

# Course Formal Methods

## Module I: Automated Reasoning

### Ch. 03: Satisfiability Modulo Theories

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it

URL: <http://disi.unitn.it/rseba/DIDATTICA/fm2021/>

Teaching assistant: **Giuseppe Spallitta** – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems  
Academic year 2020-2021

last update: Tuesday 13<sup>th</sup> April, 2021

Copyright notice: *some material (text, figures) displayed in these slides is courtesy of R. Alur, M. Benerecetti, A. Cimatti, M. Di Natale, P. Pandya, M. Pistore, M. Roveri, and S. Tonetta, who detain its copyright. Some examples displayed in these slides are taken from [Clarke, Grunberg & Peled, "Model Checking", MIT Press], and their copyright is detained by the authors. All the other material is copyrighted by Roberto Sebastiani. Every commercial use of this material is strictly forbidden by the copyright laws without the authorization of the authors. No copy of these slides can be displayed in public without containing this copyright notice.*

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

- 1 Introduction
  - **What is a Theory?**
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

# FOL Theories

## Traditional Definition (FOL)

Given a FOL signature  $\Sigma$ , a  $\Sigma$ -Theory  $\mathcal{T}$  (hereafter simply “theory”) is a (possibly infinite) set of FOL closed formulas (axioms)

- Typically used to provide some **intended interpretation** to the symbols in the signature  $\Sigma$
- FOL formulas deduces from these axioms via inference rules
- Definition used by logicians,
- Very low practical use in AR & Formal Verification

# Example: A FOL Theory of Positive Integer Numbers (aka “Peano Arithmetic”, $\mathcal{P}$ )

## • Signature

- (basic) unary predicate symbol:  $\text{NatNum}$  (“natural number”)
- (basic) unary function symbol:  $S$  (“successor”)
- (basic) constant symbol:  $0$
- (derived) binary function symbols:  $+$ ,  $*$  (infix)
- (derived) constant symbols:  $1, 2, 3, 4, 5, 6, \dots$

## • Axioms

- 1  $\text{NatNum}(0)$
- 2  $\forall x. (\text{NatNum}(x) \rightarrow \text{NatNum}(S(x)))$
- 3  $\forall x. (\text{NatNum}(x) \rightarrow (0 \neq S(x)))$
- 4  $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow ((x \neq y) \rightarrow (S(x) \neq S(y))))$
- 5  $\forall x. (\text{NatNum}(x) \rightarrow (x = (0 + x)))$
- 6  $\forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow (S(x) + y) = S(x + y))$
- 7  $1 = S(0), 2 = S(1), 3 = S(2), \dots$

## • Formulas deduced

- ex:  $\mathcal{P} \vdash \text{NatNum}(25)$
- ex:  $\mathcal{P} \vdash \forall x, y. ((\text{NatNum}(x) \wedge \text{NatNum}(y)) \rightarrow ((x + y) = (y + x)))$

# FOL Theories (cont.)

## SMT Definition

Given a FOL signature  $\Sigma$ , a  $\Sigma$ -Theory  $\mathcal{T}$  (hereafter simply “theory”) is one (or more) model(s) constraining the interpretations of  $\Sigma$

- Provides an **intended interpretation** to the symbols in  $\Sigma$ 
  - constants mapped into domain elements
    - ex: “1” mapped into the number one
  - predicate symbols mapped into relations on domain elements
    - ex: “. < .” mapped into the arithmetical relation “less then”
  - function symbols mapped into functions on domain elements
    - ex: “S(.)” mapped into the arithmetical function “successor of”

These symbols are called **interpreted**

- Compliant with previous definition: **model(s) satisfying all axioms**
- Ad hoc “ $\mathcal{T}$ -aware” decision procedures for reasoning on formulas
- Very effective in practical applications

## Example: Linear Arithmetic on the Integers ( $\mathcal{LIA}$ )

- Domain: integer numbers
- Numerical constants interpreted as **numbers**
  - ex: “1”, “1346231” mapped directly into the corresponding number
- function and predicates interpreted as **arithmetical operations**
  - “+” as addition, “\*” as multiplication, “<” as less-then, . etc.
- **ILP solvers** used to do logical reasoning
  - ex:  $(3x - 2y \leq 3) \wedge (4y - 2z < -7) \models (6x - 2z < -1)$



# Satisfiability, Validity, Entailment (Modulo a Theory $\mathcal{T}$ )

## Definitions

- Idea: **We restrict to models satisfying  $\mathcal{T}$**  (“ $\mathcal{T}$ -models”)
- A formula is **satisfiable in  $\mathcal{T}$**  (aka “ $\varphi$  is  $\mathcal{T}$ -satisfiable”) iff some model satisfying  $\mathcal{T}$  satisfies also  $\varphi$ 
  - ex:  $(x < 3)$  satisfiable in  $\mathcal{LIA}$
- A formula  $\varphi$  is **valid in  $\mathcal{T}$**  (aka “ $\varphi$  is  $\mathcal{T}$ -valid” or “ $\models_{\mathcal{T}} \varphi$ ”) iff all models satisfying  $\mathcal{T}$  satisfy also  $\varphi$ 
  - ex:  $(x < 3) \rightarrow (x < 4)$  valid in  $\mathcal{LIA}$
- A formula  $\varphi$  **entails  $\psi$  in  $\mathcal{T}$**  (aka “ $\varphi$   $\mathcal{T}$ -entails  $\psi$ ” or “ $\varphi \models_{\mathcal{T}} \psi$ ”) iff all models satisfying  $\mathcal{T}$  and  $\varphi$  satisfy also  $\psi$ 
  - ex:  $(x < 3) \models_{\mathcal{LIA}} (x < 4)$

## Properties

- $\varphi$  is  $\mathcal{T}$ -valid iff  $\neg\varphi$  is  $\mathcal{T}$ -unsatisfiable
  - $\varphi \models_{\mathcal{T}} \psi$  iff  $\varphi \rightarrow \psi$  is  $\mathcal{T}$ -valid
- $\implies \varphi \models_{\mathcal{T}} \psi$  iff  $\varphi \wedge \neg\psi$   $\mathcal{T}$ -unsatisfiable

- 1 Introduction
  - What is a Theory?
  - **Satisfiability Modulo Theories**
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

# Satisfiability Modulo Theories (SMT( $\mathcal{T}$ ))

## Satisfiability Modulo Theories (SMT( $\mathcal{T}$ ))

The problem of deciding the satisfiability of (typically quantifier-free) formulas in some decidable first-order theory  $\mathcal{T}$

- $\mathcal{T}$  can also be a **combination of theories**  $\bigcup_i \mathcal{T}_i$ .

# SMT( $\mathcal{T}$ ): Theories of Interest

Some theories of interest (e.g., for formal verification)

- Equality and Uninterpreted Functions ( $\mathcal{EUF}$ ):  
 $((x = y) \wedge (y = f(z))) \rightarrow (g(x) = g(f(z)))$
- Difference logic ( $\mathcal{DL}$ ):  $((x = y) \wedge (y - z \leq 4)) \rightarrow (x - z \leq 6)$
- UTVPI ( $\mathcal{UTVPI}$ ):  $((x = y) \wedge (y - z \leq 4)) \rightarrow (x + z \leq 6)$
- Linear arithmetic over the rationals ( $\mathcal{LRA}$ ):  
 $(T_\delta \rightarrow (s_1 = s_0 + 3.4 \cdot t - 3.4 \cdot t_0)) \wedge (\neg T_\delta \rightarrow (s_1 = s_0))$
- Linear arithmetic over the integers ( $\mathcal{LIA}$ ):  
 $(x = x_l + 2^{16}x_h) \wedge (x \geq 0) \wedge (x \leq 2^{16} - 1)$
- Arrays ( $\mathcal{AR}$ ):  $(i = j) \vee read(write(a, i, e), j) = read(a, j)$
- Bit vectors ( $\mathcal{BV}$ ):  
 $x_{[16]}[15 : 0] = (y_{[16]}[15 : 8] :: z_{[16]}[7 : 0]) \ll w_{[8]}[3 : 0]$
- Non-Linear arithmetic over the reals ( $\mathcal{NLA}(\mathbb{R})$ ):  
 $((c = a \cdot b) \wedge (a_1 = a - 1) \wedge (b_1 = b + 1)) \rightarrow (c = a_1 \cdot b_1 + 1)$
- ...

# Satisfiability Modulo Theories (SMT( $\mathcal{T}$ )): Example

## Example: SMT( $\mathcal{LIA} \cup \mathcal{EUF} \cup \mathcal{AR}$ )

$$\varphi \stackrel{\text{def}}{=} (d \geq 0) \wedge (d < 1) \wedge ((f(d) = f(0)) \rightarrow (\text{read}(\text{write}(V, i, x), i + d) = x + 1))$$

- involves arithmetical, arrays, and uninterpreted function/predicate symbols, plus Boolean operators
  - Is it satisfiable?
  - No:

$$\begin{aligned} & \varphi \\ \implies_{\mathcal{LIA}} & (d = 0) \\ \implies_{\mathcal{EUF}} & (f(d) = f(0)) \\ \implies_{\text{Bool}} & (\text{read}(\text{write}(V, i, x), i + d) = x + 1) \\ \implies_{\mathcal{LIA}} & (\text{read}(\text{write}(V, i, x), i) = x + 1) \\ \implies_{\mathcal{LIA}} & \neg(\text{read}(\text{write}(V, i, x), i) = x) \\ \implies_{\mathcal{AR}} & \perp \end{aligned}$$

# SMT and SMT solvers

## Common fact about SMT problems from various applications

SMT requires capabilities for **heavy Boolean reasoning** combined with capabilities for **reasoning in expressive decidable F.O. theories**

- SAT alone not expressive enough
- standard automated theorem proving inadequate (e.g., arithmetic)
- may involve also numerical computation (e.g., simplex)

## Modern SMT solvers

- combine **SAT solvers** with  $\mathcal{T}$ -specific **decision procedures** (**theory solvers** or  **$\mathcal{T}$ -solvers**)
  - contributions from SAT, Automated Theorem Proving (ATP), formal verification (FV) and operational research (OR)

## Notational remark (1): most/all examples in $\mathcal{LRA}$

For better readability, in most/all the examples of this presentation we will use the theory of linear arithmetic on rational numbers ( $\mathcal{LRA}$ ) because of its intuitive semantics. E.g.:

$$(\neg A_1 \vee (3x_1 - 2x_2 - 3 \leq 5)) \wedge (A_2 \vee (-2x_1 + 4x_3 + 2 = 3))$$

Nevertheless, analogous examples can be built with all other theories of interest.

## Notational remark (2): “constants” vs. “variables”

- Consider, e.g., the formula:  
 $(\neg A_1 \vee (3x_1 - 2x_2 - 3 \leq 5)) \wedge (A_2 \vee (-2x_1 + 4x_3 + 2 = 3))$
- How do we call  $A_1, A_2$ ?:
  - (a) Boolean/propositional **variables**?
  - (b) uninterpreted **0-ary predicates**?
- How do we call  $x_1, x_2, x_3$ ?:
  - (a) domain **variables**?
  - (b) uninterpreted Skolem **constants/0-ary uninterpreted functions**?
- Hint:
  - (a) typically used in SAT, CSP and OR communities
  - (b) typically used in logic & ATP communities

Hereafter we call  $A_1, A_2$  “Boolean/propositional **variables**” and  $x_1, x_2, x_3$  “domain **variables**” (logic purists, please forgive me!)



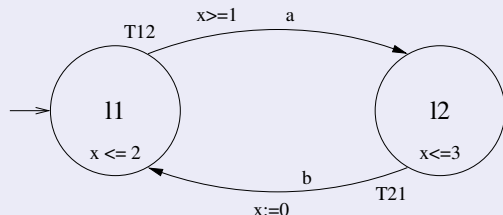
- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - **Motivations and Goals of SMT**
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

# Some Motivating Applications

Interest in SMT triggered by some real-world applications

- Verification of Hybrid & Timed Systems
- Verification of RTL Circuit Designs & of Microcode
- SW Verification
- Planning with Resources
- Temporal reasoning
- Scheduling
- Compiler optimization
- ...

# Verification of Timed Systems



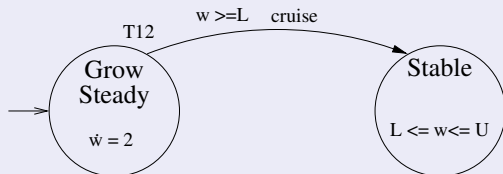
- Bounded/inductive model checking of Timed Systems [6, 33, 53],

...

