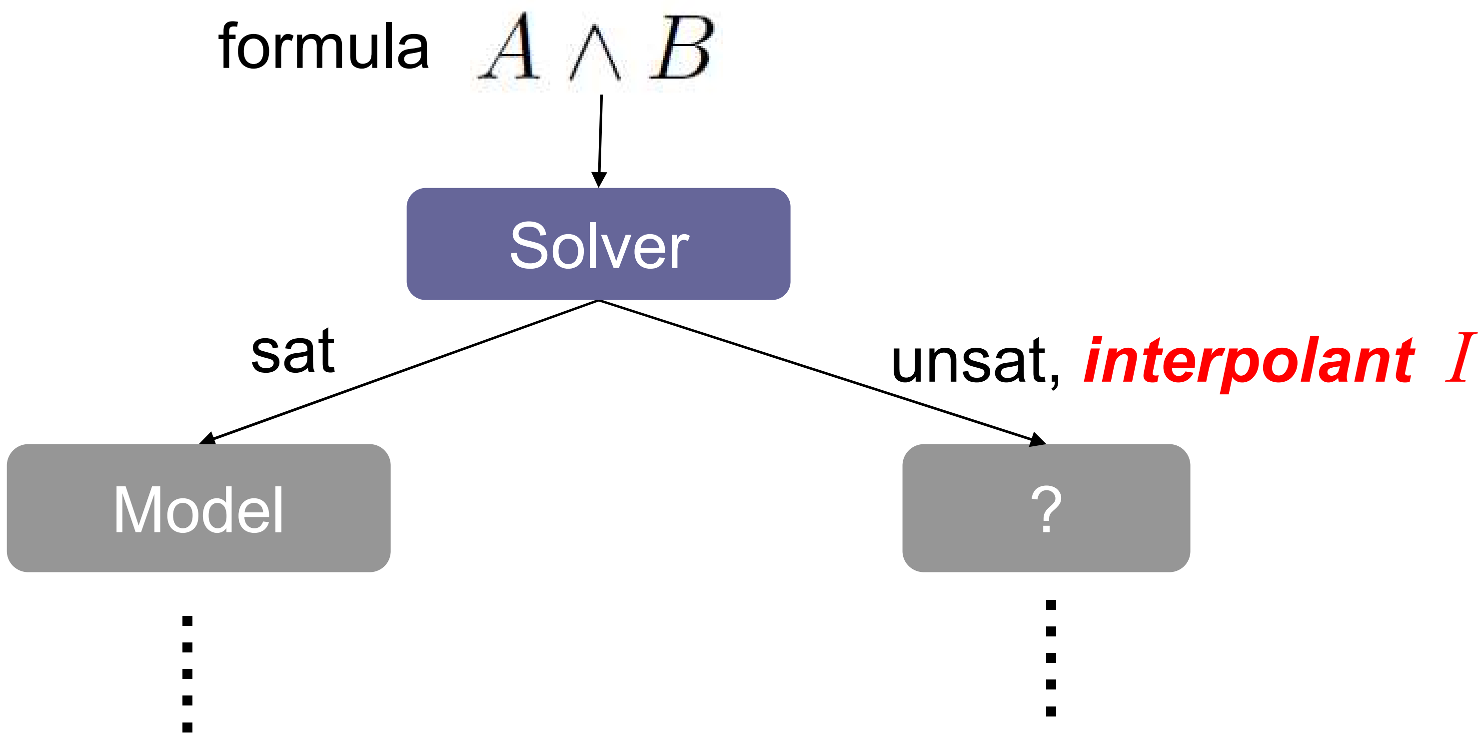


# Proof Checking

```
(check
      (% ...
      (% ...
      (% v (proof φ)
      (: (proof false)
          P
          ))...)
```

