Probabilistic Reasoning for the Verification of Side-Channel Countermeasures

EGRAPHS talk

Robin Coutelier

TU Wien, Vienna, Austria
robin.coutelier@tuwien.ac.at

November 20th 2025

TU WIEN Informatics

for(syte)

Acknowledgements

The authors acknowledge support from the ERC Consolidator Grant ARTIST 101002685; the TU Wien Doctoral College TrustACPS; the FWF SpyCoDe SFB projects F8504; the WWTF Grant ForSmart 10.47379/ICT22007

Who am I?

About Me

- PhD student at TU Wien, Vienna, Austria
- Advisor: Prof. Laura Kovács
- Research interests: SAT, SMT, Automated Theorem Proving

Who am I?

About Me

- PhD student at TU Wien, Vienna, Austria
- Advisor: Prof. Laura Kovács
- Research interests: SAT, SMT, Automated Theorem Proving

CV

- M.Sc. in Computer Engineering, University of Liège, Belgium (2023)
- B.Sc. in Engineering, University of Liège, Belgium (2021)
- Chinese language studies, Yunnan Normal University, China (2018)

Who am I?

About Me

- PhD student at TU Wien, Vienna, Austria
- Advisor: Prof. Laura Kovács
- Research interests: SAT, SMT, Automated Theorem Proving

CV

- M.Sc. in Computer Engineering, University of Liège, Belgium (2023)
- B.Sc. in Engineering, University of Liège, Belgium (2021)
- Chinese language studies, Yunnan Normal University, China (2018)

I am not

- A security expert
- A term rewriting expert

Side-Channel Attack

Credits: Prof. Roderick Bloem

Side-Channel Attack

Credits: Prof. Roderick Bloem

Side-Channel Attack

Credits: Prof. Roderick Bloem

(a) At home

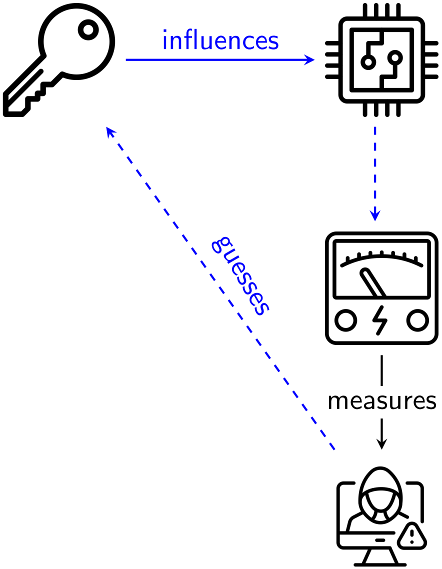
Side-Channel Attack

Credits: Prof. Roderick Bloem

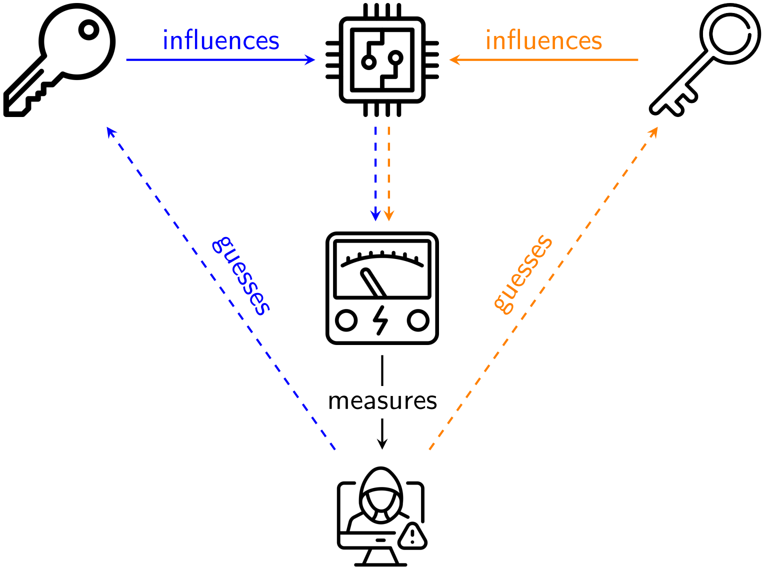
(a) At home

(b) Away from home

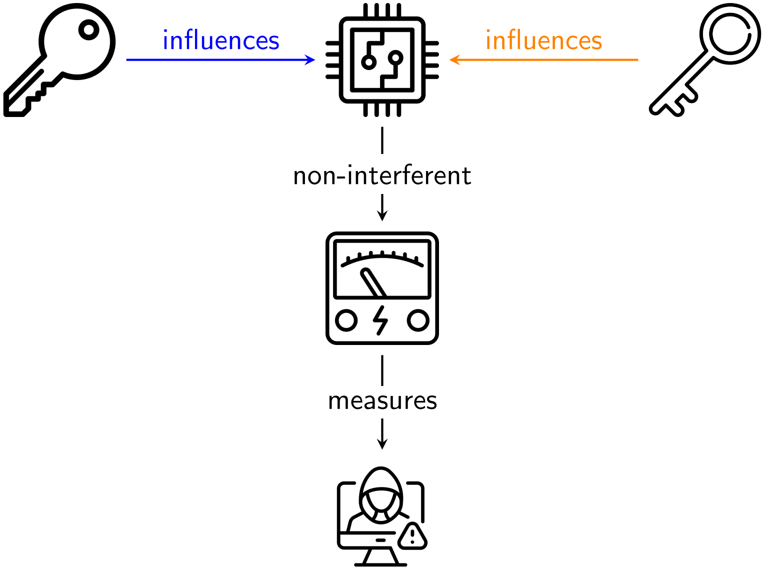
Power Side-Channel Attack



Power Side-Channel Attack



Power Side-Channel Attack



Shares for Secret Masking

Secret Masking

Let s be a secret, we split the secret into n shares $s_1, s_2, \ldots, s_n$ such that:

$$s = s_1 \oplus s_2 \oplus \ldots \oplus s_n$$

Shares for Secret Masking

Secret Masking

Let s be a secret, we split the secret into n shares $s_1, s_2, \ldots, s_n$ such that:

$$s = s_1 \oplus s_2 \oplus \ldots \oplus s_n$$

$\mathbf{s}$	$\mathbf{s_1}$	$\mu\mathbf{P}$
0	0	$0W$
1	1	$1W$

Shares for Secret Masking

Secret Masking

Let s be a secret, we split the secret into n shares $s_1, s_2, \ldots, s_n$ such that:

$$s = s_1 \oplus s_2 \oplus \ldots \oplus s_n$$

$\mathbf{s}$	$\mathbf{s_1}$	$\mathbf{s_2}$	$\mu\mathbf{P}$	$\mu^2\mathbf{P}$
0	0	0	$1W$	$1W^2$
0	1	1		
1	0	1	$1W$	$0W^2$
1	1	0		

Shares for Secret Masking

Secret Masking

Let s be a secret, we split the secret into n shares $s_1, s_2, \ldots, s_n$ such that:

$$s = s_1 \oplus s_2 \oplus \ldots \oplus s_n$$

$\mathbf{s}$	$\mathbf{s_1}$	$\mathbf{s_2}$	$\mathbf{s_3}$	$\mu\mathbf{P}$	$\mu^{\mathbf{2}}\mathbf{P}$	$\mu^{\mathbf{3}}\mathbf{P}$
0	0	0	0	1.5W	0.75W^2	-0.75W^3
0	1	1	0			
0	1	0	1			
0	0	1	1			
1	0	0	1	1.5W	0.75W^2	0.75W^3
1	1	1	1			
1	0	1	0			
1	1	0	0			

Probe-Isolating Non-Interference

Abstract Goal

$$\forall s, s'.\ P(power|s) \sim P(power|s')$$

Probe-Isolating Non-Interference

Abstract Goal

$$\forall s, s'.\ P(power|s) \sim P(power|s')$$

Probe isolation - Idealized attacker model

The attacker can sample t wires in the circuit.
Can the attacker obtain information about the secret s?

Probe-Isolating Non-Interference

Abstract Goal

$$\forall s, s'.\ P(power|s) \sim P(power|s')$$

Probe isolation - Idealized attacker model

The attacker can sample t wires in the circuit.
Can the attacker obtain information about the secret s?

Practical Goal

$$\bigwedge_{W \in \mathcal{C}} \forall w, s, s'.\ probe(W) \Rightarrow P(W = w|S = s) = P(W = w|S = s')$$

Probe-Isolating Non-Interference

Abstract Goal

$$\forall s, s'.\ P(power|s) \sim P(power|s')$$

Probe isolation - Idealized attacker model

The attacker can sample t wires in the circuit.
Can the attacker obtain information about the secret s?

Practical Goal

$$\bigwedge_{W \in \mathcal{C}} \forall w, s.\ probe(W) \Rightarrow P(W = w | S = s) = P(W = w)$$

Probe-Isolating Non-Interference

Abstract Goal

$$\forall s, s'.\ P(power|s) \sim P(power|s')$$

Probe isolation - Idealized attacker model

The attacker can sample t wires in the circuit.
Can the attacker obtain information about the secret s?

Practical Goal (simplified)

$$\bigwedge_{W \in \mathcal{C}} \forall w, s_1, \ldots, s_n.\ probe(W) \Rightarrow \bigvee_{i=1}^{n} P(W = w | S_i = s_i) = P(W = w)$$

Proof by Refutation

Goal

Let $\mathcal{C}(\bar{x})$ be the definition of the circuit, and $\mathcal{P}(\bar{x})$ be the security property for some input $\bar{x}$. We wish to prove that the following is always true:

$$\forall \bar{x}.\ \mathcal{C}(\bar{x}) \Rightarrow \mathcal{P}(\bar{x})$$

Proof by Refutation

Goal

Let $\mathcal{C}(\bar{x})$ be the definition of the circuit, and $\mathcal{P}(\bar{x})$ be the security property for some input $\bar{x}$. We wish to prove that the following is always true:

$$\forall \bar{x}.\ \mathcal{C}(\bar{x}) \Rightarrow \mathcal{P}(\bar{x})$$

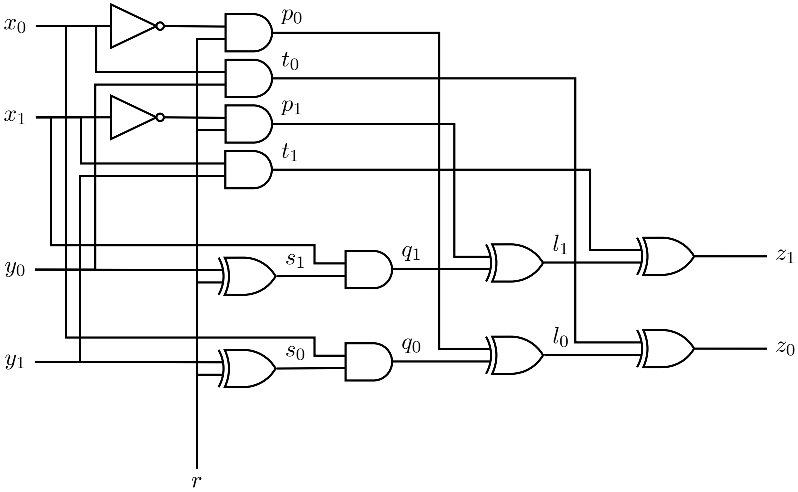
Proof by Refutation

We prove the negation is unsatisfiable on $\bar{a}$:

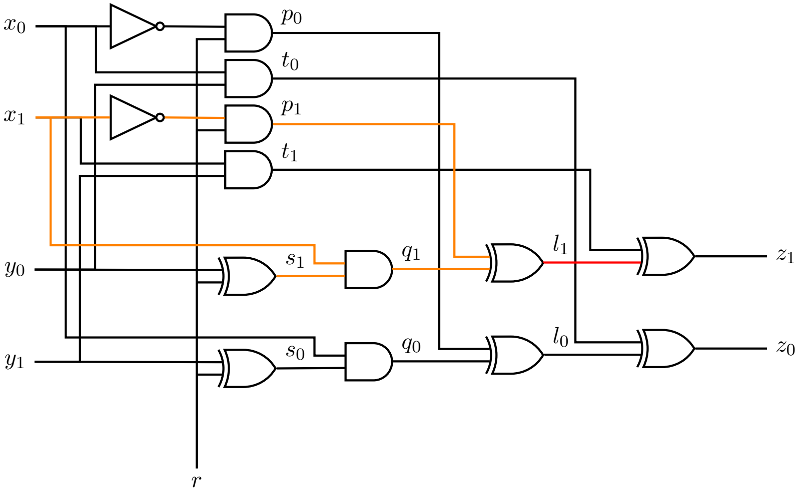
$$\mathcal{C}(\bar{a}) \wedge \neg \mathcal{P}(\bar{a})$$

We eliminated the universal quantifier and have a ground formula in CNF.

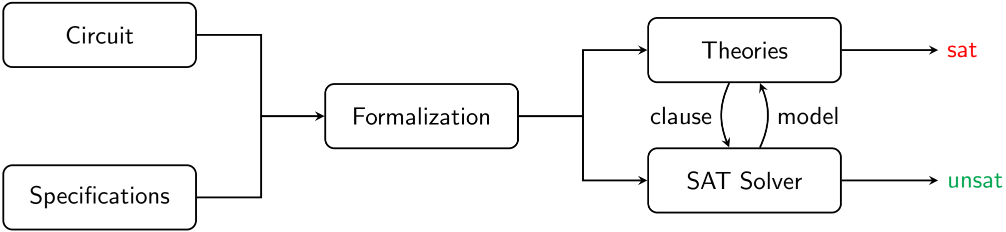
PINI AND Gate Example [Sal23]



PINI AND Gate Example [Sal23]



SMT-like Verification for PINI



SAT and Theory split

Objective

$$\bigwedge_{W \in \mathcal{C}} probe(W) \Rightarrow \bigvee_{i=1}^{n} P(W = w | S_i = s_i) = P(W = w)$$

SAT Part

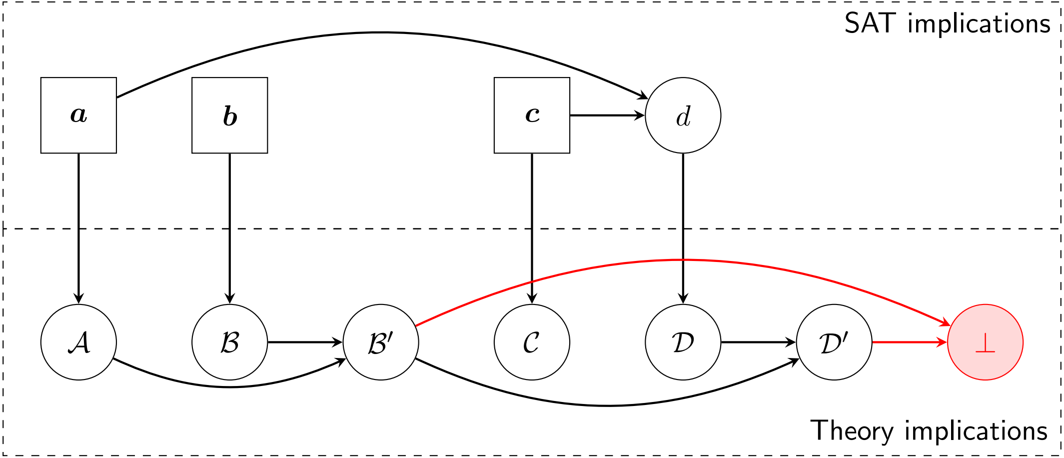
$$\bigwedge_{W \in \mathcal{C}} p_W \Rightarrow \bigvee_{i=1}^{n} b_i$$