- Timed Automata encoded into  $\mathcal{T}$ -formulas:
  - **discrete information** (locations, transitions, events) with Boolean vars.
  - **timed information** (clocks, elapsed time) with differences ( $t_3 - x_3 \leq 2$ ), equalities ( $x_4 = x_3$ ) and linear constraints ( $t_8 - x_8 = t_2 - x_2$ ) on  $\mathbb{Q}$

⇒ SMT on  $\mathcal{DL}(\mathbb{Q})$  or  $\mathcal{LRA}$  required

# Verification of Hybrid Systems ...



- Bounded model checking of Hybrid Systems [5],...
  - Hybrid Automata encoded into  $\mathcal{L}$ -formulas:
    - discrete information (locs, trans., events) with Boolean vars.
    - timed information (clocks, elapsed time) with differences ( $t_3 - x_3 \leq 2$ ), equalities ( $x_4 = x_3$ ) and linear constraints ( $t_8 - x_8 = t_2 - x_2$ ) on  $\mathbb{Q}$
    - Evolution of Physical Variables (e.g., speed, pressure) with linear ( $\omega_4 = 2\omega_3$ ) and non-linear constraints ( $P_1 V_1 = 4T_1$ ) on  $\mathbb{Q}$
  - Undecidable under simple hypotheses!
- ⇒ SMT on  $\mathcal{DL}(\mathbb{Q})$ ,  $\mathcal{LRA}$  or  $\mathcal{NLCA}(\mathbb{R})$  required

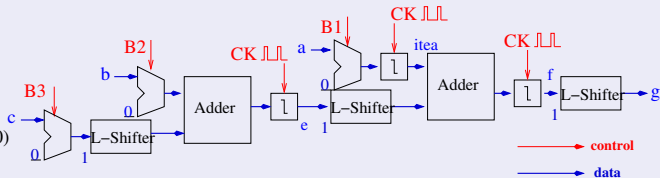
# Verification of HW circuit designs & microcode

$$g = 2 * f$$

$$f = itea + 2 * e$$

$$itea' = ITE(B1; a; 0)$$

$$e' = ITE(B2; b; 0) + 2 * ite(B3; c; 0)$$



- SAT/SMT-based **Model Checking & Equiv. Checking** of RTL designs, **symbolic simulation** of  $\mu$ -code [24, 21, 39]
  - **Control paths** handled by Boolean reasoning
  - **Data paths** information abstracted into theory-specific terms
    - **words** (bit-vectors, integers,  $\mathcal{EUF}$  vars, ... ):  $a[31 : 0]$ ,  $a$
    - **word operations**: ( $\mathcal{BV}$ ,  $\mathcal{EUF}$ ,  $\mathcal{AR}$ ,  $\mathcal{LIA}$ ,  $\mathcal{NLA}(\mathbb{Z})$  operators)
 
$$x_{[16]}[15 : 0] = (y_{[16]}[15 : 8] :: z_{[16]}[7 : 0]) \ll w_{[8]}[3 : 0],$$

$$(a = a_L + 2^{16} a_H), (m_1 = store(m_0, l_0, v_0)), \dots$$
  - Trades **heavy Boolean reasoning** ( $\approx 2^{64}$  factors) with  **$\mathcal{T}$ -solving**
- $\Rightarrow$  SMT on  $\mathcal{BV}$ ,  $\mathcal{EUF}$ ,  $\mathcal{AR}$ , modulo- $\mathcal{LIA}$  [ $\mathcal{NLA}(\mathbb{Z})$ ] required

# Verification of SW systems

```
...
10. i = 0;
11. acc = 0.0;
12. while (i < dim) {
13.     acc += V[i];
14.     i++;
15. }
...
```

```
....
( $pc = 10$ )  $\rightarrow ((i' = 0) \wedge (pc' = 11))$ 
( $pc = 11$ )  $\rightarrow ((acc' = 0.0) \wedge (pc' = 12))$ 
( $pc = 12$ )  $\rightarrow ((i < dim) \rightarrow \wedge (pc' = 13))$ 
( $pc = 12$ )  $\rightarrow (\neg(i < dim) \rightarrow \wedge (pc' = 16))$ 
( $pc = 13$ )  $\rightarrow ((acc' = acc + read(V, i)) \wedge (pc' = 14))$ 
( $pc = 14$ )  $\rightarrow (i' = i + 1) \wedge (pc' = 15)$ 
( $pc = 15$ )  $\rightarrow (pc' = 16)$ 
...
```

- Verification of SW code

- BMC, K-induction, Check of proof obligations, interpolation-based model checking, symbolic simulation, concolic testing, ...

$\Rightarrow$  SMT on  $BV$ ,  $\mathcal{EUF}$ ,  $\mathcal{AR}$ , (modulo-) $\mathcal{LIA}$  [ $\mathcal{NLA}(\mathbb{Z})$ ] required

## Planning with Resources [72]

- SAT-bases planning augmented with numerical constraints
- Straightforward to encode into SMT( $\mathcal{LRA}$ )

### Example (sketch) [72]

<i>(Deliver)</i>	$\wedge$ // goal
<i>(MaxLoad)</i>	$\wedge$ // load constraint
<i>(MaxFuel)</i>	$\wedge$ // fuel constraint
<i>(Move <math>\rightarrow</math> MinFuel)</i>	$\wedge$ // move requires fuel
<i>(Move <math>\rightarrow</math> Deliver)</i>	$\wedge$ // move implies delivery
<i>(GoodTrip <math>\rightarrow</math> Deliver)</i>	$\wedge$ // a good trip requires
<i>(GoodTrip <math>\rightarrow</math> AllLoaded)</i>	$\wedge$ // a full delivery
<hr/>	
<i>(MaxLoad <math>\rightarrow</math> (load <math>\leq</math> 30))</i>	$\wedge$ // load limit
<i>(MaxFuel <math>\rightarrow</math> (fuel <math>\leq</math> 15))</i>	$\wedge$ // fuel limit
<i>(MinFuel <math>\rightarrow</math> (fuel <math>\geq</math> 7 + 0.5load))</i>	$\wedge$ // fuel constraint
<i>(AllLoaded <math>\rightarrow</math> (load = 45))</i>	//

## (Disjunctive) Temporal Reasoning [69, 2]

- Temporal reasoning problems encoded as disjunctions of difference constraints

$$((x_1 - x_2 \leq 6) \quad \vee \quad (x_3 - x_4 \leq -2)) \quad \wedge$$

$$((x_2 - x_3 \leq -2) \quad \vee \quad (x_4 - x_5 \leq 5)) \quad \wedge$$

$$((x_2 - x_1 \leq 4) \quad \vee \quad (x_3 - x_7 \leq -6)) \quad \wedge$$

...

- Straightforward to encode into SMT( $\mathcal{DL}$ )



# Goal

Provide an overview of standard “lazy” SMT:

- foundations
- SMT-solving techniques
- beyond solving: advanced SMT functionalities
- ongoing research

We do **not** cover related approaches like:

- Eager SAT encodings
- Rewrite-based approaches

We refer to [64, 10] for an overview and references.

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 **Efficient SMT solving**
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

# Outline

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 **Efficient SMT solving**
  - **Combining SAT with Theory Solvers**
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

## Modern “lazy” SMT( $\mathcal{T}$ ) solvers

A prominent “lazy” approach [42, 2, 72, 3, 8, 33] (aka “DPLL( $\mathcal{T}$ )”)

- a **CDCL SAT solver** is used to enumerate truth assignments  $\mu_i$  for (the Boolean abstraction of) the input formula  $\varphi$
  - a theory-specific solver  **$\mathcal{T}$ -solver** checks the  $\mathcal{T}$ -satisfiability of the **set of  $\mathcal{T}$ -literals** corresponding to each assignment
- 
- Built on top of modern SAT CDCL solvers
    - benefit for free from all modern CDCL techniques (e.g., **Boolean preprocessing, backjumping & learning, restarts,...**)
    - benefit for free from all state-of-the-art data structures and implementation tricks (e.g., **two-watched literals,...**)
  - Many techniques to maximize the benefits of integration [64, 10]
  - Many lazy SMT tools available ( **Barcelogic, CVC4, MathSAT, OpenSMT, Yices, Z3, ...** )

# Basic schema: example

$\varphi =$

$$c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$$

$$c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$$

$$c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$$

*true, false*

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

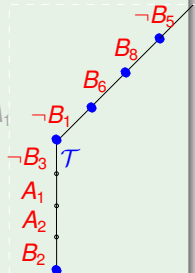
$$B_3 \vee A_2$$

$$\neg B_4 \vee \neg B_5 \vee \neg A_1$$

$$A_1 \vee B_3$$

$$B_6 \vee B_7 \vee \neg A_1$$

$$A_1 \vee B_8 \vee A_2$$



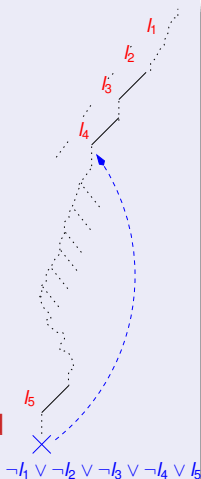
$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1, \neg B_3, A_1, A_2, B_2\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2), \neg(3v_1 - 2v_2 \leq 3), (v_1 - v_5 \leq 1)\}$$

$\implies$  unsatisfiable in  $\mathcal{LRA} \implies$  backtrack

# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning [47, 72, 3, 8, 33]

- Similar to Boolean backjumping & learning
- important property of  $\mathcal{T}$ -solver:
  - **extraction of  $\mathcal{T}$ -conflict sets**: if  $\mu$  is  $\mathcal{T}$ -unsatisfiable, then  $\mathcal{T}$ -solver( $\mu$ ) returns the subset  $\eta$  of  $\mu$  causing the  $\mathcal{T}$ -unsatisfiability of  $\mu$  ( $\mathcal{T}$ -conflict set)
- If so, the  **$\mathcal{T}$ -conflict clause**  $C := \neg\eta$  is used to drive the backjumping & learning mechanism of the SAT solver  
⇒ lots of search saved
- **the less redundant is  $\eta$ , the more search is saved**



# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning: example

$\varphi =$

$$c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$$

$$c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$$

$$c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$$

$$c_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$$

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

$$B_3 \vee A_2$$

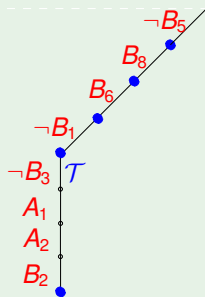
$$\neg B_4 \vee \neg B_5 \vee \neg A_1$$

$$A_1 \vee B_3$$

$$B_6 \vee B_7 \vee \neg A_1$$

$$A_1 \vee B_8 \vee A_2$$

$$B_5 \vee \neg B_8 \vee \neg B_7$$



*true, false*

$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1, \neg B_3, A_1, A_2, B_2\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2), \neg(3v_1 - 2v_2 \leq 3), (v_1 - v_5 \leq 1)\}$$

$$\eta = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_1 - v_5 \leq 1)\}$$

$$\eta^p = \{\neg B_5, B_8, B_2\}$$

# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning: example

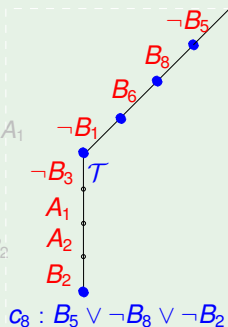
$\varphi =$

- $c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$
- $c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$
- $c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$
- $c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$
- $c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$
- $c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$
- $c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$
- $c_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$

*true, false*

$\varphi^p =$

- $\neg B_1 \vee A_1$
- $\neg A_2 \vee B_2$
- $B_3 \vee A_2$
- $\neg B_4 \vee \neg B_5 \vee \neg A_1$
- $A_1 \vee B_3$
- $B_6 \vee B_7 \vee \neg A_1$
- $A_1 \vee B_8 \vee A_2$
- $B_5 \vee \neg B_8 \vee \neg B_2$



$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1, \neg B_3, A_1, A_2, B_2\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2), \neg(3v_1 - 2v_2 \leq 3), (v_1 - v_5 \leq 1)\}$$

$$\eta = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_1 - v_5 \leq 1)\}$$

$$\eta^p = \{\neg B_5, B_8, B_2\}$$



# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning: example

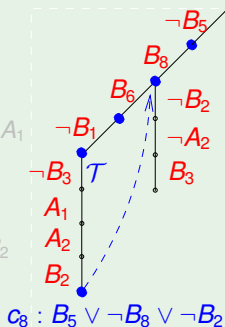
$\varphi =$

- $c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$
- $c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$
- $c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$
- $c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$
- $c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$
- $c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$
- $c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$
- $c_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$

*true, false*

$\varphi^p =$

- $\neg B_1 \vee A_1$
- $\neg A_2 \vee B_2$
- $B_3 \vee A_2$
- $\neg B_4 \vee \neg B_5 \vee \neg A_1$
- $A_1 \vee B_3$
- $B_6 \vee B_7 \vee \neg A_1$
- $A_1 \vee B_8 \vee A_2$
- $B_5 \vee \neg B_8 \vee \neg B_2$



$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1, \neg B_3, A_1, A_2, B_2\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2), \neg(3v_1 - 2v_2 \leq 3), (v_1 - v_5 \leq 1)\}$$

$$\eta = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_1 - v_5 \leq 1)\}$$

$$\eta^p = \{\neg B_5, B_8, B_2\}$$

# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning: example (2)

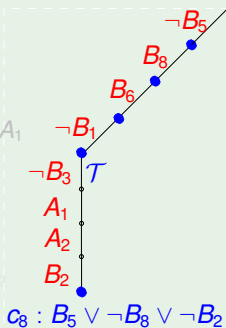
$\varphi =$

- $c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$
- $c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$
- $c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$
- $c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$
- $c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$
- $c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$
- $c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$
- $c_8' : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$
- $c_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$

*true, false*

$\varphi^p =$

- $\neg B_1 \vee A_1$
- $\neg A_2 \vee B_2$
- $B_3 \vee A_2$
- $\neg B_4 \vee \neg B_5 \vee \neg A_1$
- $A_1 \vee B_3$
- $B_6 \vee B_7 \vee \neg A_1$
- $A_1 \vee B_8 \vee A_2$
- $B_5 \vee \neg B_8 \vee B_1$
- $B_5 \vee \neg B_8 \vee \neg B_2$