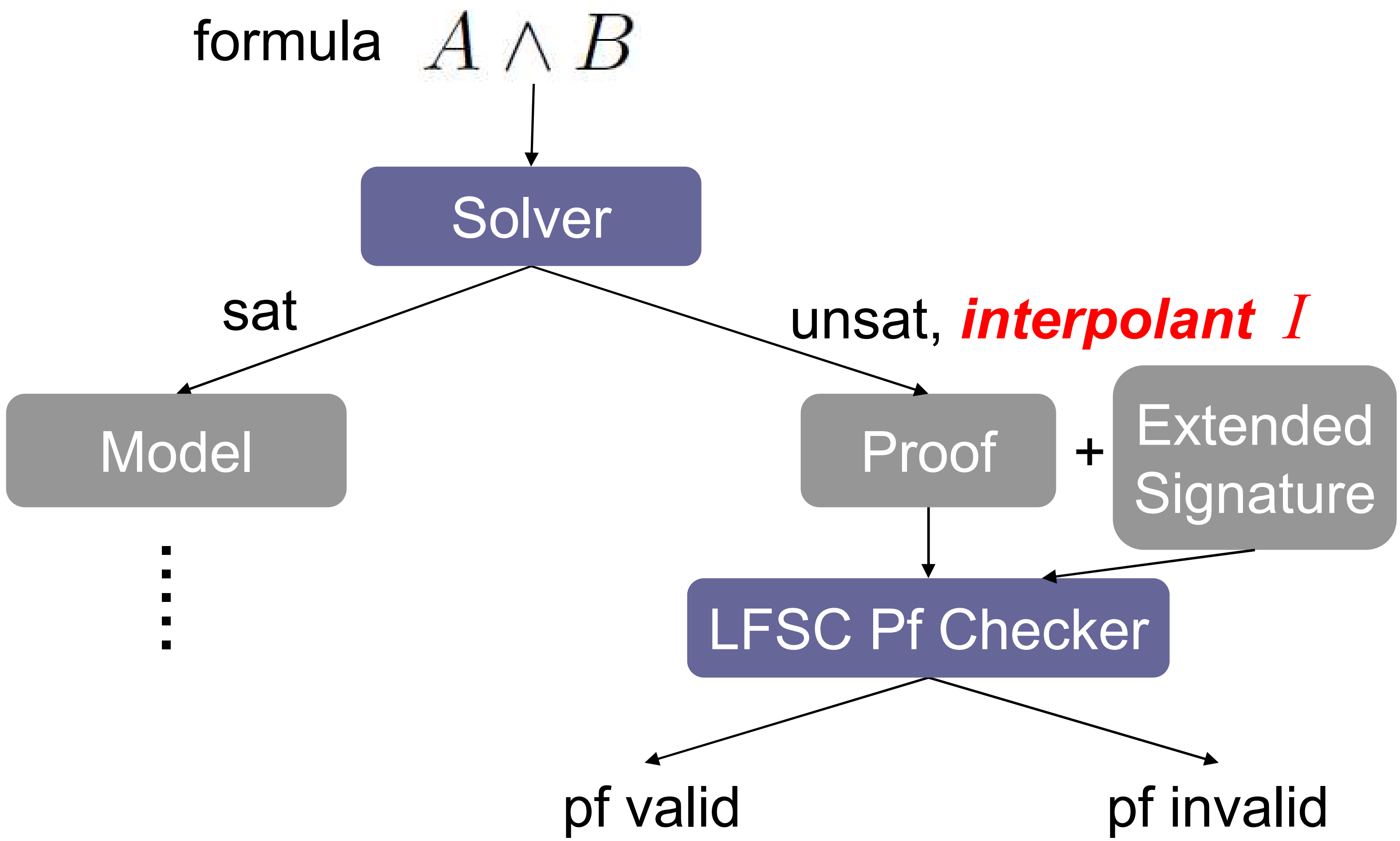
- LFSC proofs reside in *check* commands
- (: T s) - Check whether term s has type T
- Use of **(proof** $\varphi$**)** type for formula $\varphi$
- If success, we have certified $\varphi \models \perp$

# Proof Checking: Interpolants




- Since LFSC is meta-framework, we can extend signature to type-check proofs about interpolants

# Proof Checking: Interpolants



# Proof Checking: Interpolants

```
(check
      (% ...
      (% u (proof A)
      (% v (proof B)
      (: (interpolant I )
           P
          ))...)
```

- Use of **(interpolant** $I$**)** type for formula $I$
- If $P$ has type **(interpolant** $I$**)**,
  - $I$ is a certified interpolant for $(A, B)$

# Proof Checking: Interpolants

- SMT solver produces interpolant + proof
- LFSC verifies that proof:
  (1) Successfully type checks, and
  (2) Shows claimed interpolant is an interpolant.
- If success, we have a certified interpolant
- Solver + Checker must agree on the interpolant

# Interpolant Generation via Proof Checking

- Alternatively:

  *Use proof checker as the interpolant generator*

- Solver writes proof in same signature
  - Constructs term of type `(interpolant` $I$`)`,
    - for some value of $I$, unknown a priori
  - Value of $I$ computed by type inference

# Interpolant Generation via Proof Checking

- LFSC terms may contain hole symbols "_"
- For example:

(trans _ _ _ (= $t_1$ $t_2$) (= $t_2$ $t_3$))