Theory Part

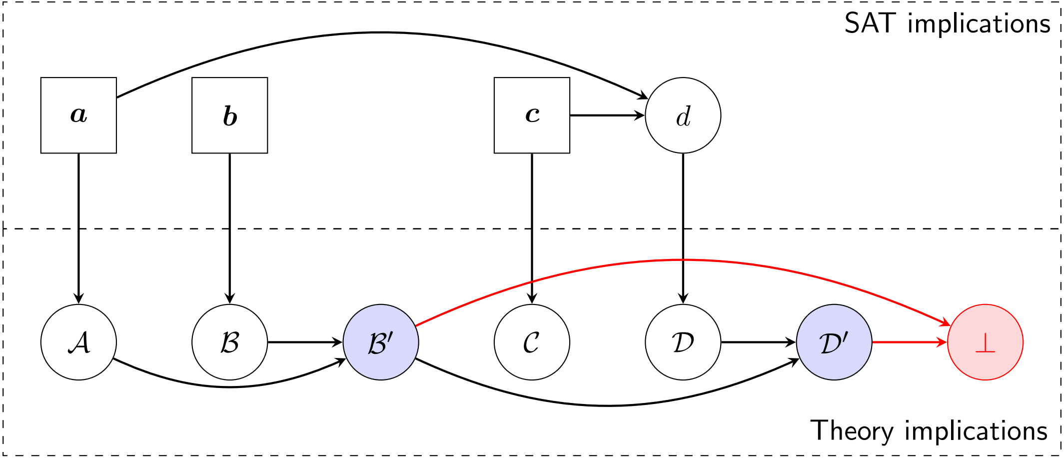
$$p_W \Leftrightarrow probe(W)$$

$$b_i \Leftrightarrow P(W = w | S_i = s_i) = P(W = w)$$

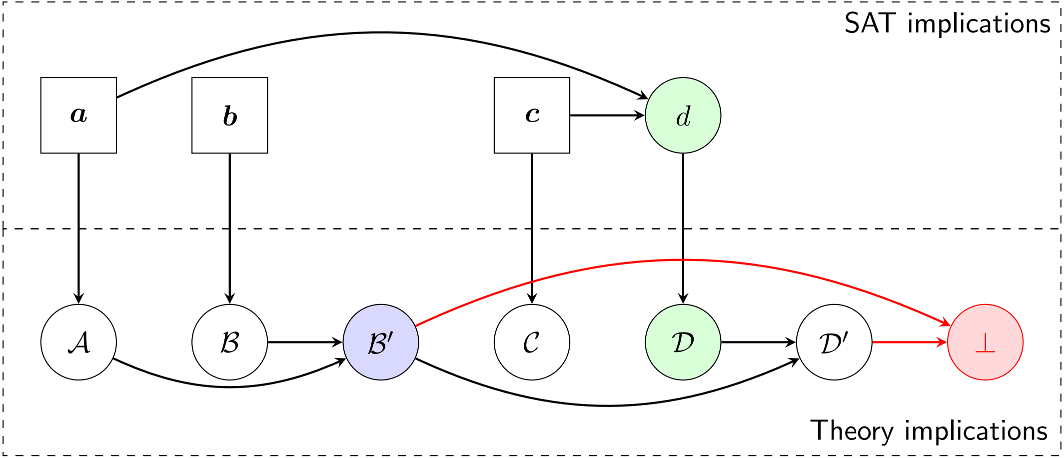
SMT: Theory Conflict Analysis



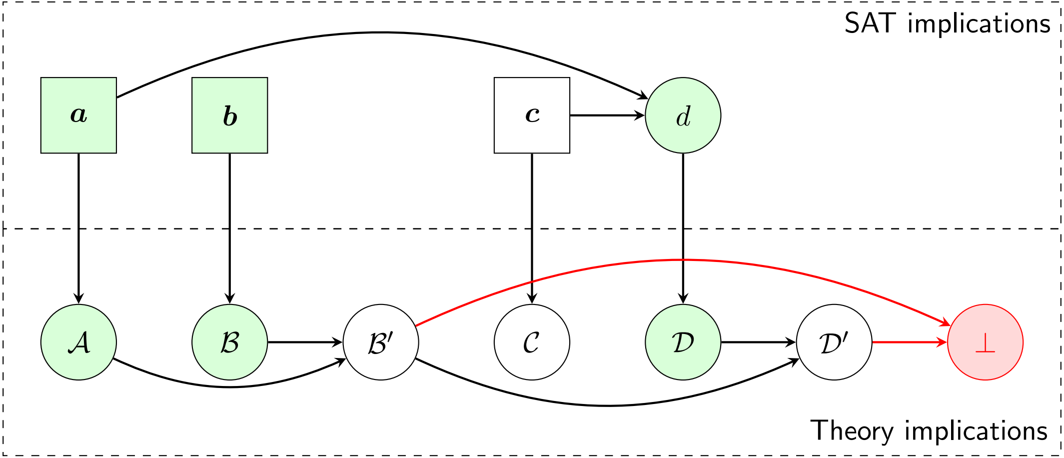
SMT: Theory Conflict Analysis



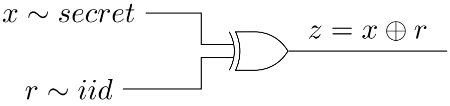
SMT: Theory Conflict Analysis



SMT: Theory Conflict Analysis



Motivating Example



Goal

Prove that the output z is independent of the secret x.

$$P(z|x,r) = x + r - 2x * r \qquad \text{(circuit)}$$

$$P(z,x) \neq P(z) * P(x) \qquad \text{(query)}$$

Motivating Example

Goal

Prove that the output z is independent of the secret x.

$$P(z|x,r) = x + r - 2x * r \qquad \text{(circuit)}$$
$$P(z,x) \neq P(z) * P(x) \qquad \text{(query)}$$

What do we need?