$c_8$ : theory conflicting clause

$$\begin{array}{c}
 \overbrace{B_5 \vee \neg B_8 \vee \neg B_2}^{c_2} \quad \overbrace{\neg A_2 \vee B_2}^{c_2} \quad (B_2) \\
 \hline
 B_5 \vee \neg B_8 \vee \neg A_2 \\
 \hline
 B_5 \vee \neg B_8 \vee B_3 \\
 \hline
 \overbrace{B_3 \vee A_2}^{c_3} \quad (\neg A_2) \\
 \hline
 B_5 \vee \neg B_8 \vee B_1 \\
 \hline
 \overbrace{B_5 \vee B_1 \vee \neg B_3}^{c_{\mathcal{T}}} \quad (B_3)
 \end{array}$$

$c_8'$ : mixed Boolean+theory conflict clause

# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning: example (2)

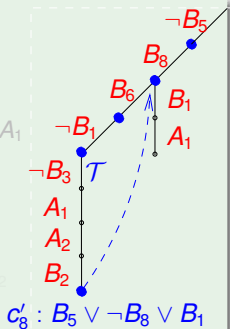
$\varphi =$

- $c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$
- $c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$
- $c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$
- $c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$
- $c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$
- $c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$
- $c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$
- $c'_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$
- $c_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$

*true, false*

$\varphi^p =$

- $\neg B_1 \vee A_1$
- $\neg A_2 \vee B_2$
- $B_3 \vee A_2$
- $\neg B_4 \vee \neg B_5 \vee \neg A_1$
- $A_1 \vee B_3$
- $B_6 \vee B_7 \vee \neg A_1$
- $A_1 \vee B_8 \vee A_2$
- $B_5 \vee \neg B_8 \vee B_1$
- $B_5 \vee \neg B_8 \vee \neg B_2$



$c_8$ : theory conflicting clause

$$\begin{array}{c}
 \overbrace{B_5 \vee \neg B_8 \vee \neg B_2}^{c_1} \quad \overbrace{\neg A_2 \vee B_2}^{c_2} \\
 \hline
 B_5 \vee \neg B_8 \vee \neg A_2 \quad (B_2) \\
 \hline
 B_5 \vee \neg B_8 \vee B_3 \\
 \hline
 \overbrace{B_3 \vee A_2}^{c_3} \quad (\neg A_2) \\
 \hline
 B_5 \vee \neg B_8 \vee B_1 \\
 \hline
 \overbrace{B_5 \vee B_1 \vee \neg B_3}^{c_T} \quad (B_3)
 \end{array}$$

$c'_8$ : mixed Boolean+theory conflict clause

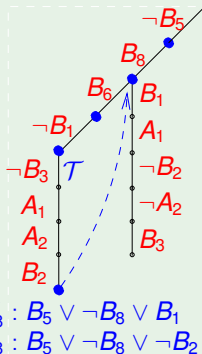
# $\mathcal{T}$ -Backjumping & $\mathcal{T}$ -learning: example (2)

$\varphi =$

- $c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$
  - $c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$
  - $c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$
  - $c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$
  - $c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$
  - $c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$
  - $c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$
  - $c'_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$
  - $c_8 : (3v_1 - v_3 \leq 6) \vee \neg(v_3 = 3v_5 + 4) \vee \dots$
- true, false*

$\varphi^p =$

- $\neg B_1 \vee A_1$
- $\neg A_2 \vee B_2$
- $B_3 \vee A_2$
- $\neg B_4 \vee \neg B_5 \vee \neg A_1$
- $A_1 \vee B_3$
- $B_6 \vee B_7 \vee \neg A_1$
- $A_1 \vee B_8 \vee A_2$
- $B_5 \vee \neg B_8 \vee B_1$
- $B_5 \vee \neg B_8 \vee \neg B_2$



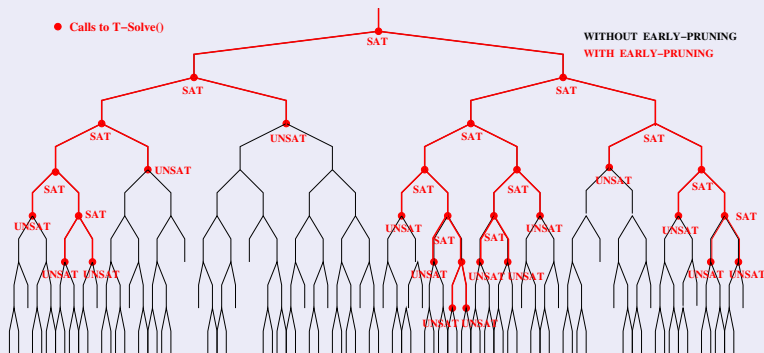
$c_8$ : theory conflicting clause

$$\begin{array}{c}
 \overbrace{B_5 \vee \neg B_8 \vee \neg B_2}^{c_2} \quad \overbrace{\neg A_2 \vee B_2}^{c_2} \quad (B_2) \\
 \hline
 B_5 \vee \neg B_8 \vee \neg A_2 \\
 \hline
 B_5 \vee \neg B_8 \vee B_3 \\
 \hline
 \overbrace{B_3 \vee A_2}^{c_3} \quad (\neg A_2) \\
 \hline
 B_5 \vee \neg B_8 \vee B_1 \\
 \hline
 \overbrace{B_5 \vee B_1 \vee \neg B_3}^{c_{\mathcal{T}}} \quad (B_3)
 \end{array}$$

$c'_8$ : mixed Boolean+theory conflict clause

# Early Pruning [42, 2, 72]

- Introduce a  $\mathcal{T}$ -satisfiability test on **intermediate assignments**: if  $\mathcal{T}$ -solver returns UNSAT, the procedure backtracks.
  - benefit: prunes drastically the Boolean search
  - Drawback: possibly **many useless calls to  $\mathcal{T}$ -solver**



## Early Pruning [42, 2, 72] (cont.)

- Different strategies for interleaving Boolean search steps and  $\mathcal{T}$ -solver calls
  - **Eager E.P.** [72, 11, 70, 41]): invoke  $\mathcal{T}$ -solver every time a new  $\mathcal{T}$ -atom is added to the assignment (unit propagations included)
  - **Selective E.P.**: Do not call  $\mathcal{T}$ -solver if the have been added only literals which hardly cause any  $\mathcal{T}$ -conflict with the previous assignment (e.g., Boolean literals, disequalities  $(x - y \neq 3)$ ,  $\mathcal{T}$ -literals introducing new variables  $(x - z = 3)$  )
  - **Weakened E.P.**: for intermediate checks only, use **weaker** but faster versions of  $\mathcal{T}$ -solver (e.g., check  $\mu$  on  $\mathbb{R}$  rather than on  $\mathbb{Z}$ ):  
 $\{(x - y \leq 4), (z - x \leq -6), (z = y), (3x + 2y - 3z = 4)\}$

## Early pruning: example

$$\begin{aligned}\varphi = & \{ \neg(2v_2 - v_3 > 2) \vee A_1 \} \wedge \\ & \{ \neg A_2 \vee (2v_1 - 4v_5 > 3) \} \wedge \\ & \{ (3v_1 - 2v_2 \leq 3) \vee A_2 \} \wedge \\ & \{ \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1 \} \wedge \\ & \{ A_1 \vee (3v_1 - 2v_2 \leq 3) \} \wedge \\ & \{ (v_1 - v_5 \leq 1) \vee (v_5 = 5 - 3v_4) \vee \neg A_1 \} \wedge \\ & \{ A_1 \vee (v_3 = 3v_5 + 4) \vee A_2 \}.\end{aligned}$$

$$\begin{aligned}\varphi^p = & \{ \neg B_1 \vee A_1 \} \wedge \\ & \{ \neg A_2 \vee B_2 \} \wedge \\ & \{ B_3 \vee A_2 \} \wedge \\ & \{ \neg B_4 \vee \neg B_5 \vee \neg A_1 \} \wedge \\ & \{ A_1 \vee B_3 \} \wedge \\ & \{ B_6 \vee B_7 \vee \neg A_1 \} \wedge \\ & \{ A_1 \vee B_8 \vee A_2 \}.\end{aligned}$$

- Suppose it is built the intermediate assignment:

$$\mu^p = \neg B_1 \wedge \neg A_2 \wedge B_3 \wedge \neg B_5.$$

corresponding to the following set of  $\mathcal{T}$ -literals

$$\mu' = \neg(2v_2 - v_3 > 2) \wedge \neg A_2 \wedge (3v_1 - 2v_2 \leq 3) \wedge \neg(3v_1 - v_3 \leq 6).$$

- If  $\mathcal{T}$ -solver is invoked on  $\mu'$ , then it returns UNSAT, and DPLL backtracks **without exploring any extension of  $\mu'$** .

# Early pruning: remark

## Incrementality & Backtrackability of $\mathcal{T}$ -solvers

- With early pruning, lots of **incremental calls to  $\mathcal{T}$ -solver**:

$\mathcal{T}\text{-solver}(\mu_1)$	$\Rightarrow \text{Sat}$	Undo $\mu_4, \mu_3, \mu_2$	
$\mathcal{T}\text{-solver}(\mu_1 \cup \mu_2)$	$\Rightarrow \text{Sat}$	$\mathcal{T}\text{-solver}(\mu_1 \cup \mu'_2)$	$\Rightarrow \text{Sat}$
$\mathcal{T}\text{-solver}(\mu_1 \cup \mu_2 \cup \mu_3)$	$\Rightarrow \text{Sat}$	$\mathcal{T}\text{-solver}(\mu_1 \cup \mu'_2 \cup \mu'_3)$	$\Rightarrow \text{Sat}$
$\mathcal{T}\text{-solver}(\mu_1 \cup \mu_2 \cup \mu_3 \cup \mu_4)$	$\Rightarrow \text{Unsat}$	...	

$\Rightarrow$  Desirable features of  $\mathcal{T}$ -solvers:

- **incrementality**:  $\mathcal{T}\text{-solver}(\mu_1 \cup \mu_2)$  reuses computation of  $\mathcal{T}\text{-solver}(\mu_1)$  without restarting from scratch
- **backtrackability (resettability)**:  $\mathcal{T}\text{-solver}$  can efficiently undo steps and return to a previous status on the stack

$\Rightarrow$   $\mathcal{T}$ -solver requires a **stack-based interface**



## $\mathcal{T}$ -Propagation [2, 3, 41]

- strictly related to early pruning
- important property of  $\mathcal{T}$ -solver:
  - **$\mathcal{T}$ -deduction**: when a partial assignment  $\mu$  is  $\mathcal{T}$ -satisfiable,  $\mathcal{T}$ -solver may be able to return also an assignment  $\eta$  to some unassigned atom occurring in  $\varphi$  s.t.  $\mu \models_{\mathcal{T}} \eta$ .
- If so:
  - the literal  $\eta$  is then unit-propagated;
  - optionally, a  **$\mathcal{T}$ -deduction clause**  $C := \neg\mu' \vee \eta$  can be learned,  $\mu'$  being the subset of  $\mu$  which caused the deduction ( $\mu' \models_{\mathcal{T}} \eta$ )
  - **lazy explanation**: compute  $C$  only if needed for conflict analysis

$\implies$  may prune drastically the search

Both  $\mathcal{T}$ -deduction clauses and  $\mathcal{T}$ -conflict clauses are called  **$\mathcal{T}$ -lemmas** since they are valid in  $\mathcal{T}$

# T-propagation: example

$\varphi =$

$$c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$$

$$c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$$

$$c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$$

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

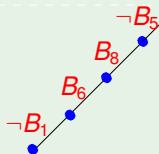
$$B_3 \vee A_2$$

$$\neg B_4 \vee \neg B_5 \vee \neg A_1$$

$$A_1 \vee B_3$$

$$B_6 \vee B_7 \vee \neg A_1$$

$$A_1 \vee B_8 \vee A_2$$



*true, false*

$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2)\}$$

$$\models_{\mathcal{LRA}} \underbrace{\neg(3v_1 - 2v_2 \leq 3)}_{\neg B_3}$$

$\Rightarrow$  propagate  $\neg B_3$  [and learn the deduction clause  $B_5 \vee B_1 \vee \neg B_3$ ]

# T-propagation: example

$\varphi =$

$$c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$$

$$c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$$

$$c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$$

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

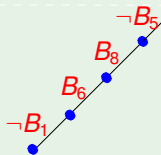
$$B_3 \vee A_2$$

$$\neg B_4 \vee \neg B_5 \vee \neg A_1$$

$$A_1 \vee B_3$$

$$B_6 \vee B_7 \vee \neg A_1$$

$$A_1 \vee B_8 \vee A_2$$



*true, false*

$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2)\}$$

$$\models_{\mathcal{LRA}} \underbrace{\neg(3v_1 - 2v_2 \leq 3)}_{\neg B_3}$$

$\Rightarrow$  propagate  $\neg B_3$  [and learn the deduction clause  $B_5 \vee B_1 \vee \neg B_3$ ]

# $\mathcal{T}$ -propagation: example

$\varphi =$

$$c_1 : \neg(2v_2 - v_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (v_1 - v_5 \leq 1)$$

$$c_3 : (3v_1 - 2v_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3v_1 - 2v_2 \leq 3)$$

$$c_6 : (v_2 - v_4 \leq 6) \vee (v_5 = 5 - 3v_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (v_3 = 3v_5 + 4) \vee A_2$$

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

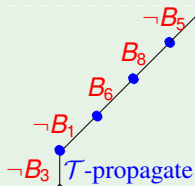
$$B_3 \vee A_2$$

$$\neg B_4 \vee \neg B_5 \vee \neg A_1$$

$$A_1 \vee B_3$$

$$B_6 \vee B_7 \vee \neg A_1$$

$$A_1 \vee B_8 \vee A_2$$



*true, false*

$$\mu^p = \{\neg B_5, B_8, B_6, \neg B_1\}$$

$$\mu = \{\neg(3v_1 - v_3 \leq 6), (v_3 = 3v_5 + 4), (v_2 - v_4 \leq 6), \neg(2v_2 - v_3 > 2)\}$$

$$\models_{\mathcal{LRA}} \underbrace{\neg(3v_1 - 2v_2 \leq 3)}_{\neg B_3}$$

$\implies$  propagate  $\neg B_3$  [and learn the deduction clause  $B_5 \vee B_1 \vee \neg B_3$ ]

## Pure-literal filtering [72, 3, 16]

### Property

If we have non-Boolean  $\mathcal{T}$ -atoms occurring only positively [negatively] **in the original formula  $\varphi$**  (learned clauses are not considered), we can drop every negative [positive] occurrence of them from the assignment to be checked by  $\mathcal{T}$ -solver (and from the  $\mathcal{T}$ -deducible ones).