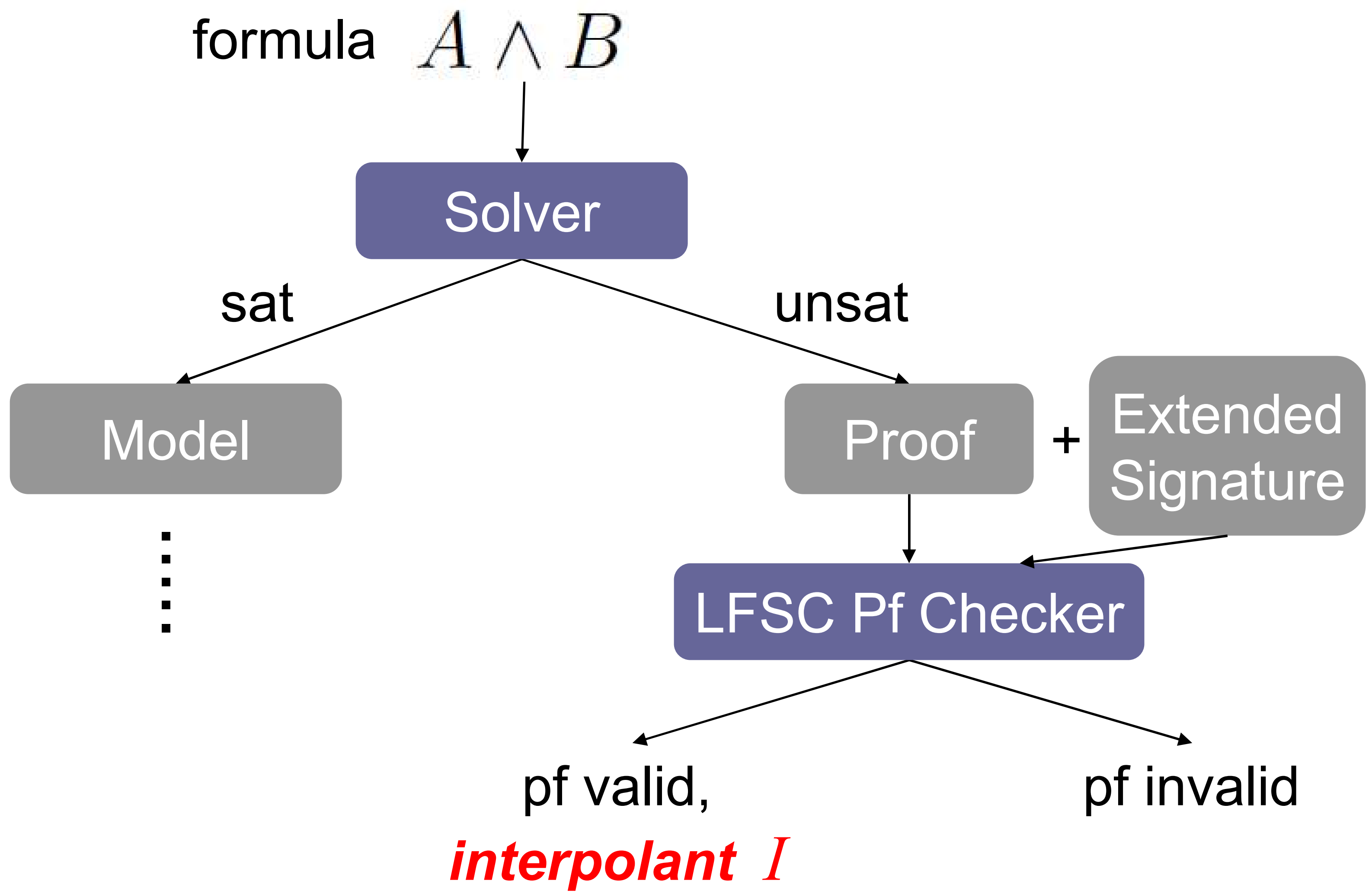
- Allow proof checker to fill in value of interpolant
  - Certified correct by construction

# Interpolant Generation via Proof Checking

```
(check
      (% ...
      (% u (proof A)
      (% v (proof B)
      (: (interpolant _)
           P
           ))...)
```

- The interpolant field is left unspecified “_”
- If P has type **(interpolant** $I$**)** for some $I$,
  - Value of $I$ is given to user
  - $I$ is a certified interpolant for $(A, B)$

# Interpolant Generation via Proof Checking



# Interpolant Generating Calculi in LFSC

$$\frac{\varphi_1 \quad \cdots \quad \varphi_n}{\varphi'} \quad \Rightarrow \quad \frac{\varphi_1[A_1] \quad \cdots \quad \varphi_n[A_n]}{\varphi'[A']}$$

