

# SAT and SMT Solving

---

Sarah Winkler

SS 2018

Department of Computer Science

University of Innsbruck

- Summary of Last Week
- Nelson-Oppen Combination Method

## Summary of Last Week

---

## 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

$formula := (formula \vee formula) \mid (formula \wedge formula) \mid (\neg formula) \mid atom$

$atom := term \ rel \ term \mid true \mid false$

$rel := = \mid \neq \mid \geq_u \mid \geq_s \mid >_u \mid >_s$

$term := (term \ binop \ term) \mid (unop \ term) \mid var \mid constant \mid term[i:j] \mid$   
 $(formula \ ? \ term : \ term)$

$binop := + \mid - \mid \times \mid \div_u \mid \div_s \mid \%_u \mid \%_s \mid \ll \mid \gg_u \mid \gg_s \mid \& \mid \mid \mid \wedge \mid ::$

$unop := \sim \mid -$

- ▶ 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:

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

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

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

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

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

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

## Definition (Bit Blasting: Atoms)

for bit vectors  $\mathbf{x}_k$  and  $\mathbf{y}_k$  set

- ▶ equality

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

- ▶ inequality

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

- ▶ unsigned greater-than or equal

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

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

- ▶ unsigned greater-than

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

## 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

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

- ▶ bitwise or

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

- ▶ bitwise exclusive or

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

- ▶ bitwise negation

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

## Definition (Bit Blasting: Addition and Subtraction)

- ▶ addition

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

where

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

ripple-carry adder:  
 $c_k$  are carry bits

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

- ▶ unary minus

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

- ▶ subtraction

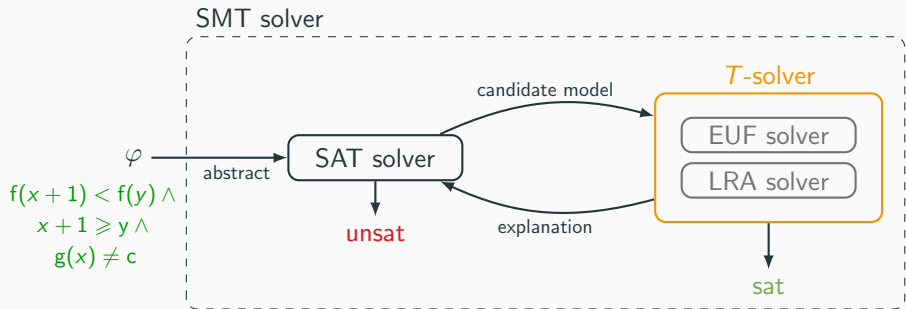
$$\mathbf{B}_t(\mathbf{x}_k + \mathbf{y}_k) = \mathbf{B}_t(\mathbf{x}_k + (-\mathbf{y}_k))$$



# Nelson-Oppen Combination Method

---

# How to Be Lazy



## Theory $T$

- ▶ equality logic
- ▶ equality + uninterpreted functions (EUF)
- ▶ linear arithmetic (LRA and LIA)
- ▶ bitvectors (BV)

## $T$ -solving method

- equality graphs ✓
- congruence closure ✓
- DPLL( $T$ ) Simplex (+ cuts) ✓
- bit-blasting ✓

## Theory combinations

## Nelson-Oppen method

## Definitions

- ▶ (first-order) **theory** consists of
  - ▶ signature  $\Sigma$ : set of function and predicate symbols
  - ▶ axioms  $T$ : set of sentences in first-order logic in which only function and predicate symbols of  $\Sigma$  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  $\Sigma_1$
- ▶  $T_2$  over signature  $\Sigma_2$

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

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

## Example

combination of linear arithmetic and uninterpreted functions:

$$x \geq y \wedge y - z \geq x \wedge f(f(y) - f(x)) \neq f(z) \wedge z \geq 0$$

## Assumptions

two stably infinite theories

- ▶  $T_1$  over signature  $\Sigma_1$
- ▶  $T_2$  over signature  $\Sigma_2$

such that

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

# Nelson-Oppen Method: Nondeterministic Version

*Input* quantifier-free conjunction  $\varphi$  in theory combination  $T_1 \oplus T_2$

*Output* satisfiable or unsatisfiable

## 1 purification

$\varphi \approx \varphi_1 \wedge \varphi_2$  for  $\Sigma_1$ -formula  $\varphi_1$  and  $\Sigma_2$ -formula  $\varphi_2$

## 2 guess and check

- ▶  $V$  is set of shared variables in  $\varphi_1$  and  $\varphi_2$
- ▶ guess equivalence relation  $E$  on  $V$
- ▶ **arrangement**  $\alpha(V, E)$  is formula

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

- ▶ 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  $\varphi$  in combination of LIA and EUF:

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

- ▶  $V = \{x, y, z\}$
- ▶ 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
- ▶  $\varphi$  is unsatisfiable

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



## Definition

theory  $T$  is **convex** if

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

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

## Example

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

$$1 \leq x \leq 2 \wedge y = 1 \wedge z = 2 \models_T x = y \vee x = z$$

holds but none of

$$1 \leq x \leq 2 \wedge y = 1 \wedge z = 2 \models_T x = y$$

$$1 \leq x \leq 2 \wedge y = 1 \wedge z = 2 \models_T x = z$$

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

## Example

combination of LRA and EUF:

$$x \geq y \wedge y - z \geq x \wedge f(f(y) - f(x)) \neq f(z) \wedge z \geq 0$$

purification

$$\varphi_1: x \geq y \wedge y - z \geq x \wedge w_1 = w_2 - w_3 \wedge z \geq 0$$

$$\varphi_2: f(w_1) \neq f(z) \wedge w_2 = f(y) \wedge w_3 = f(x)$$

implied equalities between shared variables

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

test satisfiability of  $\varphi_2 \wedge E$

$$\varphi_2 \wedge E \implies \perp$$

unsatisfiable

## Nelson-Oppen Method: Deterministic Version

Input     quantifier-free conjunction  $\varphi$  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  $\Sigma_1$ -formula  $\varphi_1$  and  $\Sigma_2$ -formula  $\varphi_2$
- 2   **V**: set of shared variables in  $\varphi_1$  and  $\varphi_2$   
      **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

combination of LIA and EUF:

$$1 \leq x \wedge x \leq 2 \wedge f(x) \neq f(1) \wedge f(x) \neq f(2)$$

purification

$$\varphi_1: 1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2$$

$$\varphi_2: f(x) \neq f(w_1) \wedge f(x) \neq f(w_2)$$

implied equalities between shared variables

$$E: x = w_2$$

test satisfiability of  $\varphi_2 \wedge E \implies \perp$

# Application: Checking Program Equivalence

## Relevance of Program Equivalence

correctness of compiler optimizations, regression testing, 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;  
}
```

uninterpreted functions

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

bit vectors

Assert non-equivalence by SMT encoding:

$$\mathbf{one}_{32} = \mathbf{foo}(\mathbf{z}_{32}) + \mathbf{y}_{32} \wedge \mathbf{z}_{32} = \mathbf{x}_{32} \& (-\mathbf{1})_{32} \wedge \mathbf{y}_{32} = \mathbf{z}_{32} \times \mathbf{2}_{32} \wedge$$

$$\mathbf{two}_{32} = \mathbf{foo}(\mathbf{x}_{32}) + (\mathbf{x}_{32} \ll \mathbf{1}_{32}) \wedge$$

$$\mathbf{one}_{32} \neq \mathbf{two}_{32}$$

## Remarks

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



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.