- increases the chances of finding a model
- reduces the effort for the  $\mathcal{T}$ -solver
- eliminates unnecessary “nasty” negated literals (e.g. negative equalities like  $\neg(3v_1 - 9v_2 = 3)$  in  $\mathcal{LIA}$  force splitting:  $(3v_1 - 9v_2 > 3) \vee (3v_1 - 9v_2 < 3)$ ).
- may weaken the effect of early pruning.

## Pure literal filtering: example

$$\begin{aligned}\varphi = & \{ \neg(2v_2 - v_3 > 2) \vee A_1 \} \wedge \\ & \{ \neg A_2 \vee (2v_1 - 4v_5 > 3) \} \wedge \\ & \{ (3v_1 - 2v_2 \leq 3) \vee A_2 \} \wedge \\ & \{ \neg(2v_3 + v_4 \geq 5) \vee \neg(3v_1 - v_3 \leq -2) \vee \neg A_1 \} \wedge \\ & \{ A_1 \vee (3v_1 - 2v_2 \leq 3) \} \wedge \\ & \{ (v_1 - v_5 \leq 1) \vee (v_5 = 5 - 3v_4) \vee \neg A_1 \} \wedge \\ & \{ A_1 \vee (v_3 = 3v_5 + 4) \vee A_2 \} \wedge \\ & \{ (2v_2 - v_3 > 2) \vee \neg(3v_1 - 2v_2 \leq 3) \vee (3v_1 - v_3 \leq -2) \} \text{ learned}\end{aligned}$$

$$\mu' = \{ \neg(2v_2 - v_3 > 2), \neg A_2, (3v_1 - 2v_2 \leq 3), \neg A_1, (v_3 = 3v_5 + 4), (3v_1 - v_3 \leq -2) \}.$$

$\implies$  Sat:  $v_1 = v_2 = v_3 = 0, v_5 = -4/3$  is a solution

### Note

- $(3v_1 - v_3 \leq -2)$  “filtered out” from  $\mu'$  because it occurs only negatively in the original formula  $\varphi$
- $\mu' \cup \{(3v_1 - v_3 \leq -2)\}$  is  $\mathcal{LRA}$ -unsatisfiable

## Preprocessing atoms [42, 47, 4]

### Source of inefficiency:

Semantically equivalent but syntactically different atoms are not recognized to be identical [resp. one the negation of the other]

⇒ they may be assigned different [resp. identical] truth values.

⇒ lots of redundant unsatisfiable assignment generated

### Solution

Rewrite a priori trivially-equivalent atoms/literals into the same atom/literal.

## Preprocessing atoms (cont.)

- **Sorting:**  $(v_1 + v_2 \leq v_3 + 1), (v_2 + v_1 \leq v_3 + 1), (v_1 + v_2 - 1 \leq v_3) \implies (v_1 + v_2 - v_3 \leq 1)$ ;
- **Rewriting dual operators:**  
 $(v_1 < v_2), (v_1 \geq v_2) \implies (v_1 < v_2), \neg(v_1 < v_2)$
- **Exploiting associativity:**  
 $(v_1 + (v_2 + v_3) = 1), ((v_1 + v_2) + v_3) = 1 \implies (v_1 + v_2 + v_3 = 1)$ ;
- **Factoring**  $(v_1 + 2.0v_2 \leq 4.0), (-2.0v_1 - 4.0v_2 \geq -8.0), \implies (0.25v_1 + 0.5v_2 \leq 1.0)$ ;
- **Exploiting properties of  $\mathcal{T}$ :**  
 $(v_1 \leq 3), (v_1 < 4) \implies (v_1 \leq 3)$  if  $v_1 \in \mathbb{Z}$ ;
- ...



## Static Learning [2, 4]

- Often possible to quickly detect a priori short and “obviously unsatisfiable” pairs or triplets of literals occurring in  $\varphi$ .
    - mutual exclusion  $\{x = 0, x = 1\}$ ,
    - congruence  $\{(x_1 = y_1), (x_2 = y_2), \neg(f(x_1, x_2) = f(y_1, y_2))\}$ ,
    - transitivity  $\{(x - y = 2), (y - z \leq 4), \neg(x - z \leq 7)\}$ ,
    - substitution  $\{(x = y), (2x - 3z \leq 3), \neg(2y - 3z \leq 3)\}$
    - ...
  - Preprocessing step: detect these literals and add blocking clauses to the input formula:  
(e.g.,  $\neg(x = 0) \vee \neg(x = 1)$ )
- ⇒ **No assignment including one such group of literals is ever generated**: as soon as all but one literals are assigned, the remaining one is immediately assigned false by unit-propagation.

## Other optimization techniques

- $\mathcal{T}$ -deduced-literal filtering
- Ghost-literal filtering
- $\mathcal{T}$ -solver layering
- $\mathcal{T}$ -solver clustering
- ...

(see [64, 10] for an overview)

## Other SAT-solving techniques for SMT?

Frequently-asked question:

Are CDCL SAT solvers the only suitable Boolean Engines for SMT?

Some previous attempts:

- Ordered Binary Decision Diagrams (OBDDs) [73, 55, 1]
- Stochastic Local Search [46]

CDCL based currently much more efficient.

# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- the SAT solver:
  - “sees” only  $\varphi^p$
  - finds  $\mu^p$ s.t.  $\mu^p \models \varphi^p$
  - cannot state if  $\mu^p \models \tau^p$
  - invokes  $\mathcal{T}$ -solver on  $\mu^p$
- the  $\mathcal{T}$ -solver:
  - “sees”  $\tau^p$
  - checks if  $\mu^p \models \tau^p$ :

# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

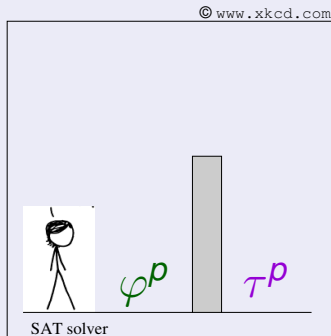
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- **the SAT solver:**

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- **the  $\mathcal{T}$ -solver:**

- “sees”  $\tau^p$
- checks if  $\mu^p \models \tau^p$ :



# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

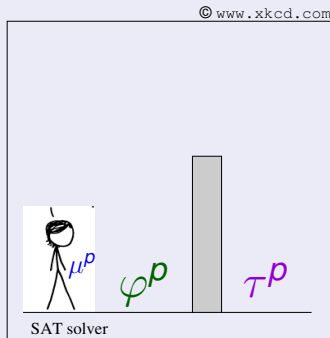
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- **the SAT solver:**

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- the  $\mathcal{T}$ -solver:

- “sees”  $\tau^p$
- checks if  $\mu^p \models \tau^p$ :



# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

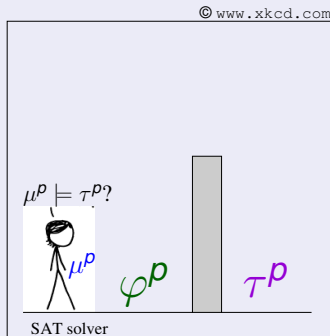
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- **the SAT solver:**

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- **cannot state if  $\mu^p \models \tau^p$**
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- **the  $\mathcal{T}$ -solver:**

- “sees”  $\tau^p$
- checks if  $\mu^p \models \tau^p$ :



# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

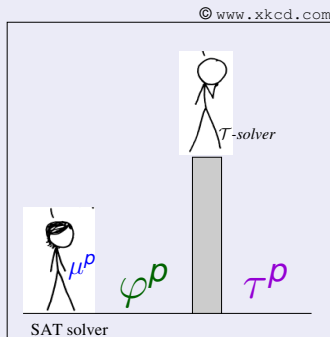
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- the SAT solver:

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- the  $\mathcal{T}$ -solver:

- “sees”  $\tau^p$
- checks if  $\mu^p \models \tau^p$ :
  - if yes, returns SAT
  - if no, returns UNSAT and some falsified clauses  $c_1^p, \dots, c_k^p \in \tau^p$





# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

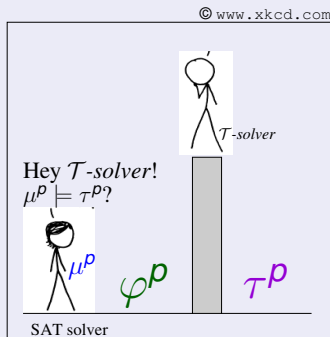
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- **the SAT solver:**

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- **invokes  $\mathcal{T}$ -solver** on  $\mu^p$

- **the  $\mathcal{T}$ -solver:**

- “sees”  $\tau^p$
- checks if  $\mu^p \models \tau^p$ :
  - if yes, returns SAT
  - if no, returns UNSAT and some falsified clauses  $c_1^p, \dots, c_k^p \in \tau^p$



# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

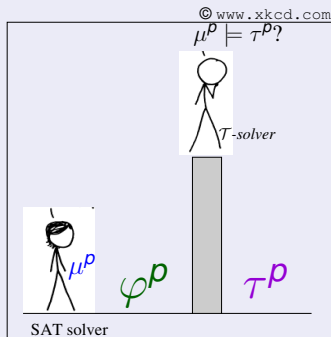
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- the SAT solver:

- “sees” only  $\varphi^p$
- finds  $\mu^p$ s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- the  $\mathcal{T}$ -solver:

- “sees”  $\tau^p$
  - checks if  $\mu^p \models \tau^p$ :
    - if yes, returns SAT
    - if no, returns UNSAT and some falsified clauses
- $c_1^p, \dots, c_k^p \in \tau^p$



# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

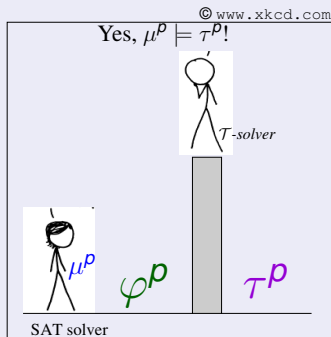
$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- the SAT solver:

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- the  $\mathcal{T}$ -solver:

- “sees”  $\tau^p$
  - checks if  $\mu^p \models \tau^p$ :
    - if yes, returns SAT
    - if no, returns UNSAT and some falsified clauses
- $c_1^p, \dots, c_k^p \in \tau^p$



# SMT formulas = “partially-invisible” SAT formulas

An SMT problem  $\varphi$  from the perspective of a SAT solver:

- a “partially-invisible” Boolean CNF formula  $\varphi^p \wedge \tau^p$ :
  - $\varphi^p$ : the Boolean abstraction of the input formula  $\varphi$
  - $\tau^p$ : (the B. abst. of) the set  $\tau$  of all  $\mathcal{T}$ -lemmas on atoms in  $\varphi$ .

$\varphi$   $\mathcal{T}$ -satisfiable iff  $\varphi^p \wedge \tau^p$  satisfiable.

- the SAT solver:

- “sees” only  $\varphi^p$
- finds  $\mu^p$  s.t.  $\mu^p \models \varphi^p$
- cannot state if  $\mu^p \models \tau^p$
- invokes  $\mathcal{T}$ -solver on  $\mu^p$

- the  $\mathcal{T}$ -solver:

- “sees”  $\tau^p$
- checks if  $\mu^p \models \tau^p$ :
  - if yes, returns SAT
  - if no, returns UNSAT and some falsified clauses  $c_1^p, \dots, c_k^p \in \tau^p$



# Example

$\varphi$  :

$$c_1 : \{A_1\}$$

$$c_2 : \{\neg A_1 \vee (x - z > 4)\}$$

$$c_3 : \{\neg A_3 \vee A_1 \vee (y \geq 1)\}$$

$$c_4 : \{\neg A_2 \vee \neg(x - z > 4) \vee \neg A_1\}$$

$$c_5 : \{(x - y \leq 3) \vee \neg A_4 \vee A_5\}$$

$$c_6 : \{\neg(y - z \leq 1) \vee (x + y = 1) \vee \neg A_5\}$$

$$c_7 : \{A_3 \vee \neg(x + y = 0) \vee A_2\}$$

$$c_8 : \{\neg A_3 \vee (z + y = 2)\}$$

$\tau$  : (all possible  $\mathcal{T}$ -lemmas on the  $\mathcal{T}$ -atoms of  $\varphi$ )

$$c_9 : \{\neg(x + y = 0) \vee \neg(x + y = 1)\}$$

$$c_{10} : \{\neg(x - z > 4) \vee \neg(x - y \leq 3) \vee \neg(y - z \leq 1)\}$$

$$c_{11} : \{(x - z > 4) \vee (x - y \leq 3) \vee (y - z \leq 1)\}$$

$$c_{12} : \{\neg(x - z > 4) \vee \neg(x + y = 1) \vee \neg(z + y = 2)\}$$

$$c_{13} : \{\neg(x - z > 4) \vee \neg(x + y = 0) \vee \neg(z + y = 2)\}$$

... ..

$$\mu_1^p : \{A_1, B_1, \neg A_2, A_3, \neg A_4, \neg A_5, \neg B_6, B_5, B_3, B_4, B_7, \neg B_2\}$$

$$\mu_1 : \{\underline{(x - z > 4)}, \neg(x + y = 0), \underline{(x + y = 1)}, \underline{(x - y \leq 3)}, \underline{(y - z \leq 1)}, \underline{(z + y = 2)}, \neg(y \geq 1)\}$$

satisfies  $\varphi^p$ , but violates both  $c_{10}$  and  $c_{12}$  in  $\tau^p$ .