- Interpolant generating calculi encoded in LFSC
- Augment rules with extra information
  - Encoding of partial interpolants $\varphi\,[\varphi_1\ \varphi_2\ c]$
    - where $[\varphi_1\ \varphi_2\ c]$ is annotation for $\varphi$
    - **(p_interpolant** $\varphi$ $\varphi_1$ $\varphi_2$ $c$**)** type

# Interpolant Generating Calculi in LFSC

- Tested LFSC framework for interpolants
- Examined theory of equality (EUF)
  - Simple calculus for interpolation in EUF
  - 203 lines of type declarations
  - 21 lines of side condition code
- Use CVC3 for proof generation

- Preliminary experiments on other theories
  - Boolean, QF_LRA, QFPA, …

# Interpolating Calculus for EUF

- Interpolating Calculus for EUF
  – Proposed by McMillan ’03
- Modified version of this calculus
  – Based on method given by Fuchs et. al. ’09
  – Simpler, flexibility in interpolants produced
- Extension of standard EUF proof calculus
  – Reflexivity, Symmetry, Transitivity, Congruence
  – Deduces only *colorable* equalities

# Interpolating Calculus for EUF

- A term, literal, or formula is:
  - $A$-colorable ($B$-colorable) if:
    - Its free non-logical symbols contained in $L(A)$ ($L(B)$)
  - colorable if:
    - It is either $A$-colorable or $B$-colorable
  - $AB$-colorable if:
    - It is both $A$-colorable and $B$-colorable

# Interpolating Calculus for EUF

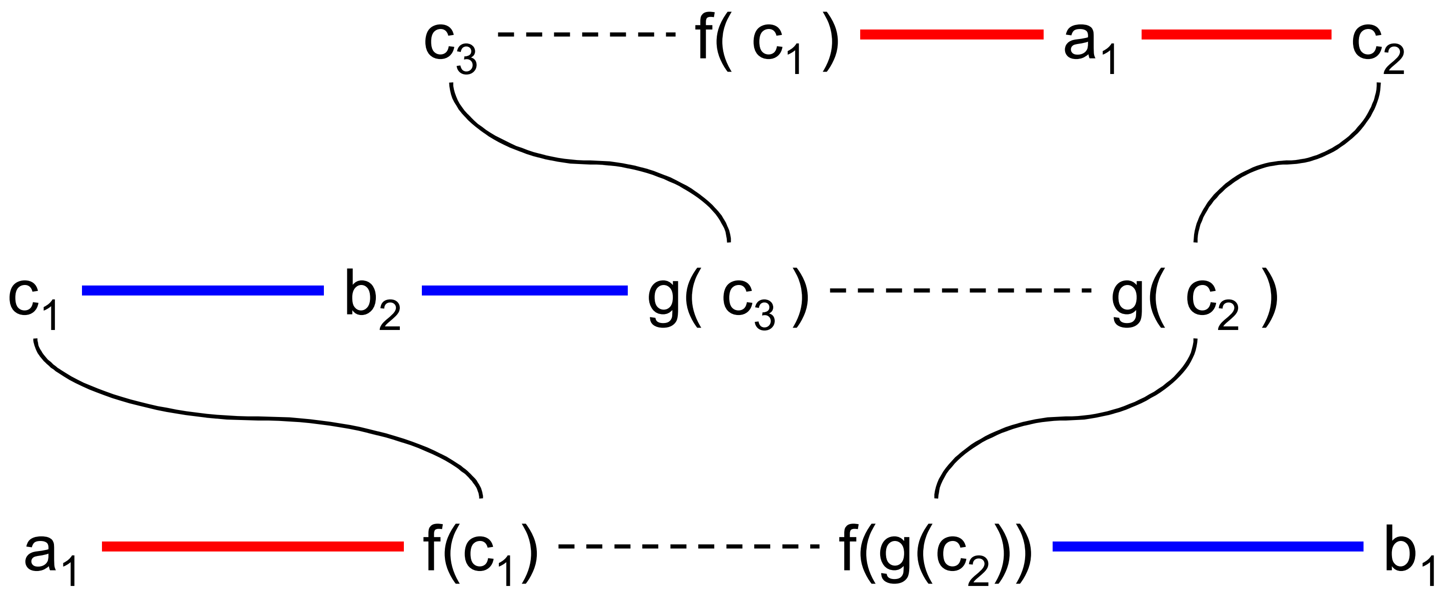
- To produce certified interpolant:
  - Obtain standard proof of UNSAT from CVC3

CVC3 → Proof $\xrightarrow{\text{lifting}}$ Int Proof .....

