universität innsbruck

SAT and SMT Solving

Sarah Winkler

Computational Logic Group
Department of Computer Science
University of Innsbruck

lecture 11
SS 2019

Outline

- Summary of Last Week
- Collision Attacks
- Nelson-Oppen Combination Method

Definition (Bit Vector Theory)

- variable $\mathbf{x}_k$ is list of length k of propositional variables $x_{k-1} \dots x_2 x_1 x_0$
- constant n_k is bit list of length k
- formulas built according to grammar

$$
\begin{aligned}
\mathit{formula} &:= (\mathit{formula} \vee \mathit{formula}) \mid (\mathit{formula} \wedge \mathit{formula}) \mid (\neg \mathit{formula}) \mid \mathit{atom} \\
\mathit{atom} &:= \mathit{term}\ \mathit{rel}\ \mathit{term} \mid \mathit{true} \mid \mathit{false} \\
\mathit{rel} &:= \ = \mid \neq \mid \geqslant_u \mid \geqslant_s \mid >_u \mid >_s \\
\mathit{term} &:= (\mathit{term}\ \mathit{binop}\ \mathit{term}) \mid (\mathit{unop}\ \mathit{term}) \mid \mathit{var} \mid \mathit{constant} \mid \mathit{term}[i{:}j] \mid \\
&\qquad (\mathit{formula}\ ?\ \mathit{term} : \mathit{term}) \\
\mathit{binop} &:= + \mid - \mid \times \mid \div_u \mid \div_s \mid \%_u \mid \%_s \mid \ll \mid \gg_u \mid \gg_s \mid \& \mid | \mid \hat{}\ \mid :: \\
\mathit{unop} &:= \sim \mid -
\end{aligned}
$$

- axioms are equality axioms plus rules for arithmetic/comparison/bitwise operations on bit vectors of length k
- solution assigns bit list of length k to variables $\mathbf{x}_k$

Remarks

- theory is decidable because carrier is finite
- common decision procedures use translation to SAT (bit blasting)
  - eager: no DPLL(T), bit-blast entire formula to SAT problem
  - lazy: second SAT solver as BV theory solver, bit-blast only BV atoms
- solvers heavily rely on preprocessing via rewriting

Definition (Bit Blasting: Formulas)

bit blasting transformation $\mathbf{B}$ transforms BV formula into propositional formula:

$$\mathbf{B}(\varphi \vee \psi) = \mathbf{B}(\varphi) \vee \mathbf{B}(\psi)$$

$$\mathbf{B}(\varphi \wedge \psi) = \mathbf{B}(\varphi) \wedge \mathbf{B}(\psi)$$

$$\mathbf{B}(\neg\varphi) = \neg\mathbf{B}(\varphi)$$

$$\mathbf{B}(t_1 \ \mathit{rel} \ t_2) = \mathbf{B}_r(u_1 \ \mathit{rel} \ u_2) \wedge \varphi_1 \wedge \varphi_2 \quad \text{if } \mathbf{B}_t(t_1) = (u_1, \varphi_1) \text{ and } \mathbf{B}_t(t_2) = (u_2, \varphi_2)$$

bit blasting $\mathbf{B}_t$ for term t returns (result u, side condition φ)

$\mathbf{B}_r$ transforms atom into propositional formula

Definition (Bit Blasting: Atoms)
for bit vectors $\mathbf{x}_k$ and $\mathbf{y}_k$ set

- equality

$$\mathbf{B}_r(\mathbf{x}_{k+1} = \mathbf{y}_{k+1}) = (x_k \leftrightarrow y_k) \wedge \cdots \wedge (x_1 \leftrightarrow y_1) \wedge (x_0 \leftrightarrow y_0)$$

- inequality

$$\mathbf{B}_r(\mathbf{x}_{k+1} \neq \mathbf{y}_{k+1}) = (x_k \oplus y_k) \vee \cdots \vee (x_1 \oplus y_1) \vee (x_0 \oplus y_0)$$

- unsigned greater-than or equal

$$\mathbf{B}_r(\mathbf{x}_1 \geqslant_u \mathbf{y}_1) = y_0 \rightarrow x_0$$

$$\mathbf{B}_r(\mathbf{x}_{k+1} \geqslant_u \mathbf{y}_{k+1}) = (x_k \wedge \neg y_k) \vee ((x_k \leftrightarrow y_k) \wedge \mathbf{B}(\mathbf{x}[k-1{:}0] \geqslant \mathbf{y}[k-1{:}0]))$$