$\varphi^p$  :

$$c_1 : \{A_1\}$$

$$c_2 : \{\neg A_1 \vee B_1\}$$

$$c_3 : \{\neg A_3 \vee A_1 \vee B_2\}$$

$$c_4 : \{\neg A_2 \vee \neg B_1 \vee \neg A_1\}$$

$$c_5 : \{B_3 \vee \neg A_4 \vee A_5\}$$

$$c_6 : \{\neg B_4 \vee B_5 \vee \neg A_5\}$$

$$c_7 : \{A_3 \vee \neg B_6 \vee A_2\}$$

$$c_8 : \{\neg A_3 \vee B_7\}$$

$\tau^p$  :

$$c_9 : \{\neg B_6 \vee \neg B_5\}$$

$$c_{10} : \{\neg B_1 \vee \neg B_3 \vee \neg B_4\}$$

$$c_{11} : \{B_1 \vee B_3 \vee B_4\}$$

$$c_{12} : \{\neg B_1 \vee \neg B_5 \vee \neg B_7\}$$

$$c_{13} : \{\neg B_1 \vee \neg B_6 \vee \neg B_7\}$$

... ..

# Exercise

Consider the following formula in the theory  $\mathcal{EUF}$ .

$$\begin{aligned}\varphi = & \{(f(x) = f(f(y))) \vee A_2\} \wedge \\ & \{\neg(h(x, f(y)) = h(g(x), y)) \vee \underline{\neg(h(x, g(z) = h(f(x), y)))} \vee \neg A_1\} \wedge \\ & \{A_1 \vee (h(x, y) = h(y, x))\} \wedge \\ & \{\underline{(x = f(x))} \vee A_3 \vee \neg A_1\} \wedge \\ & \{\underline{\neg(w(x) = g(f(y)))} \vee A_1\} \wedge \\ & \{\underline{\neg A_2} \vee (w(g(x)) = w(f(x)))\} \wedge \\ & \{A_1 \vee \underline{(y = g(z))} \vee A_2\}\end{aligned}$$

and consider the partial truth assignment  $\mu$  given by the underlined literals above:

$$\{\underline{\neg(w(x) = g(f(y)))}, \neg A_2, \neg(h(x, g(z) = h(f(x), y))}, (x = f(x)), (y = g(z))\}.$$

- 1 Does (the Boolean abstraction of)  $\mu$  propositionally satisfy (the Boolean abstraction of)  $\varphi$ ?
- 2 Is  $\mu$  satisfiable in  $\mathcal{EUF}$ ?
  - 1 If no, find a minimal conflict set for  $\mu$  and the corresponding conflict clause  $C$ .
  - 2 If yes, show one unassigned literal which can be deduced from  $\mu$ , and show the corresponding deduction clause  $C$ .

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 **Efficient SMT solving**
  - Combining SAT with Theory Solvers
  - **Theory Solvers for Theories of Interest (hints)**
  - SMT for Combinations of Theories
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

## Summary: desirable properties for $\mathcal{T}$ -solver

- Correctness & Completeness: be correct & complete
- Time efficiency: be fast
- Incrementality & backtrackability:  $\mathcal{T}$ -solver( $\mu_1 \cup \mu_2$ ) reuses computation of  $\mathcal{T}$ -solver( $\mu_1$ )
- Diagnosis capabilities:  $\mathcal{T}$ -solver able to produce conflict sets
- Deduction capabilities:  $\mathcal{T}$ -solver able to deduce assignments



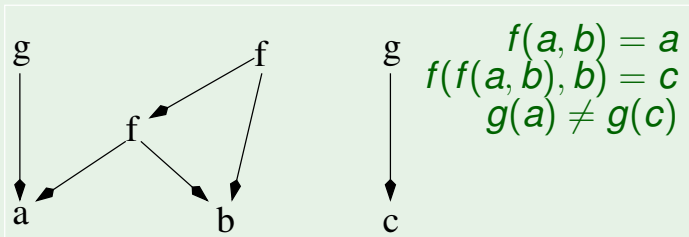
# $\mathcal{T}$ -solvers for Equality and Uninterpreted Functions ( $\mathcal{EUF}$ )

- Typically used as a “core”  $\mathcal{T}$ -solver
- $\mathcal{EUF}$  polynomial:  $O(n \cdot \log(n))$
- Fully incremental and backtrackable (stack-based)
- use a congruence closure data structures (**E-Graphs**) [36, 59, 32], based on the Union-Find data-structure for equivalence classes
- Supports efficient  $\mathcal{T}$ -propagation
  - Exhaustive for positive equalities
  - Incomplete for disequalities
- Supports Lazy explanations and conflict generation
  - However, minimality not guaranteed
- Supports efficient extensions (e.g., Integer offsets, Bit-vector slicing and concatenation)

## T-solvers for $\mathcal{EUF}$ : Example

Idea (sketch): given the set of terms occurring in the formula represented as nodes in a DAG (aka **term bank**),

- if  $(t = s)$ , then merge the eq. classes of  $t$  and  $s$ 
  - e.g. use the **union-find** data structure
- if  $\forall i \in 1 \dots k, t_i$  and  $s_i$  pairwise belong to the same eq. classes, then merge the eq. classes of  $f(t_1, \dots, t_k)$  and  $f(s_1, \dots, s_k)$
- if  $(t \neq s)$  and  $t$  and  $s$  belong to the same eq. class, then conflict

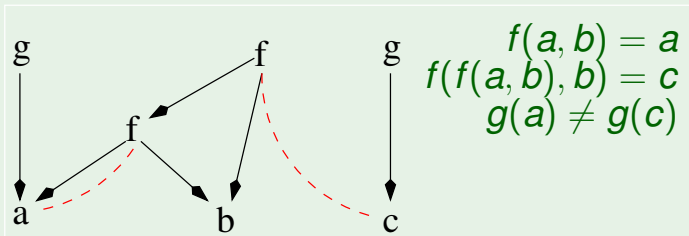


Example borrowed from [36].

## T-solvers for $\mathcal{EUF}$ : Example

Idea (sketch): given the set of terms occurring in the formula represented as nodes in a DAG (aka **term bank**),

- if  $(t = s)$ , then merge the eq. classes of  $t$  and  $s$ 
  - e.g. use the **union-find** data structure
- if  $\forall i \in 1 \dots k, t_i$  and  $s_i$  pairwise belong to the same eq. classes, then merge the eq. classes of  $f(t_1, \dots, t_k)$  and  $f(s_1, \dots, s_k)$
- if  $(t \neq s)$  and  $t$  and  $s$  belong to the same eq. class, then conflict

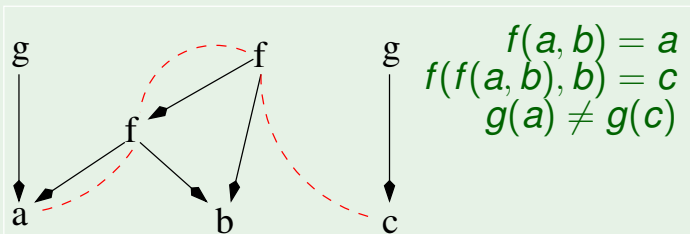


Example borrowed from [36].

## T-solvers for $\mathcal{EUF}$ : Example

Idea (sketch): given the set of terms occurring in the formula represented as nodes in a DAG (aka **term bank**),

- if  $(t = s)$ , then merge the eq. classes of  $t$  and  $s$ 
  - e.g. use the **union-find** data structure
- if  $\forall i \in 1 \dots k, t_i$  and  $s_i$  pairwise belong to the same eq. classes, then merge the eq. classes of  $f(t_1, \dots, t_k)$  and  $f(s_1, \dots, s_k)$
- if  $(t \neq s)$  and  $t$  and  $s$  belong to the same eq. class, then conflict

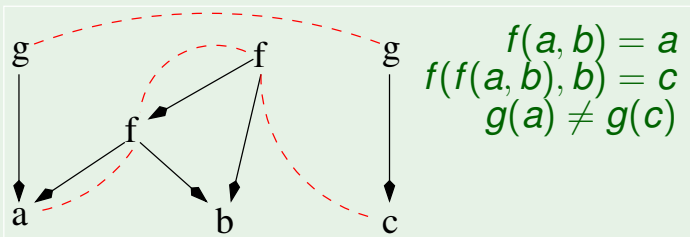


Example borrowed from [36].

## T-solvers for $\mathcal{EUF}$ : Example

Idea (sketch): given the set of terms occurring in the formula represented as nodes in a DAG (aka **term bank**),

- if  $(t = s)$ , then merge the eq. classes of  $t$  and  $s$ 
  - e.g. use the **union-find** data structure
- if  $\forall i \in 1 \dots k, t_i$  and  $s_i$  pairwise belong to the same eq. classes, then merge the eq. classes of  $f(t_1, \dots, t_k)$  and  $f(s_1, \dots, s_k)$
- if  $(t \neq s)$  and  $t$  and  $s$  belong to the same eq. class, then conflict



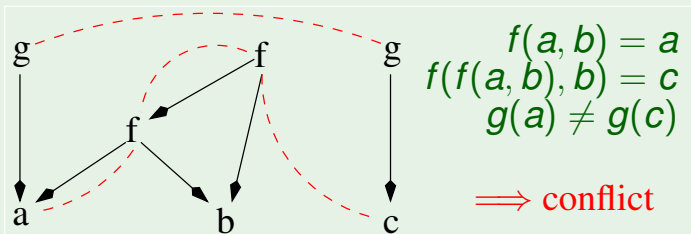
$$\begin{aligned} f(a, b) &= a \\ f(f(a, b), b) &= c \\ g(a) &\neq g(c) \end{aligned}$$

Example borrowed from [36].

## T-solvers for $\mathcal{EUF}$ : Example

Idea (sketch): given the set of terms occurring in the formula represented as nodes in a DAG (aka **term bank**),

- if  $(t = s)$ , then merge the eq. classes of  $t$  and  $s$ 
  - e.g. use the **union-find** data structure
- if  $\forall i \in 1 \dots k, t_i$  and  $s_i$  pairwise belong to the same eq. classes, then merge the eq. classes of  $f(t_1, \dots, t_k)$  and  $f(s_1, \dots, s_k)$
- if  $(t \neq s)$  and  $t$  and  $s$  belong to the same eq. class, then conflict

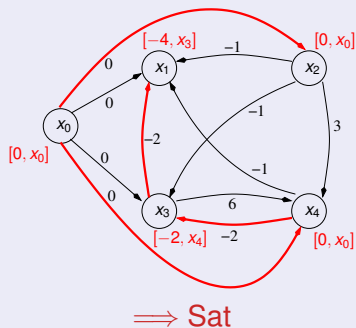


Example borrowed from [36].

# $\mathcal{T}$ -solvers for Difference logic ( $\mathcal{DL}$ )

- $\mathcal{DL}$  polynomial:  $O(\#vars \cdot \#constraints)$
- variants of the Bellman-Ford shortest-path algorithm: a negative cycle reveals a conflict [60, 31]
- Ex:

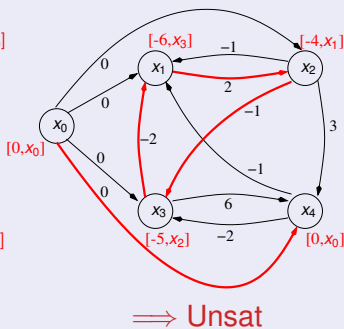
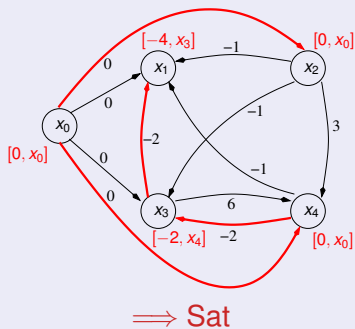
$$\{(x_1 - x_2 \leq -1), (x_1 - x_4 \leq -1), (x_1 - x_3 \leq -2), (x_2 - x_1 \leq 2), (x_3 - x_4 \leq -2), (x_3 - x_2 \leq -1), (x_4 - x_2 \leq 3), (x_4 - x_3 \leq 6)\}$$



# T-solvers for Difference logic ( $\mathcal{DL}$ )

- $\mathcal{DL}$  polynomial:  $O(\#vars \cdot \#constraints)$
- variants of the Bellman-Ford shortest-path algorithm: a negative cycle reveals a conflict [60, 31]
- Ex:

$$\{(x_1 - x_2 \leq -1), (x_1 - x_4 \leq -1), (x_1 - x_3 \leq -2), (x_2 - x_1 \leq 2), (x_3 - x_4 \leq -2), (x_3 - x_2 \leq -1), (x_4 - x_2 \leq 3), (x_4 - x_3 \leq 6)\}$$





# $\mathcal{T}$ -solvers for Linear arithmetic over the rationals ( $\mathcal{LRA}$ )

- EX:  $\{(s_1 - s_2 \leq 5.2), (s_1 = s_0 + 3.4 \cdot t - 3.4 \cdot t_0), \neg(s_1 = s_0)\}$
- $\mathcal{LRA}$  polynomial
- variants of the simplex LP algorithm [38]
- [38] allows for detecting conflict sets & performing  $\mathcal{T}$ -propagation
- strict inequalities  $t < 0$  rewritten as  $t + \epsilon \leq 0$ ,  $\epsilon$  treated symbolically

$$\begin{array}{c} \mathcal{B} \\ \left[ \begin{array}{c} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_N \end{array} \right] \end{array} = \begin{array}{c} \left[ \begin{array}{ccc} \dots & A_{1j} & \dots \\ & \vdots & \\ A_{i1} & \dots & A_{ij} & \dots & A_{iM} \\ & \vdots & & & \\ \dots & A_{Nj} & \dots \end{array} \right] \end{array} \begin{array}{c} \mathcal{N} \\ \left[ \begin{array}{c} x_{N+1} \\ \vdots \\ x_j \\ \vdots \\ x_{N+M} \end{array} \right] \end{array} ;$$

Invariant:  $\beta(x_j) \in [l_j, u_j] \forall x_j \in \mathcal{N}$

## Remark: infinite precision arithmetic

In order to avoid incorrect results due to numerical errors and to overflows, all  $\mathcal{T}$ -solvers for  $\mathcal{LRA}$ ,  $\mathcal{LIA}$  and their subtheories which are based on numerical algorithms must be implemented on top of infinite-precision-arithmetic software packages.