  - Proof is "lifted" to a proof with:
    – Only colorable equalities
    – Color annotations
  - Lifting process can be described by colored congruence graphs

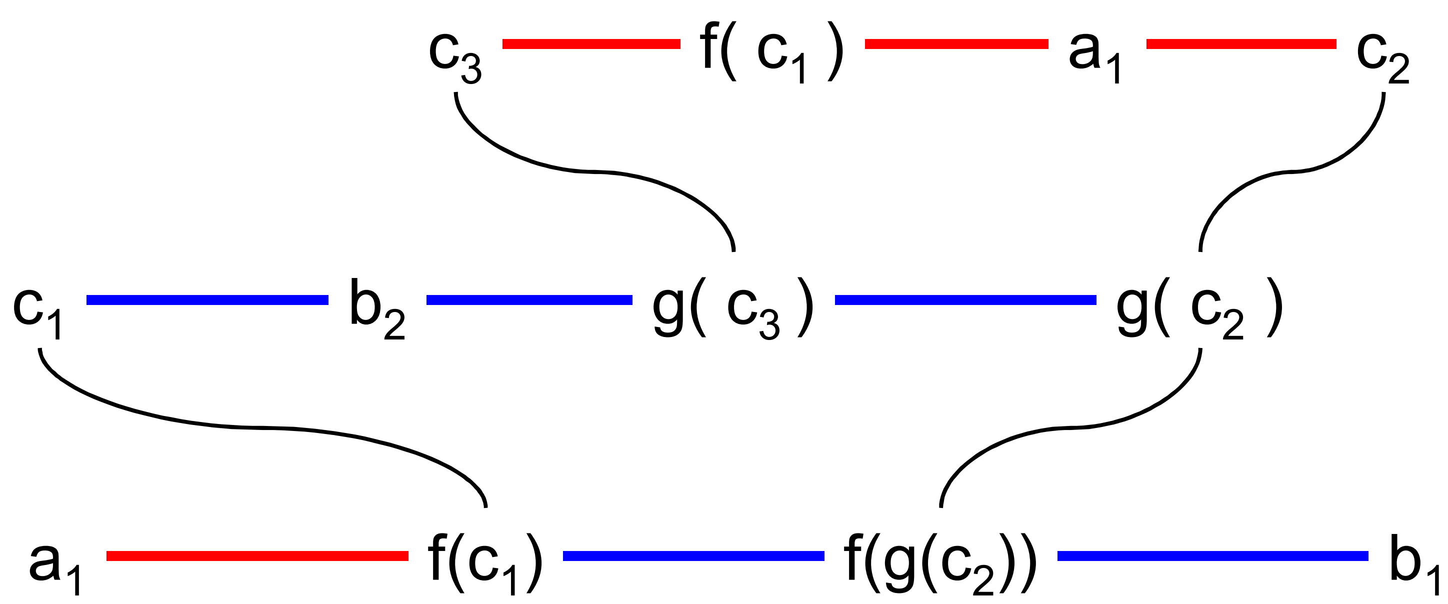
# Colored Congruence Graphs

- Proof lifting via Colored Congruence Graphs
  - Edges are assumptions or applications of congruence
  - Edges annotated with a ***color***



# Colored Congruence Graphs

- Edges between AB-colorable terms can be colored either A or B



# Interpolating Calculus for EUF

- Build equality chains of A-, B- colorable terms

$t_1$ ——— - - - ——— $t_2$, $t_1$ ——— - - - ——— $t_2$, etc.

- Partial Interpolant of form $t_1 \approx t_2\,[\varphi, \psi, c]$

where
$$\begin{array}{ll}
(1) & A \models \varphi; \\
(2) & B, \varphi \models \psi; \\
(3) & A, \psi \models t_1 \approx t_2; \text{ and} \\
(4) & t_i \text{ is } c\text{-colorable for } i = 1, 2.
\end{array}$$

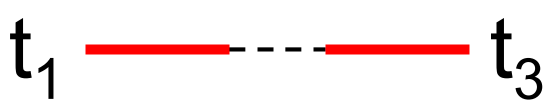
# Interpolating Calculus for EUF

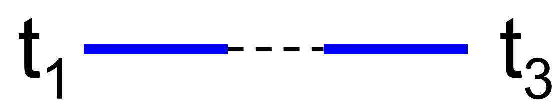
- Rules for A- and B- colored chains

$$\frac{t_1 \approx t_2 \quad t_2 \approx t_3}{t_1 \approx t_3}\ \text{trans}$$

$$\frac{\begin{array}{c} t_1 \approx t_2\,[\varphi_1, \psi_1, c] \\ t_2 \approx t_3\,[\varphi_2, \psi_2, c'] \\ \{\ t_1,\ t_3 \text{ are } A\text{-colorable }\} \end{array}}{t_1 \approx t_3\,[\varphi_1 \wedge \varphi_2, \psi_1 \wedge \psi_2, A]}$$

$$\frac{\begin{array}{c} t_1 \approx t_2\,[\varphi_1, \psi_1, c] \\ t_2 \approx t_3\,[\varphi_2, \psi_2, c'] \\ \{\ t_1,\ t_3 \text{ are } B\text{-colorable }\} \end{array}}{\begin{array}{r} t_1 \approx t_3\,[\varphi_1 \wedge \varphi_2 \wedge (\psi_1 \wedge \psi_2) \rightarrow \\ (t_1 \approx t_3), t_1 \approx t_3, B] \end{array}}$$