- Probability reasoning
- Equality reasoning
- Non-linear real arithmetic

Solving Steps

$P(z\|x,r) = x + r - 2x * r$	input
$P(z,x) \neq P(z)P(x)$	input
$P(z,x) = P(z\|x)P(x)$	chain rule on $P(z,x)$
$P(z\|x) = P(z\|x,r)P(r\|x) + P(z\|x,1-r)P(1-r\|x)$	law of total probability
$P(r\|x) = P(1-r\|x) = P(r) = 0.5$	r is uniform iid
$P(z\|x) = 0.5(x + r - 2x * r) + 0.5(x + (1-r) - 2x * (1-r))$	substitution
$P(z\|x) = 0.5$	simplification
$P(z\|x) = P(z)$	independence
$P(z,x) = P(z)P(x)$	Conflict with input

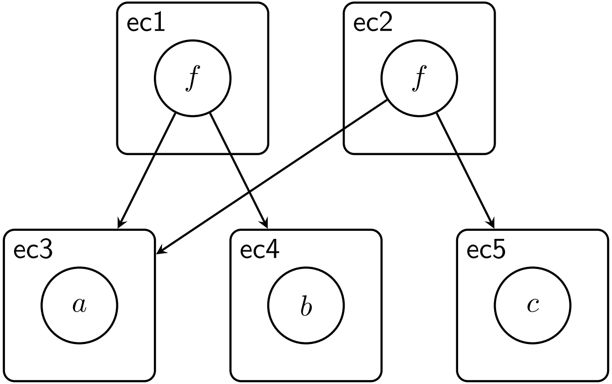
Solving Steps

$P(z\|x,r) = x + r - 2x * r$	input
$P(z,x) \neq P(z)P(x)$	input
$P(z,x) = P(z\|x)P(x)$	chain rule on $P(z,x)$
$P(z\|x) = P(z\|x,r)P(r\|x) + P(z\|x,1-r)P(1-r\|x)$	law of total probability
$P(r\|x) = P(1-r\|x) = P(r) = 0.5$	r is uniform iid
$P(z\|x) = 0.5(x + r - 2x * r) + 0.5(x + (1-r) - 2x * (1-r))$	substitution
$P(z\|x) = 0.5$	AC + inverse
$P(z\|x) = P(z)$	independence
$P(z,x) = P(z)P(x)$	Conflict with input

Solving Steps

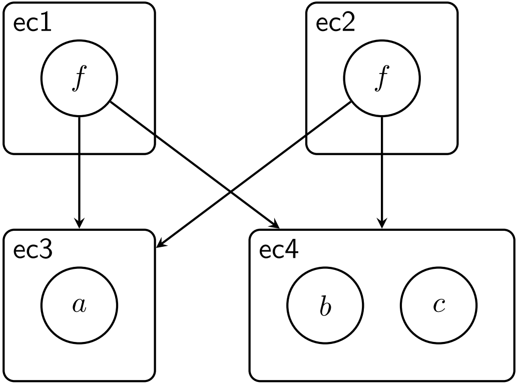
$P(z\|x,r) = x + r - 2x * r$	input
$P(z,x) \neq P(z)P(x)$	input
$P(z,x) = P(z\|x)P(x)$	chain rule on $P(z,x)$
$P(z\|x) = P(z\|x,r)P(r\|x) + P(z\|x,1-r)P(1-r\|x)$	law of total probability
$P(r\|x) = P(1-r\|x) = P(r) = 0.5$	r is uniform iid
$P(z\|x) = 0.5(x + r - 2x * r) + 0.5(x + (1-r) - 2x * (1-r))$	substitution
$P(z\|x) = 0.5$	AC + inverse
$P(z\|x) = P(z)$	independence
$P(z,x) = P(z)P(x)$	Conflict with input

Egraphs



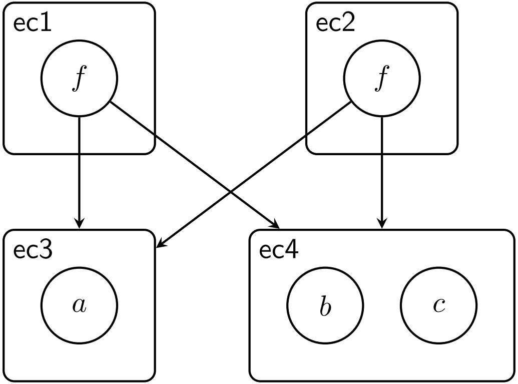
Assert : $b = c$

Egraphs



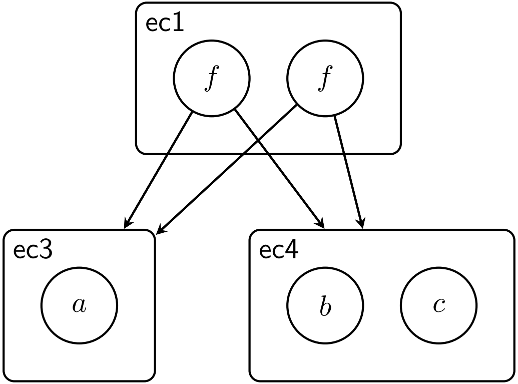
Assert : $b = c$

Egraphs



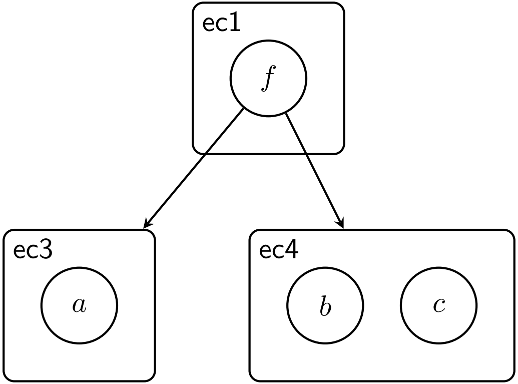
Congruence : $f(ec3, ec4)$

Egraphs



Congruence : $f(ec3, ec4)$

Egraphs



Congruence : $f(ec3, ec4)$

Why Egraphs?