# $\mathcal{T}$ -solvers for Linear arithmetic over the integers ( $\mathcal{LIA}$ )

- EX:  $\{(x := x_l + 2^{16}x_h), (x \geq 0), (x \leq 2^{16} - 1)\}$
- $\mathcal{LIA}$  NP-complete
- combination of many techniques: simplex, branch&bound, cutting planes, ... [38, 44]

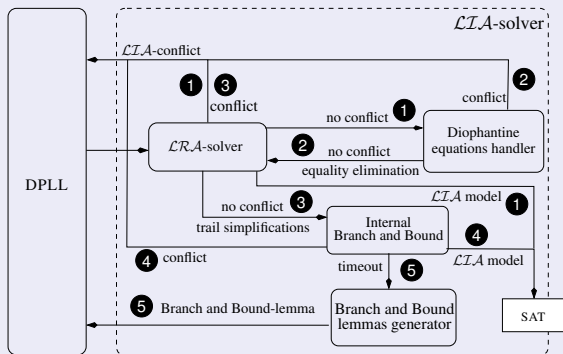


Figure courtesy of A. Griggio [44]

## $\mathcal{T}$ -solvers for Arrays ( $\mathcal{AR}$ )

- EX:  $(write(A, i, v) = write(B, i, w)) \wedge \neg(v = w)$
- NP-complete
- congruence closure ( $\mathcal{EUF}$ ) plus on-the-fly instantiation of array's axioms:

$$\forall a. \forall i. \forall e. (read(write(a, i, e), i) = e), \quad (1)$$

$$\forall a. \forall i. \forall j. \forall e. ((i \neq j) \rightarrow read(write(a, i, e), j) = read(a, j)) \quad (2)$$

$$\forall a. \forall b. (\forall i. (read(a, i) = read(b, i)) \rightarrow (a = b)). \quad (3)$$

- EX:

*Input* :  $(write(A, i, v) = write(B, i, w)) \wedge \neg(v = w)$

*inst. (1)* :  $(read(write(A, i, v), i) = v)$

$(read(write(B, i, w), i) = w)$

$\models_{\mathcal{EUF}}$   $(v = w)$

$\models_{Bool}$   $\perp$

- many strategies discussed in the literature (e.g., [36, 43, 19, 35])

# $\mathcal{T}$ -solvers for Bit vectors ( $\mathcal{BV}$ )

## Bit vectors ( $\mathcal{BV}$ )

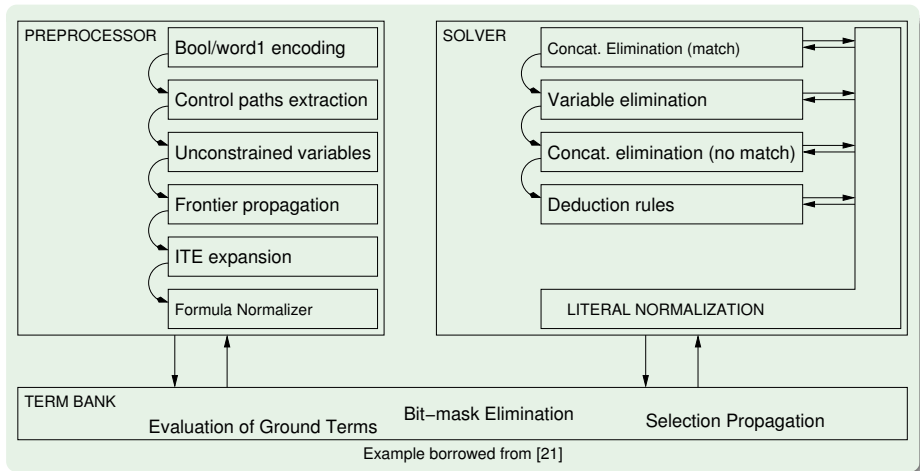
- EX:  
 $\{(x_{[16]}[15 : 0] = (y_{[16]}[15 : 8] :: z_{[16]}[7 : 0]) \ll w_{[16]}[3 : 0]), \dots\}$
- NP-hard
- involve complex word-level operations: word partition/concat, modulo- $2^N$  arithmetic, shifts, bitwise-operations, multiplexers, ...
- $\mathcal{T}$ -solving: combination of rewriting & simplification techniques with either:
  - final encoding into  $\mathcal{LIA}$  [18, 21]
  - final encoding into SAT (**lazy bit-blasting**) [24, 40, 20, 39]

## Eager approach

Most solvers use an **eager** approach for  $\mathcal{BV}$  (e.g., [20]):

- Heavy preprocessing, based on rewriting rules
- bit-blasting

# T-solvers for Bit vectors ( $BV$ ) [cont.]



# $\mathcal{T}$ -solvers for Bit vectors ( $\mathcal{BV}$ ) [cont.]

## Lazy bit-blasting

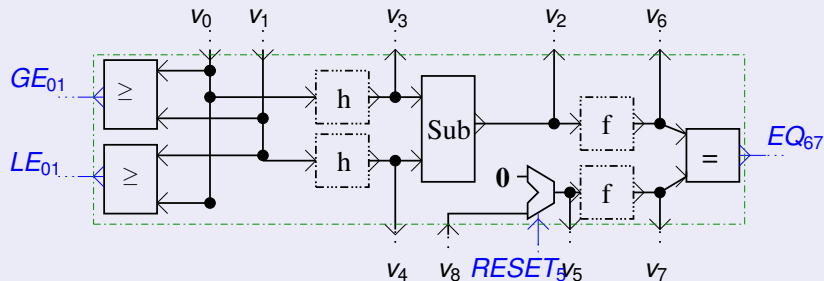
- Two nested SAT solvers
- bit-blast each  $\mathcal{BV}$  atom  $\psi_i$   
 $\implies \Phi \stackrel{\text{def}}{=} \bigwedge_i (A_i \leftrightarrow BB(\psi_i)),$   
 $A_i$  fresh variables labeling  $\mathcal{BV}$ -atoms  $\psi_i$  in  $\varphi$   
 $\implies \varphi$   $\mathcal{BV}$ -satisfiable iff  $\varphi^p \wedge \Phi$  satisfiable
- Exploit SAT under assumptions
  - let  $\mu^p$  an assignment for  $\varphi^p$ , s.t.  $\mu^p \stackrel{\text{def}}{=} \{[\neg]A_1, \dots, [\neg]A_n\}$
  - $\mathcal{T}$ -solver for  $\mathcal{BV}$ :  $SAT_{\text{assumption}}(\Phi, \mu^p)$
  - If UNSAT, generate the **unsat core**  $\eta^p \subseteq \mu^p$ $\implies \neg\eta^p$  used as blocking clause

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 **Efficient SMT solving**
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - **SMT for Combinations of Theories**
- 3 Beyond Solving: Advanced SMT Functionalities
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)



# SMT for combined theories: $SMT(\cup_i \mathcal{T}_i)$

Problem: Many problems can be expressed as SMT problems only in combination of theories  $\cup_i \mathcal{T}_i$  —  $SMT(\cup_i \mathcal{T}_i)$



$$LIA: \quad (GE_{01} \leftrightarrow (v_0 \geq v_1)) \wedge (LE_{01} \leftrightarrow (v_0 \leq v_1)) \wedge$$

$$EUF: \quad (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge$$

$$LIA: \quad (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$$

$$EUF \text{ or } LIA: \quad (\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge$$

$$EUF: \quad (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$EUF \text{ or } LIA: \quad (EQ_{67} \leftrightarrow (v_6 = v_7)) \wedge \dots$$

## SMT for combined theories: $SMT(\mathcal{T}_1 \cup \mathcal{T}_2)$

- Combined theories may be much harder to decide [Pratt'77]
- Solvers have to be combined
- Standard approach for combining  $\mathcal{T}_i$ -solvers:  
(deterministic) Nelson-Oppen/Shostak (N.O.) [56, 58, 67]
  - based on deduction and exchange of equalities on shared variables
  - combined  $\mathcal{T}_i$ -solvers integrated with a SAT tool
- SMT-specific approaches: Delayed Theory Combination [14, 13] and Model-Based Theory Combination [34]
  - based on Boolean search on equalities on shared variables
  - $\mathcal{T}_i$ -solvers integrated directly with a SAT tool

# Background: Pure Formulas

Consider two theories  $\mathcal{T}_1, \mathcal{T}_2$  with equality and disjoint signatures  $\Sigma_1, \Sigma_2$

- W.l.o.g. we assume all input formulas  $\phi \in \mathcal{T}_1 \cup \mathcal{T}_2$  are **pure**.
  - A formula  $\phi$  is **pure** iff every atom in  $\phi$  is  $i$ -pure for some  $i \in \{1, 2\}$ .
  - An atom/literal in  $\phi$  is  **$i$ -pure** if only  $=$ , variables and symbols from  $\Sigma_i$  can occur in  $\phi$

## Purification:

Maps a formula into an equisatisfiable pure formula by labeling terms with fresh variables

$$(f(\underbrace{x + 3y}_w) = g(\underbrace{2x - y}_t)) \quad [not\ pure]$$

$$\Downarrow$$
$$(w = x + 3y) \wedge (t = 2x - y) \wedge (f(w) = g(t)) \quad [pure]$$

# Exercise

- Purify the following  $\mathcal{LIA} \cup \mathcal{EUF} \cup \mathcal{AR}$ -formula (see beginning of chapter):

$$\varphi \stackrel{\text{def}}{=} (d \geq 0) \wedge (d < 1) \wedge \\ ((f(d) = f(0)) \rightarrow (\text{read}(\text{write}(V, i, x), i + d) = x + 1))$$

# Background: Interface equalities

## Interface variables & equalities

- A variable  $v$  occurring in a pure formula  $\phi$  is an **interface variable** iff it occurs in both 1-pure and 2-pure atoms of  $\phi$ .
- An equality  $(v_i = v_j)$  is an **interface equality** for  $\phi$  iff  $v_i, v_j$  are interface variables for  $\phi$ .
- We denote the interface equality  $v_i = v_j$  by “ $e_{ij}$ ”

## Example:

$\mathcal{LIA}$ :  $(GE_{01} \leftrightarrow (v_0 \geq v_1)) \wedge (LE_{01} \leftrightarrow (v_0 \leq v_1)) \wedge$   
 $\mathcal{EUF}$ :  $(v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge$   
 $\mathcal{LIA}$ :  $(v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$   
 $\mathcal{EUF}$  or  $\mathcal{LIA}$ :  $(\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge$   
 $\mathcal{EUF}$ :  $(v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$   
 $\mathcal{EUF}$  or  $\mathcal{LIA}$ :  $(EQ_{67} \leftrightarrow (v_6 = v_7)) \wedge \dots$

$v_0, v_1, v_2, v_3, v_4, v_5$  are interface variables,  $v_6, v_7, v_8$  are not  
 $\implies (v_0 = v_1)$  is an interface equality,  $(v_0 = v_6)$  is not.

# Background: Stably-infinite & Convex Theories

## Stably-infinite Theories

A  $\Sigma$ -theory  $\mathcal{T}$  is **stably-infinite** iff every quantifier-free  $\mathcal{T}$ -satisfiable formula is satisfiable in an infinite model of  $\mathcal{T}$ .