# Encoding into LFSC

$$\frac{t_1 \approx t_2\,[\varphi_1, \psi_1, c] \quad t_2 \approx t_3\,[\varphi_2, \psi_2, c'] \quad \{\ t_1,\ t_3 \text{ are } A\text{-colorable}\ \}}{t_1 \approx t_3\,[\varphi_1 \wedge \varphi_2, \psi_1 \wedge \psi_2, A]}$$

Fields to be filled in

Premises

Side Condition

Conclusion

```
(trans-A
    (! t1 term (! t2 term (! t3 term
    (! ...
    (! p1 (p_interpolant (= t1 t2) φ1 ψ1 c)
    (! p2 (p_interpolant (= t2 t3) φ2 ψ2 c')
    (! s (^ (is_colorable A t1 t3) true)
        (p_interpolant (= t1 t3) (and φ1 φ2)
                              (and ψ1 φ2) A)))…)
```

# Interpolating Calculus for EUF

- Advantages of Calculus
  - Flexibility (Fuchs et. al. ‘09)
    - Coloring between AB-colorable equalities
      - Logical strength
      - Interpolant size
  - Fewer Side Conditions
    - Only two side conditions (term colorability)
      - 21 lines of sc code
    - Can be implemented naturally in LF

# Results

- CVC3 for proof generation
- Tested on EUF theory lemmas
  - Extracted from SMT LIB
  - Unique, ≥ 5 edges in congruence graph
- Tested various partitions of (A,B)
  - k/6 in set A for k = 1...5

# Results

- Tested configurations
  - **euf**:  proof checking
  - **eufi**:  proof checking with interpolant generation
- Proof checking fast w.r.t to solving
  - **euf** 11x faster than solving
  - **eufi** 5x faster than solving
- Interpolants come at small overhead
  - **eufi** 22% overhead with respect to solving + pf generation