Advantages

- Compact set of equalities.
- Studied by SMT and rewriting communities.
- Efficient for congruence closure.

Why Egraphs?

Advantages

- Compact set of equalities.
- Studied by SMT and rewriting communities.
- Efficient for congruence closure.

Challenges

- Probability calculus is not well understood.
- Combination of theories.
- Polynomials are known to be difficult.
- Complex term introduction.

What about EGG? [WWF+20]

Similarities.

- Saturation of Egraphs.
- Rewrite rules for theory reasoning.

Differences.

- Conflict analysis and backtracking (known from SMT but more complicated).
- Proof production (known from SMT).
- Lemma generation (application specific).
- New term and function symbol introduction (technically not difficult with custom data structures).

Challenge I: Non-Linear Real Arithmetic

NRA is non terminating with rewrites [BN98]

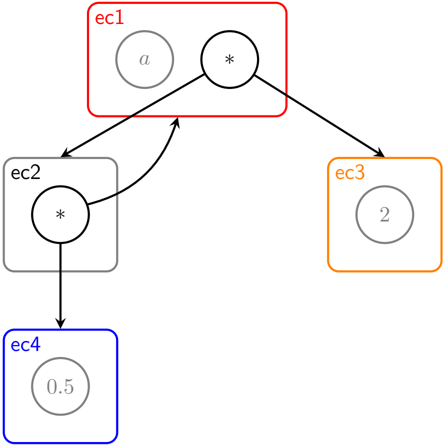
Terms like $(x * a) * \frac{1}{x}$ can be rewritten forever using associativity and commutativity of $*$ and the rule $x * \frac{1}{x} \rightarrow 1$.

Example in PINI

$P(z) = P(x) + P(r) - 2 * P(x) * P(r)$ becomes $P(z) = P(x) + P(r) - 2 * P(x) * 0.5$

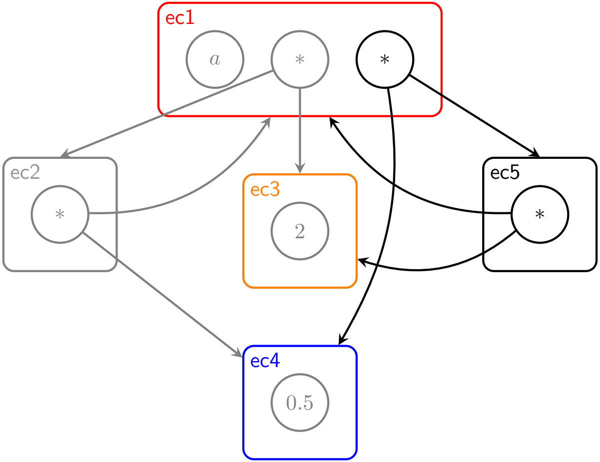
Challenge I: Illustration of Redundant Cycles

$$(x * y) * z \longrightarrow x * (y * z)$$



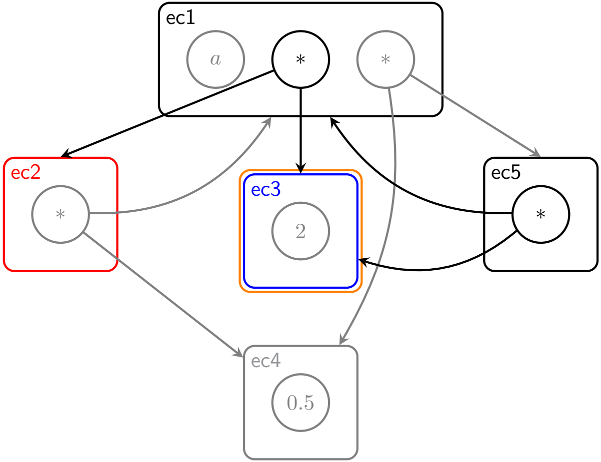
Challenge I: Illustration of Redundant Cycles

$$(x * y) * z \longrightarrow x * (y * z)$$



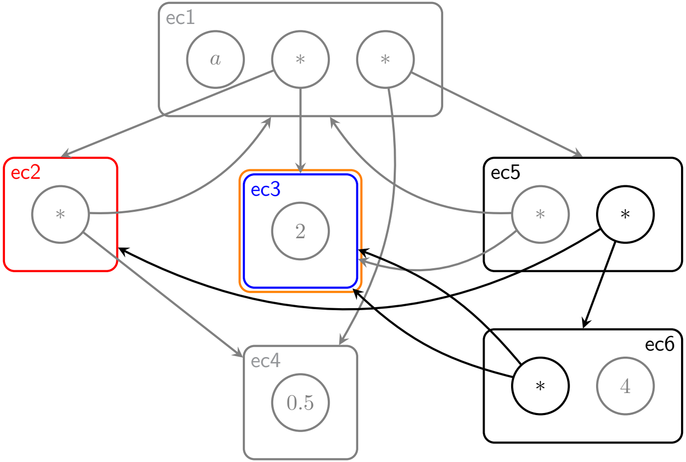
Challenge I: Illustration of Redundant Cycles

$$(x * y) * z \longrightarrow x * (y * z)$$



Challenge I: Illustration of Redundant Cycles

$$(x * y) * z \longrightarrow x * (y * z)$$



Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials

Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials
  - Tricky for deep circuits.
  - We do not want to expand the polynomials.

Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials
  - Tricky for deep circuits.
  - We do not want to expand the polynomials.
- Delegate to an external reasoner (Gröbner basis, CAD, etc.)

Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials
  - Tricky for deep circuits.
  - We do not want to expand the polynomials.
- Delegate to an external reasoner (Gröbner basis, CAD, etc.)
  - Expensive calls.
  - Theory combination issues.

Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials
  - Tricky for deep circuits.
  - We do not want to expand the polynomials.
- Delegate to an external reasoner (Gröbner basis, CAD, etc.)
  - Expensive calls.
  - Theory combination issues.
- Use heuristics to guide rewriting.

Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials
  - Tricky for deep circuits.
  - We do not want to expand the polynomials.
- Delegate to an external reasoner (Gröbner basis, CAD, etc.)
  - Expensive calls.
  - Theory combination issues.
- Use heuristics to guide rewriting.
  - Only semi-terminating

Challenge I: Existing Solutions

Countermeasures

- Standard approach: use normal forms for Polynomials
  - Tricky for deep circuits.
  - We do not want to expand the polynomials.
- Delegate to an external reasoner (Gröbner basis, CAD, etc.)
  - Expensive calls.
  - Theory combination issues.
- Use heuristics to guide rewriting.
  - Only semi-terminating

Proposal

How about detecting redundant cycles in the egraph?

Challenge I: Saturation up to Redundancy

Superposition Saturation

Consider the first-order clause

$$\forall x.\ x = f(x, a)$$

We can generate infinitely many terms:

$$f(x, a), f(f(x, a), a), f(f(f(x, a), a), a), \ldots$$

Redundancy Criterion

A clause C is redundant if it is entailed by smaller clauses $\{C_1, \ldots, C_n\}$ according to a simplification ordering $\prec$:

$$\{C_1, \ldots, C_n\} \models C \quad \text{and} \quad \forall i : C_i \prec C$$

Challenge II: Egraphs Modulo User Propagator

Why it is useful?

- Some lemmas may depend on both sides of an equality. (e.g., $P(z|x) = 0.5 \longrightarrow P(z|x) = P(z)$)
- Simplify the egraph using domain knowledge.
- More efficient reasoning for some theories.

Challenge II: Egraphs Modulo User Propagator

Why it is useful?

- Some lemmas may depend on both sides of an equality. (e.g., $P(z|x) = 0.5 \longrightarrow P(z|x) = P(z)$)
- Simplify the egraph using domain knowledge.
- More efficient reasoning for some theories.

Already exists for SAT/SMT [BEK23]

- Notify propagator on new literals
- Propagator can add new literals
- Propagator needs to explain literals

Challenge II: Egraphs Modulo User Propagator

Why it is useful?

- Some lemmas may depend on both sides of an equality. (e.g., $P(z|x) = 0.5 \longrightarrow P(z|x) = P(z)$)
- Simplify the egraph using domain knowledge.
- More efficient reasoning for some theories.

Already exists for SAT/SMT [BEK23]

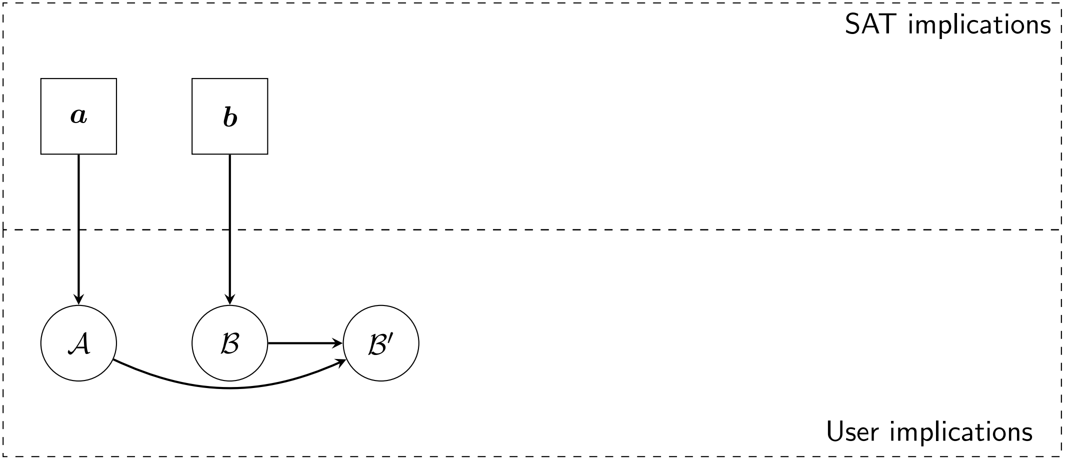
- Notify propagator on new literals
- Propagator can add new literals
- Propagator needs to explain literals