- unsigned greater-than

$$\mathbf{B}(\mathbf{x}_k >_u \mathbf{y}_k) = \mathbf{B}(\mathbf{x}_k \geqslant \mathbf{y}_k) \wedge \mathbf{B}(\mathbf{x}_k \neq \mathbf{y}_k)$$

Definition (Bit Blasting: Bitwise Operations)
for bit vectors $\mathbf{x}_k$ and $\mathbf{y}_k$ use fresh variable $\mathbf{z}_k$ and set

- bitwise and

$$\mathbf{B}_t(\mathbf{x}_k \& \mathbf{y}_k) = (\mathbf{z}_k, \varphi) \qquad \varphi = \bigwedge_{i=0}^{k-1} z_i \leftrightarrow (x_i \wedge y_i)$$

- bitwise or

$$\mathbf{B}_t(\mathbf{x}_k | \mathbf{y}_k) = (\mathbf{z}_k, \varphi) \qquad \varphi = \bigwedge_{i=0}^{k-1} z_i \leftrightarrow (x_i \vee y_i)$$

- bitwise exclusive or

$$\mathbf{B}_t(\mathbf{x}_k \;\hat{}\; \mathbf{y}_k) = (\mathbf{z}_k, \varphi) \qquad \varphi = \bigwedge_{i=0}^{k-1} z_i \leftrightarrow (x_i \oplus y_i)$$

- bitwise negation

$$\mathbf{B}_t(-\mathbf{x}_k) = (\mathbf{z}_k, \varphi) \qquad \varphi = \bigwedge_{i=0}^{k-1} z_i \leftrightarrow \neg x_i$$

Definition (Bit Blasting: Addition and Subtraction)

- addition

$$\mathbf{B}_t(\mathbf{x}_k + \mathbf{y}_k) = (\mathbf{s}_k, \varphi)$$

where

$$\varphi = (c_0 \leftrightarrow x_0 \wedge y_0) \wedge (s_0 \leftrightarrow x_0 \oplus y_0) \wedge$$
$$\bigwedge_{i=1}^{k-1} (c_i \leftrightarrow \min2(x_i, y_i, c_{i-1})) \wedge (s_i \leftrightarrow x_i \oplus y_i \oplus c_{i-1})$$

for fresh variables $\mathbf{s}_k$ and $\mathbf{c}_k$ and $\min2(a, b, d) = (a \wedge b) \vee (a \wedge d) \vee (b \wedge d)$

ripple-carry adder: $\mathbf{c}_k$ are carry bits

- unary minus

$$\mathbf{B}_t(-\mathbf{x}_k) = \mathbf{B}_t(\sim \mathbf{x}_k + \mathbf{1}_k)$$

- subtraction

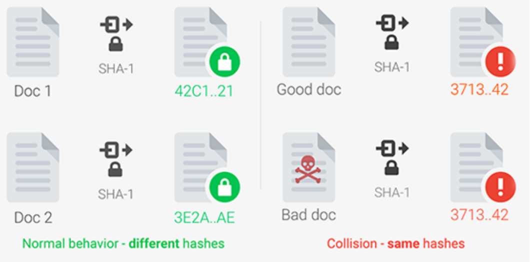
$$\mathbf{B}_t(\mathbf{x}_k + \mathbf{y}_k) = \mathbf{B}_t(\mathbf{x}_k + (-\mathbf{y}_k))$$

Outline

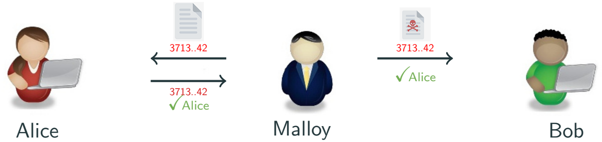
- Summary of Last Week
- Collision Attacks
- Nelson-Oppen Combination Method

Cryptographic Hash Functions

- cryptographic hash function f is one-way hash function (SHA-1, MD5, ...)
- considered infeasible to invert, and to find messages with same hash
- problem: hash collisions



Classical Collision Attack Scenario



- Malloy aims to send malicious document to Bob pretending it be from Alice

Cryptographic Hash Function

(currently) practically infeasible to invert

- maps data of arbitrary size to bit string of fixed size (hash value)
- is one-way function

Example

SHA-0, SHA-1, SHA-256, MD5, MD6, BLAKE2, RIPEMD-160, ...