- $\mathcal{EUF}$ ,  $\mathcal{DL}$ ,  $\mathcal{LRA}$ ,  $\mathcal{LIA}$  are stably-infinite
- (fixed-width) bit-vector theories are not stably-infinite

Intuition: **a variable can be given an infinite amount of distinct values**

## Convex Theories

A  $\Sigma$ -theory  $\mathcal{T}$  is **convex** iff, for every collection  $l_1, \dots, l_k, l', l''$  of literals in  $\mathcal{T}$  s.t.  $l', l''$  are in the form  $(x = y)$ ,  $x, y$  being variables, we have that:  $\{l_1, \dots, l_k\} \models_{\mathcal{T}} (l' \vee l'') \iff \{l_1, \dots, l_k\} \models_{\mathcal{T}} l' \text{ or } \{l_1, \dots, l_k\} \models_{\mathcal{T}} l''$

- $\mathcal{EUF}$ ,  $\mathcal{DL}$ ,  $\mathcal{LRA}$  are convex
- $\mathcal{LIA}$  is not convex:
  - $\{(v_0 = 0), (v_1 = 1), (v \geq v_0), (v \leq v_1)\} \models ((v = v_0) \vee (v = v_1))$ ,
  - $\{(v_0 = 0), (v_1 = 1), (v \geq v_0), (v \leq v_1)\} \not\models (v = v_0)$
  - $\{(v_0 = 0), (v_1 = 1), (v \geq 0), (v \leq v_1)\} \not\models (v = v_1)$

Intuition: **non-convexity produces “case splits”**

# SMT( $\bigcup_i \mathcal{T}_i$ ) via “classic” Nelson-Oppen

## Main Problem

- One predicate shared between distinct theories  $\mathcal{T}_i$ : **equality “=”**
- Given  $\mu \stackrel{\text{def}}{=} \bigcup_i \mu_i$  s.t. each  $\mu_i$  contains i-pure literals
  - distinct  $\mathcal{T}_i$ -solver can be invoked separately on each  $\mu_i$ ...
  - ...producing distinct  $\mathcal{T}_i$ -specific models  $\mathcal{M}_i$
- Problem: **all models must agree on interface equalities:**

$$\mathcal{M}_i \models_{\mathcal{T}_i} (v_k = v_l) \text{ iff } \mathcal{M}_j \models_{\mathcal{T}_j} (v_k = v_l),$$

for every pair of shared variables  $v_k, v_l$

## Main idea

Combine two or more  $\mathcal{T}_i$ -solvers into one ( $\bigcup_i \mathcal{T}_i$ )-solver via **Nelson-Oppen/Shostak (N.O.) combination procedure** [57, 68]

- based on the deduction and exchange of equalities between shared variables/terms (**interface equalities,  $e_{ij}$ s**)
- important improvements and evolutions [62, 7, 36]

## Schema of N.O. combination of $\mathcal{T}$ -solvers: $\text{no}(\mathcal{T}_1, \mathcal{T}_2)$

For  $i \in \{1, 2\}$ , let  $\mathcal{T}_i$  be a stably infinite theory admitting a satisfiability  $\mathcal{T}_i$ -solver, and  $\mu_i$  a set of  $i$ -pure literals.

We want to decide the  $\mathcal{T}_1 \cup \mathcal{T}_2$ -satisfiability of  $\mu_1 \cup \mu_2$

- each  $\mathcal{T}_i$ -solver, in turn
  - checks the  $\mathcal{T}_i$ -satisfiability of  $\mu_i$ ,
  - deduces all the (disjunctions of) interface equalities which derive from  $\mu_i$
  - passes them to  $\mathcal{T}_j$ -solve,  $j \neq i$ , which adds them to  $\mu_j$

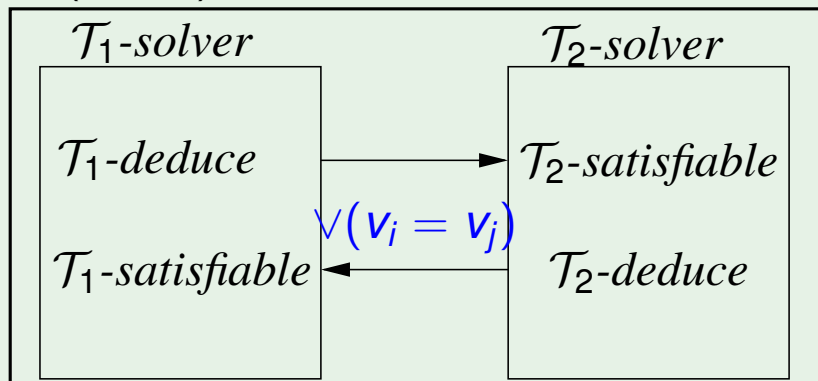
until either:

- one  $\mathcal{T}_i$ -solver detects unsatisfiability ( $\mu_1 \cup \mu_2$  is  $\mathcal{T}_1 \cup \mathcal{T}_2$ -unsat)
- no more deductions are possible ( $\mu_1 \cup \mu_2$  is  $\mathcal{T}_1 \cup \mathcal{T}_2$ -sat)
- disjunctions of literals (due to non-convexity) force case-splitting



# Schema of N.O. combination of T-solvers: $\text{no}(\mathcal{T}_1, \mathcal{T}_2)$

$\text{no}(\mathcal{T}_1, \mathcal{T}_2)$



## N.O. Example (Convex Theory)

$\mathcal{EUF}$  :  $(v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$

$\mathcal{LRA}$  :  $(v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$

$\text{Both}$  :  $(\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$

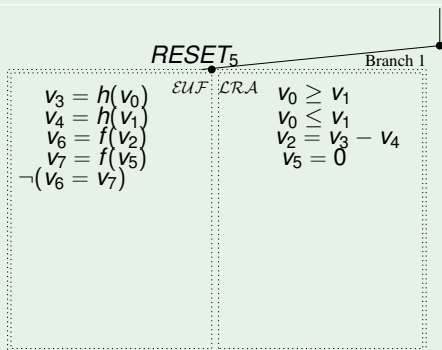


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

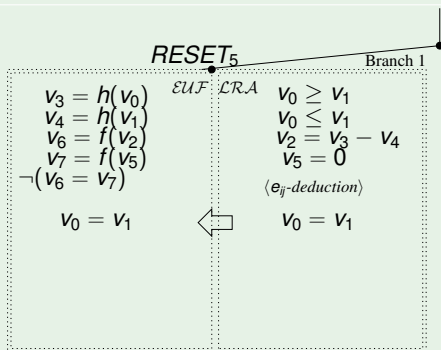


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

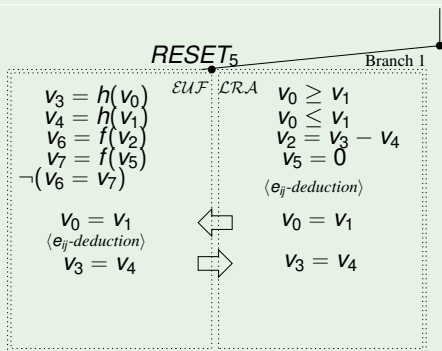


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

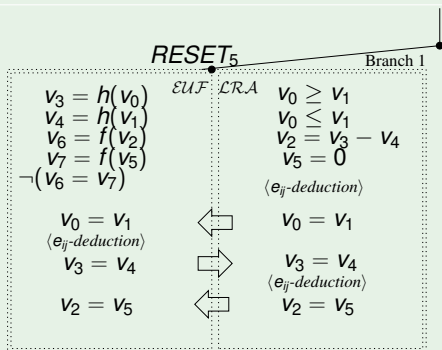


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

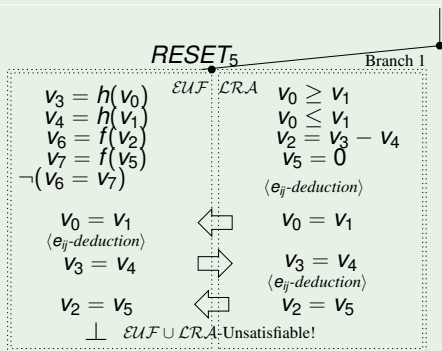


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

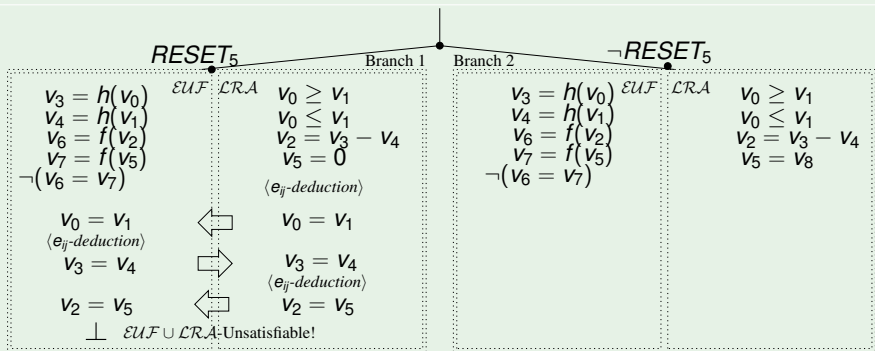


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$



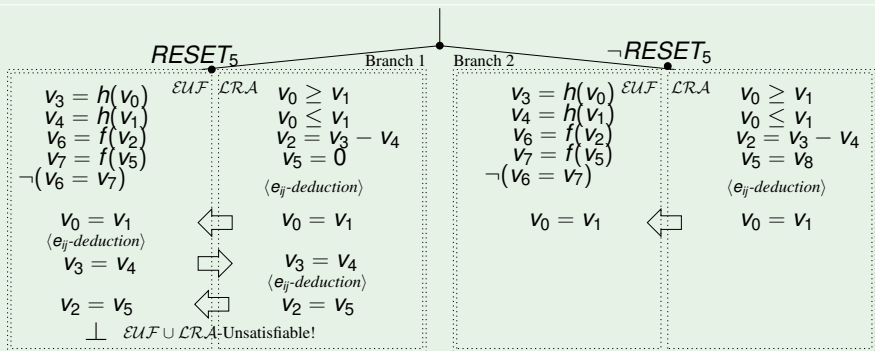


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

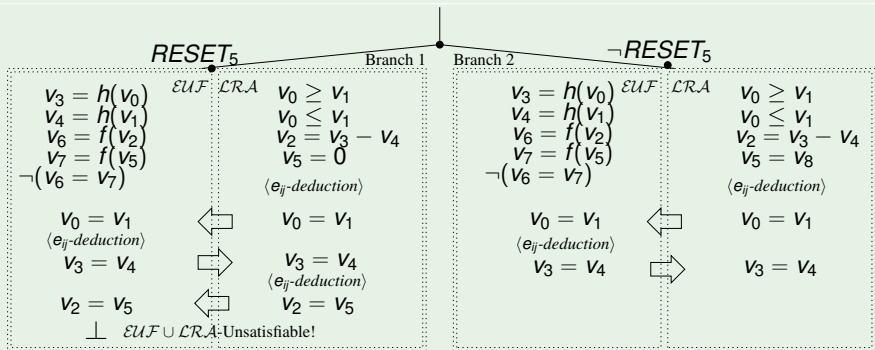


# N.O. Example (Convex Theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$

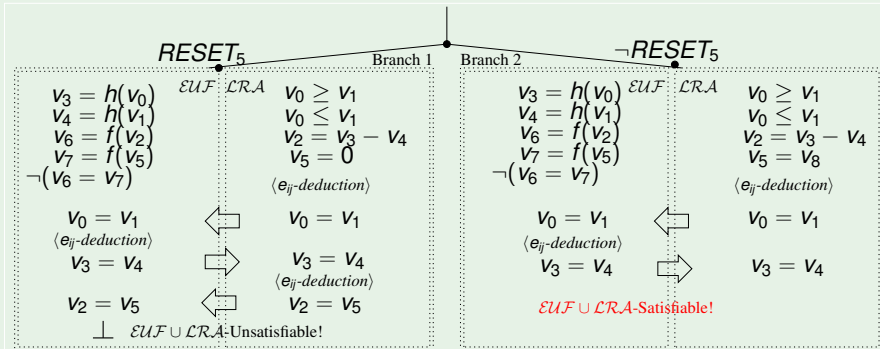


# N.O. Example (Convex Theory)

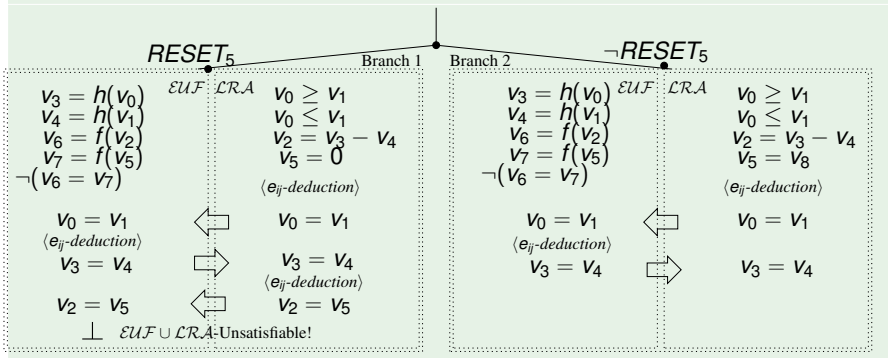
$$EUF : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$LRA : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$$

$$Both : (\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$



# N.O.: example (convex theory) [cont.]



*EUF*-conflict :

$$((v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge \neg(v_6 = v_7) \wedge (v_2 = v_5)) \rightarrow \perp$$

*LRA*-deduction :

$$((v_2 = v_3 - v_4) \wedge (v_5 = 0) \wedge (v_3 = v_4)) \rightarrow (v_2 = v_5)$$

*EUF*-deduction :

$$((v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_0 = v_1)) \rightarrow (v_3 = v_4)$$

*LRA*-deduction :

$$((v_0 \geq v_1) \wedge (v_0 \leq v_1)) \rightarrow (v_0 = v_1)$$

⇒

*EUF* ∪ *LRA*-conflict :

$$((v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge \neg(v_6 = v_7) \wedge (v_2 = v_3 - v_4) \wedge (v_5 = 0) \wedge (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_0 \geq v_1)) \rightarrow \perp$$

For the previous N.O. example:

- write the (minimal) clauses corresponding to each  $e_{ij}$ -deduction
- find the final conflict clauses by resolving the  $e_{ij}$ -deduction clauses