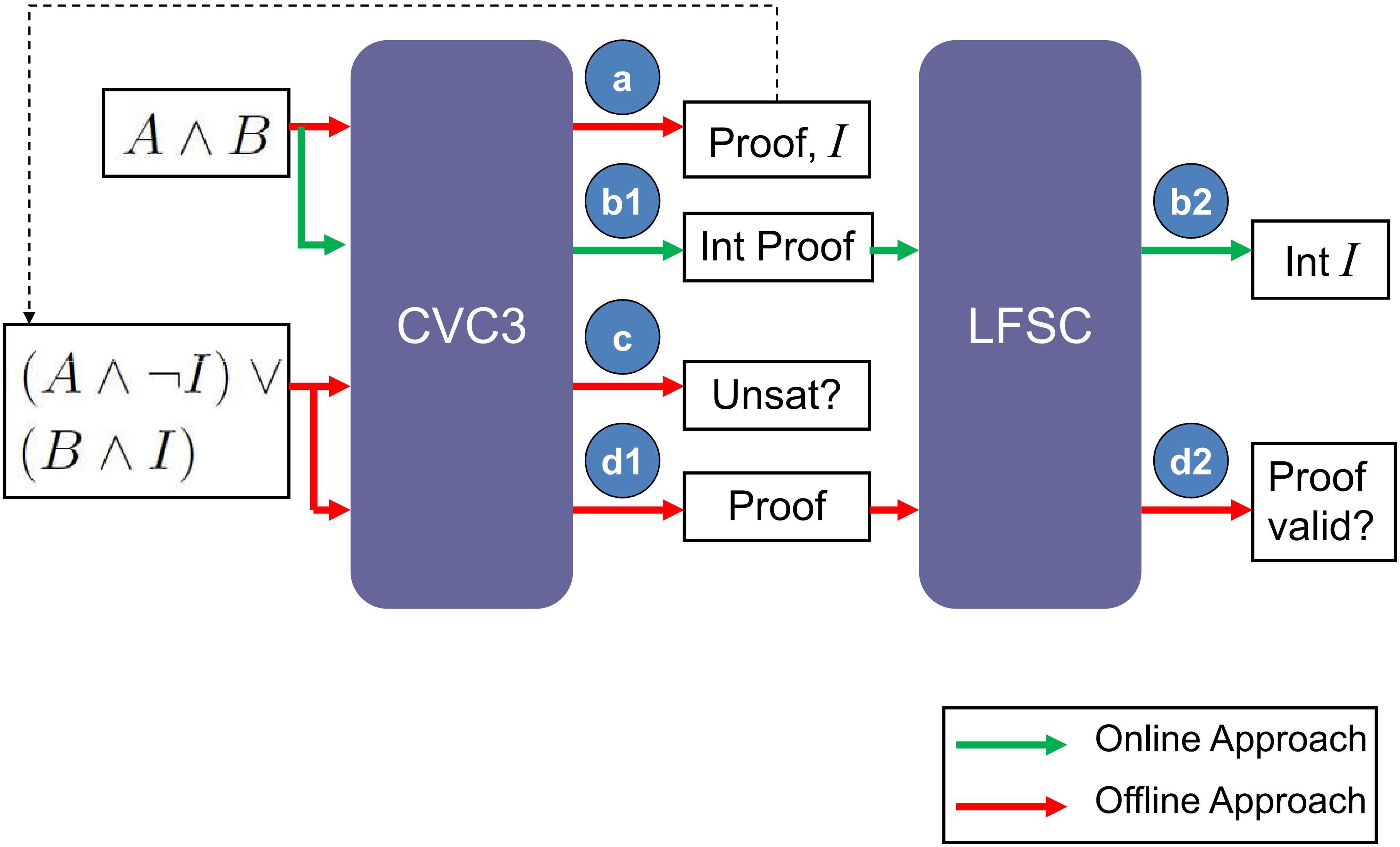
# Offline Approach to Certification

- Alternative: Verify interpolants directly
- For alleged interpolant $I$, prove:

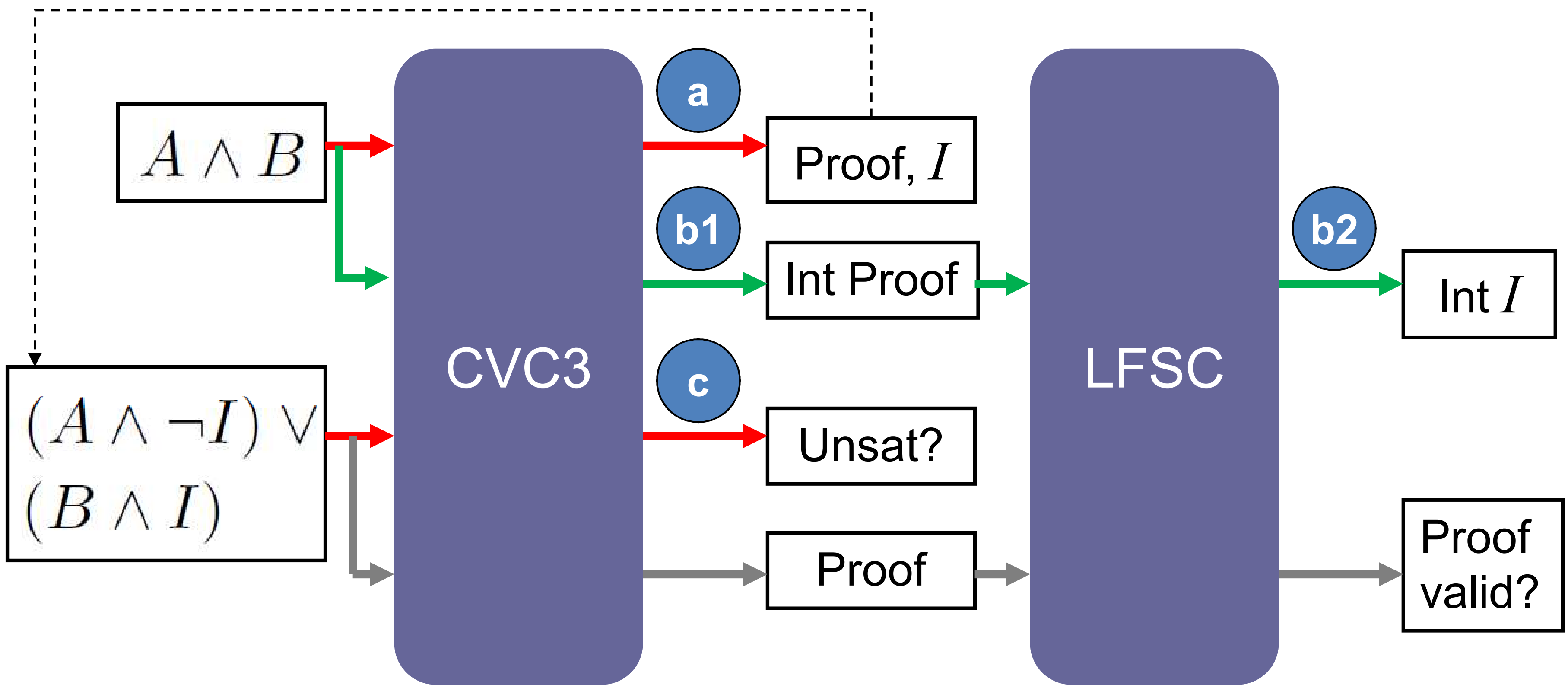
$$(1)\ A \models_T I$$
$$(2)\ B, I \models_T \bot$$