Collision Attack: Shift-Add-Xor Hash

- widely used non-cryptographic string hash function
- given string s, compute hash `sax`(s)

```
unsigned sax(char *s, int len){
    unsigned h = 0;
    for (int i = 0; i < len; i++)
        h = h ^ ((h << 5) + (h >> 2) + s[i]);
    return h;
}
```

- collision attack: `sax_collision.py`

More Cryptanalysis using SAT/SMT

- collision attacks (preimage attacks) for current hash functions such as MD4, MD5, SHA-256, CryptoHash, Keccak, ...
- exhibit classes of weak keys (or prove their absence) for block ciphers such as IDEA, WIDEA-*n*, or MESH-8
- solve inversion problems, e.g. for 20 bit DES key
- reason about crypto primitives
- help prove complexity bounds of certain operations

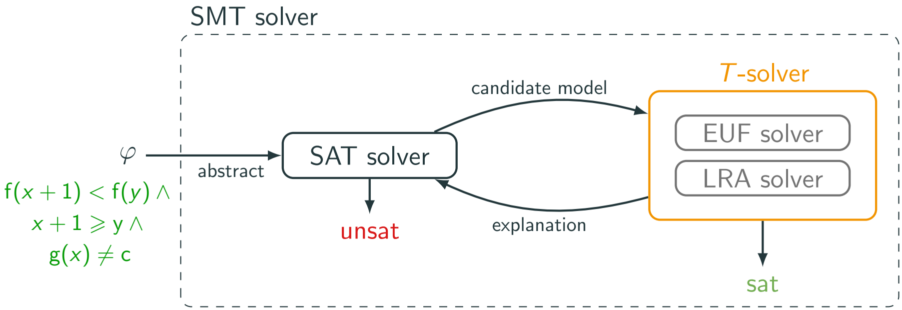
Tools for SAT/SMT-Based Cryptanalysis

- CryptoMiniSat
- CryptoSMT
- Transalg
- ...

Outline

- Summary of Last Week
- Collision Attacks
- Nelson-Oppen Combination Method
  - Nondeterministic Version
  - Deterministic Version

How to Be Lazy



Theory T	T**-solving method**	
▶ equality logic	equality graphs	✓
▶ equality + uninterpreted functions (EUF)	congruence closure	✓
▶ linear arithmetic (LRA and LIA)	DPLL(T) Simplex (+ cuts)	✓
▶ bitvectors (BV)	bit-blasting	✓

Theory combinations Nelson-Oppen method

Definitions

- (first-order) theory T consists of
  - signature Σ: set of function and predicate symbols
  - axioms $\mathcal{A}$: set of sentences in first-order logic in which only function and predicate symbols of Σ appear
- theory is stably infinite if every satisfiable quantifier-free formula has model with infinite carrier set

Definition

theory combination $T_1 \oplus T_2$ of two theories

- T_1 over signature Σ_1
- T_2 over signature Σ_2

has signature $\Sigma_1 \cup \Sigma_2$ and axioms $T_1 \cup T_2$

Outline

- Summary of Last Week
- Collision Attacks
- Nelson-Oppen Combination Method
  - Nondeterministic Version
  - Deterministic Version

Example

combination of linear arithmetic and uninterpreted functions:

$$x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge \ z \geqslant 0$$

Assumptions

two stably infinite theories

- T_1 over signature Σ_1
- T_2 over signature Σ_2

such that

- $\Sigma_1 \cap \Sigma_2 = \{=\}$
- T_1-satisfiability of quantifier-free Σ_1-formulas is decidable
- T_2-satisfiability of quantifier-free Σ_2-formulas is decidable

Nelson-Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

1. purification

$$\varphi \;\approx\; \varphi_1 \wedge \varphi_2 \quad \text{for } \Sigma_1\text{-formula } \varphi_1 \text{ and } \Sigma_2\text{-formula } \varphi_2$$

2. guess and check
   - V is set of shared variables in φ_1 and φ_2
   - guess equivalence relation E on V
   - arrangement $\alpha(V, E)$ is formula

$$\bigwedge_{x \mathrel{E} y} x = y \quad \wedge \quad \bigwedge_{\neg(x \mathrel{E} y)} x \neq y$$

   - if $\varphi_1 \wedge \alpha(V, E)$ is T_1-satisfiable and $\varphi_2 \wedge \alpha(V, E)$ is T_2-satisfiable then return satisfiable else return unsatisfiable

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ y = 1 \ \wedge\ z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:
  1. $\{\{x, y, z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
  2. $\{\{x, y\}, \{z\}\}$ — $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
  3. $\{\{x, z\}, \{y\}\}$ — $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
  4. $\{\{x\}, \{y, z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
  5. $\{\{x\}, \{y\}, \{z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
- φ is unsatisfiable