Challenges for Egraphs

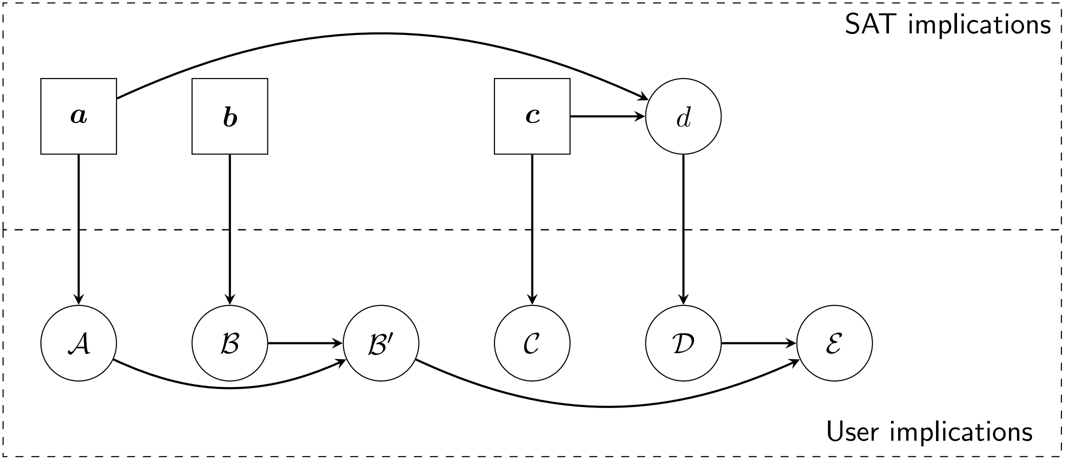
- Explosion of number of terms.
- Managing explanations over egraph changes.

SAT with User Propagators

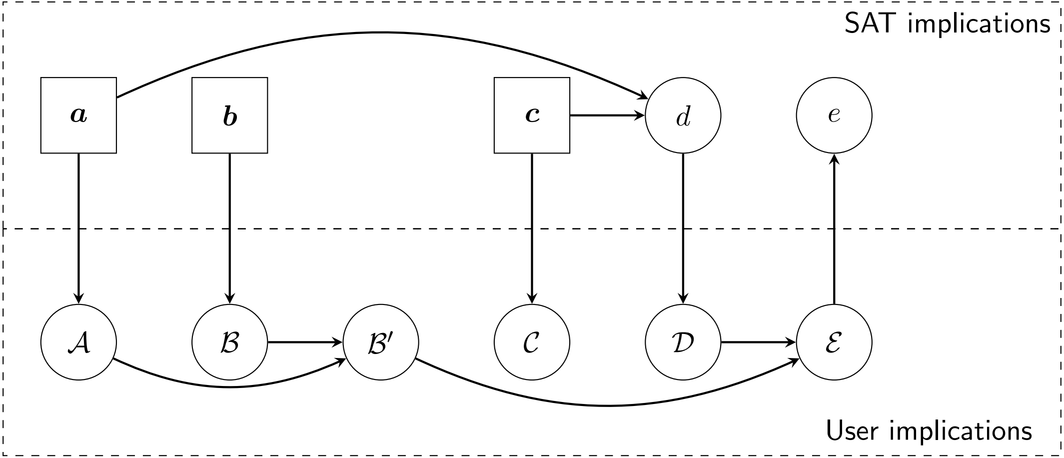
SAT with User Propagators



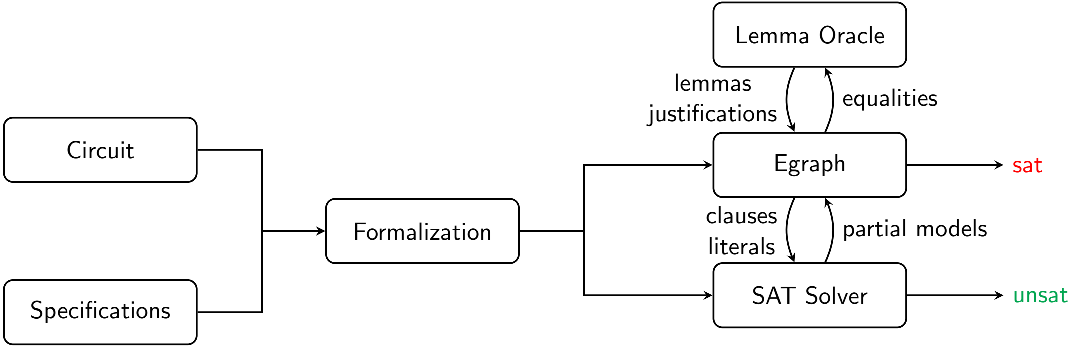
SAT with User Propagators



SAT with User Propagators



User Propagators with Egraphs



Conclusion

Summary

- Probabilities seem like a good fit for Egraphs
- Several challenges remain to be solved.
- We would like to use techniques from SMT/ATP solving within egraphs.

Challenges

- Non-linear real arithmetic requires special treatment to avoid non-termination.
- User propagators can help but require careful integration with egraphs.

Conclusion

Summary

- Probabilities seem like a good fit for Egraphs
- Several challenges remain to be solved.
- We would like to use techniques from SMT/ATP solving within egraphs.

Challenges

- Non-linear real arithmetic requires special treatment to avoid non-termination.
- User propagators can help but require careful integration with egraphs.

Discussion with the audience

- NRA: thoughts on saturation modulo redundancy?
- User propagators: any experience/ideas with egraphs?

Questions?

Thank you for your attention!

References I

Nikolaj S. Bjørner, Clemens Eisenhofer, and Laura Kovács.

Satisfiability modulo custom theories in Z3.

In *VMCAI*, volume 13881 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2023.

Franz Baader and Tobias Nipkow.

Term rewriting and all that.

Cambridge University Press, 1998.

Flaticon.

Free icons https://www.flaticon.com/free-icons/.

Accessed: 2025-06-15.

References II

Ema Salkić.

Formal verification of compositional integrity in PINI via Z3, 2023.

Max Willsey, Yisu Remy Wang, Oliver Flatt, Chandrakana Nandi, Pavel Panchekha, and Zachary Tatlock.

egg: Easy, efficient, and extensible e-graphs.

CoRR, abs/2004.03082, 2020.