Note: AB-colorability can be easily verified

# Offline vs Online Certification



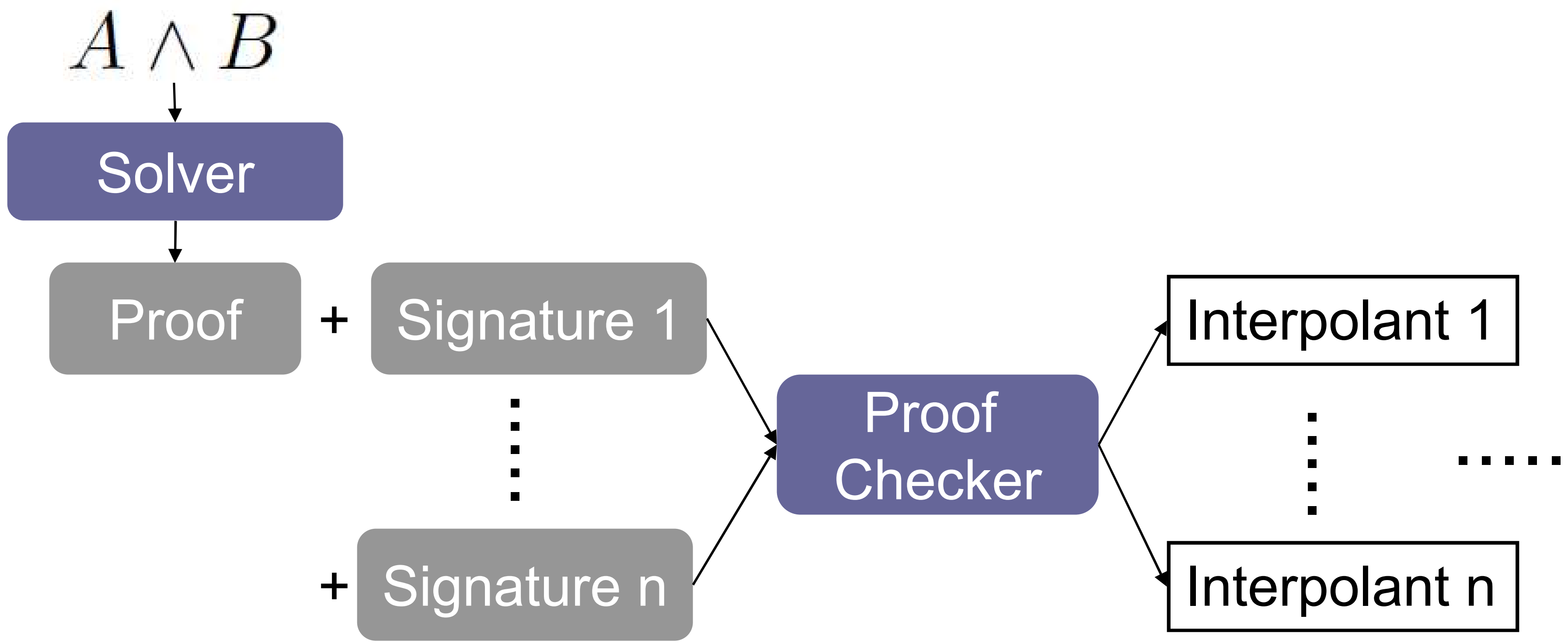
# Offline vs Online Certification




- Certification faster via LFSC
  - b1 + b2 56% of the time of a + c

# Advantages of a Framework Approach

- Proof Checking is faster than Solving
- Idea: generate multiple interpolants from same proof
  - Need only call solver once



# Conclusions

- Efficient method for certified interpolants
- Simple calculus for EUF interpolation
  - Coloring options
  - Few side conditions
- Flexibility of signature
  - Multiple interpolants from same proof
  - Certification of other properties

# Future Work

- Integration with CVC4
- Extension to other theories
  - Boolean + theory lemmas
- Use of new release of LFSC
  - Efficient generation of certified interpolants
- Applications of Interpolants
  - Use of LFSC framework for generation

# Questions?