Outline

- Summary of Last Week
- Collision Attacks
- Nelson-Oppen Combination Method
  - Nondeterministic Version
  - Deterministic Version

Definition

theory T is convex if

$$F \vDash_T \bigvee_{i=1}^{n} u_i = v_i \qquad \text{implies} \qquad (F \vDash_T u_i = v_i \quad \text{for some } 1 \leqslant i \leqslant n)$$

$\forall$ quantifier-free conjunctive formula F and variables $u_1, \ldots, u_n, v_1, \ldots, v_n$

Example

- linear arithmetic over integers (LIA) is not convex:

  $$1 \leqslant x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2 \quad \vDash_T \quad x = y \ \vee \ x = z$$

  holds but none of

  $$1 \leqslant x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2 \quad \vDash_T \quad x = y$$
  $$1 \leqslant x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2 \quad \vDash_T \quad x = z$$

- linear arithmetic over rationals and reals (LRA) is convex
- equality logic with uninterpreted functions (EUF) is convex

Example

consider φ over combination of LRA and EUF:

$$x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \;\wedge\; z \geqslant 0$$

- first purify φ:

$$\varphi_1: \; x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge$$

$$\varphi_2: \; \mathsf{f}(w_1) \neq \mathsf{f}(z) \;\wedge\; w_2 =$$

test all (finitely many) equations,
or T-propagation

- compute implied equalities between shared variables:

$$E: \; x = y \;\wedge\; w_2 = w_3 \;\wedge\; z = w_1$$

- test satisfiability of $\varphi_2 \wedge E$ in EUF and compute implied equalities

$$\varphi_2 \;\wedge\; E \;\Longrightarrow\; \bot$$

- φ is unsatisfiable

Nelson-Oppen Method: Deterministic Version

Input quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories T_1 and T_2

Output satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   E: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge E$ (and add implied equations)
   - if $\varphi_1 \wedge E$ is T_1-unsatisfiable then return unsatisfiable
   - else add new implied equalities to E
4. test satisfiability of $\varphi_2 \wedge E$ (and add implied equations)
   - if $\varphi_2 \wedge E$ is T_2-unsatisfiable then return unsatisfiable
   - else add new implied equalities to E
5. if E has been extended in steps 3 or 4 then go to step 2
   else return satisfiable

Remark

Nelson-Oppen decision procedure can be extended to non-convex theories: case-splitting for implied disjunction of equalities

Example

consider φ over combination of LIA and EUF:

$$1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(2)$$

- first purify φ:

$$\varphi_1 : \ 1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ w_1 = 1 \ \wedge \ w_2 = 2$$
$$\varphi_2 : \ f(x) \neq f(w_1) \ \wedge \ f(x) \neq f(w_2)$$

- check satisfiability and compute (disjunction of) implied equalities:

$$E : \ x = w_2$$

- test satisfiability of $\varphi_2 \wedge E$ in EUF

$$\varphi_2 \ \wedge \ E \ \implies \ \bot$$

- case split: $x = w_1$ or $x = w_2$
- φ is unsatisfiable

Application: Checking Program Equivalence

Relevance of Program Equivalence

correctness of compiler optimizations, software verification

Example (Are the following two programs equivalent?)

```
int one(int x){
  unsigned z = x & (-1);
  unsigned y = z * 2;
  return foo(z) + y;
}
```

```
int two(int x){
  return foo(x) + (x << 1);
}
```

uninterpreted functions

bit vectors

Assert non-equivalence by SMT encoding:

$$\mathbf{one}_{32} = \mathrm{foo}(\mathbf{z}_{32}) + \mathbf{y}_{32} \land \mathbf{z}_{32} = \mathbf{x}_{32} \,\&\, (-\mathbf{1})_{32} \land \mathbf{y}_{32} = \mathbf{z}_{32} \times \mathbf{2}_{32} \land$$
$$\mathbf{two}_{32} = \mathrm{foo}(\mathbf{x}_{32}) + (\mathbf{x}_{32} \ll \mathbf{1}_{32}) \land$$
$$\mathbf{one}_{32} \neq \mathbf{two}_{32}$$

Remarks

- useful to combine BV and EUF theories
- checking equivalence of programs with loops is more challenging

Bibliography

Greg Nelson and Derek C. Oppen
Simplification by Cooperating Decision Procedures
ACM Transactions on Programming Languages and Systems 2(1), pp 245–257, 1979.

Nuno P. Lopes and José Monteiro.
Automatic equivalence checking of programs with uninterpreted functions and integer arithmetic.
International Journal on Software Tools for Technology Transfer 18(4), pp 359–374, 2016.