# N.O.: example (non-convex theory)

$\mu\mathcal{L}IA$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

$\mu\mathcal{E}UF$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$

# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$

$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$

# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$



$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$

$$v_1 = v_3$$



# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$



$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$

$$v_1 = v_3$$

*( $e_{ij}$ -deduction)*

$$v_5 = v_6$$

# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$

$$v_5 = v_6$$

*( $e_{ij}$ -deduction)*

$$v_2 = v_3 \vee v_2 = v_4$$

$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$

$$v_1 = v_3$$

*( $e_{ij}$ -deduction)*

$$v_5 = v_6$$

# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$

$$v_5 = v_6$$

*( $e_{ij}$ -deduction)*

$$v_2 = v_3 \vee v_2 = v_4$$

$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$

$$v_1 = v_3$$

*( $e_{ij}$ -deduction)*

$$v_5 = v_6$$

$$v_2 = v_3$$

$\perp$

# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$

$$v_5 = v_6$$

*( $e_{ij}$ -deduction)*

$$v_2 = v_3 \vee v_2 = v_4$$

$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$



$$v_1 = v_3$$

*( $e_{ij}$ -deduction)*



$$v_5 = v_6$$



$$\begin{array}{ll} v_2 = v_3 & v_2 = v_4 \\ \perp & \perp \end{array}$$

# N.O.: example (non-convex theory)

$\mu\mathcal{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$

$$v_5 = v_6$$

*( $e_{ij}$ -deduction)*

$$v_2 = v_3 \vee v_2 = v_4$$

$\mu\mathcal{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$



$$v_1 = v_3$$

*( $e_{ij}$ -deduction)*

$$v_5 = v_6$$



$$v_1 = v_4$$

SAT!



$$v_2 = v_3$$

$\perp$

$$v_2 = v_4$$

$\perp$

# N.O.: example (non-convex theory)

$\mu_{LIA}$

$$\begin{array}{ll} v_1 \geq 0 & v_5 = v_4 - 1 \\ v_1 \leq 1 & v_3 = 0 \\ v_2 \geq v_6 & v_4 = 1 \\ v_2 \leq v_6 + 1 & \end{array}$$

*( $e_{ij}$ -deduction)*

$$v_1 = v_3 \vee v_1 = v_4$$

$$v_5 = v_6$$

*( $e_{ij}$ -deduction)*

$$v_2 = v_3 \vee v_2 = v_4$$

$\mu_{EUF}$

$$\begin{array}{l} \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}$$



$$v_1 = v_3 \quad v_1 = v_4$$

*( $e_{ij}$ -deduction)*

SAT!



$$v_5 = v_6$$

3  $e_{ij}$ -deductions,

3 branches



$$\begin{array}{ll} v_2 = v_3 & v_2 = v_4 \\ \perp & \perp \end{array}$$

# SMT( $\bigcup_i \mathcal{T}_i$ ) via “classic” Nelson-Oppen

## Main idea

Combine two or more  $\mathcal{T}_i$ -solvers into one  $(\bigcup_i \mathcal{T}_i)$ -solver via Nelson-Oppen/Shostak (N.O.) combination procedure [57, 68]

- based on the deduction and exchange of equalities between shared variables/terms (interface equalities,  $e_{ij}$ s)
- important improvements and evolutions [62, 7, 36]
- drawbacks [22, 23]:
  - require (possibly expensive) deduction capabilities from  $\mathcal{T}_i$ -solvers
  - [ with non-convex theories ] case-splits forced by the deduction of disjunctions of  $e_{ij}$ 's
  - generate (typically long)  $(\bigcup_i \mathcal{T}_i)$ -lemmas, without interface equalities  $\implies$  no backjumping & learning from  $e_{ij}$ -reasoning

# SMT( $\bigcup_i \mathcal{T}_i$ ) via Delayed Theory Combination (DTC)

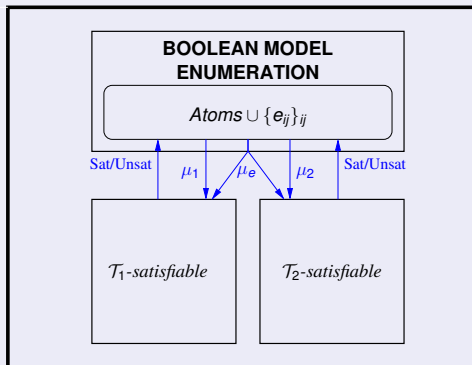
## Main idea

Delegate to the CDCL SAT solver part/most of the (possibly very expensive) reasoning effort on interface equalities previously due to the  $\mathcal{T}_i$ -solvers ( $e_{ij}$ -deduction, case-split). [14, 15, 23]

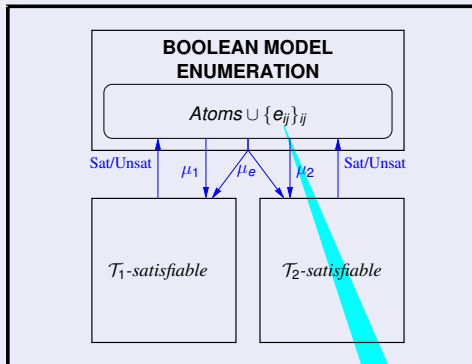
- based on Boolean reasoning on interface equalities via CDCL (plus  $\mathcal{T}$ -propagation)
- important improvements and evolutions [34, 9]
- feature wrt N.O. [22, 23]
  - do not require (possibly expensive) deduction capabilities from  $\mathcal{T}_i$ -solvers
  - with non-convex theories, case-splits on  $e_{ij}$ 's handled by SAT
  - generate  $\mathcal{T}_i$ -lemmas with interface equalities  
 $\implies$  backjumping & learning from  $e_{ij}$ -reasoning



# DTC: Basic schema



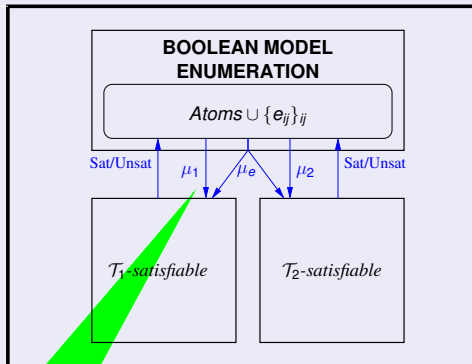
# DTC: Basic schema



The boolean solver assigns values not only to atoms in  $Atoms(\phi)$ , but also to interface equalities  $\{(v_i = v_j)\}_{ij}$ :

$$\mu = \mu_1 \cup \mu_2 \cup \mu_e, \quad \mu_e := \{[\neg](v_i = v_j) \mid v_i, v_j \in \mu_1 \cup \mu_2\}$$

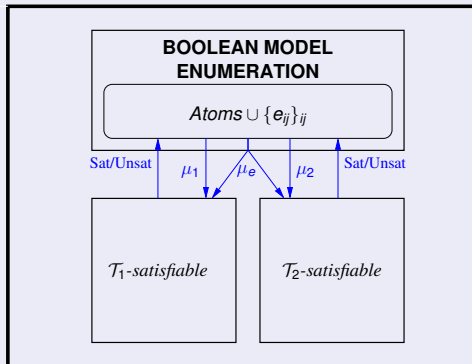
# DTC: Basic schema



Each  $\mathcal{T}_i$ -solver interacts only with the boolean solver

- receives  $\mu'_i := \mu_i \cup \mu_e$  from Bool
- checks the  $\mathcal{T}_i$ -satisfiability of  $\mu'_i$

# DTC: Basic schema



...until either:

- some  $\mu$  propositionally satisfies  $\phi$  and both  $\mu'_i := \mu_i \cup \mu_e$  are  $\mathcal{T}_i$ -consistent  
 $\implies (\phi \text{ is } \mathcal{T}_1 \cup \mathcal{T}_2\text{-sat})$
- no more assignment  $\mu$  are available  
 $\implies (\phi \text{ is } \mathcal{T}_1 \cup \mathcal{T}_2\text{-unsat})$

## DTC: enhanced schema

- **CDCL-based assignment enumeration** on  $Atoms(\phi) \cup \{e_{ij}\}_{ij}$ ,  
⇒ benefits of state-of-the-art SAT techniques
- **Early pruning**: invoke the  $\mathcal{T}_i$ -solver's before every Boolean decision  
⇒ total assignments generated only when strictly necessary
- **Branching**: branching on  $e_{ij}$ 's postponed  
⇒ Boolean search on  $e_{ij}$ 's performed only when strictly necessary
- **Theory-Backjumping & Learning**:  $e_{ij}$ 's are involved in conflicts  
⇒  $e_{ij}$ 's can be assigned by unit propagation
- **Theory-deduction & learning**: if  $\mathcal{T}_i$ -solver deduces unassigned literals  $I$  on  $Atoms(\phi) \cup \{e_{ij}\}_{ij}$ 
  - $I$  is passed back to the Boolean solver, which unit-propagates it
  - the deduction  $\mu' \models I$  is learned as a clause  $\mu' \rightarrow I$  (deduction clause)
- ...

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{EUF}$ :	$\mu_{LIA}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$$\begin{array}{l} \mu_{\text{EUF}}: \\ \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array} \quad \left. \vphantom{\begin{array}{l} \mu_{\text{EUF}}: \\ \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array}} \right\} \begin{array}{l} \mu_{\text{LIA}}: \\ v_1 \geq 0 \\ v_1 \leq 1 \\ v_2 \geq v_6 \\ v_2 \leq v_6 + 1 \end{array} \quad \begin{array}{l} v_5 = v_4 - 1 \\ v_3 = 0 \\ v_4 = 1 \end{array}$$

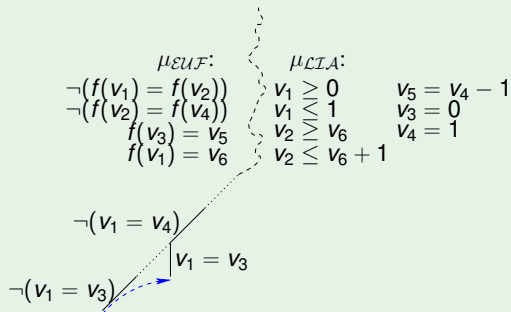
$$\neg(v_1 = v_4)$$

$$\neg(v_1 = v_3)$$

$\text{LIA-unsat}, C_{13}$

$$C_{13} : (\mu'_{\text{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)



$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$



# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{\mathcal{EUF}}:$	$\mu_{\mathcal{LIA}}:$	
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	

$$\neg(v_1 = v_4)$$

$$\neg(v_1 = v_3) \quad | \quad v_1 = v_3$$

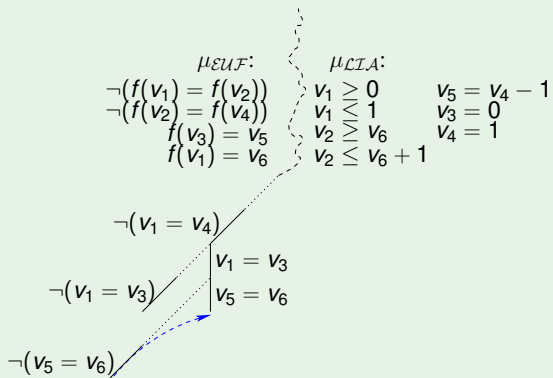
$$\neg(v_5 = v_6)$$

$\mathcal{EUF}$ -unsat,  $C_{56}$

$$C_{13} : (\mu'_{\mathcal{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

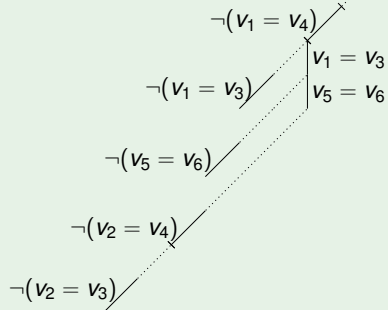


$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{EUF}$ :	$\mu_{LIA}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	



$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

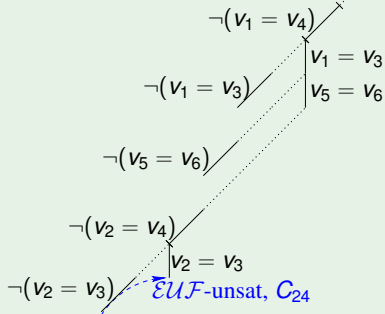
$$C_{23} : (\mu''_{LIA} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$LIA$ -unsat,  $C_{23}$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{\text{EUF}}:$	$\mu_{\text{LIA}}:$
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$
$f(v_3) = v_5$	$v_2 \geq v_6$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$

$$\begin{aligned} v_5 &= v_4 - 1 \\ v_3 &= 0 \\ v_4 &= 1 \end{aligned}$$



$$C_{13} : (\mu'_{\text{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

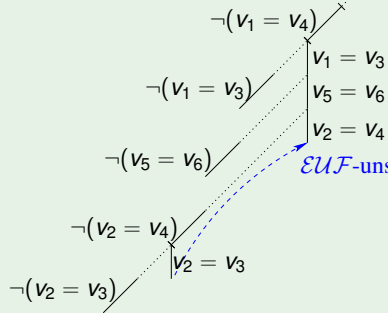
$$C_{56} : (\mu'_{\text{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

$$C_{23} : (\mu''_{\text{LIA}} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{\text{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{\mathcal{EUF}}$ :	$\mu_{\mathcal{LIA}}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	



$\mathcal{EUF}$ -unsat,  $C_{14}$

$$C_{13} : (\mu'_{\mathcal{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

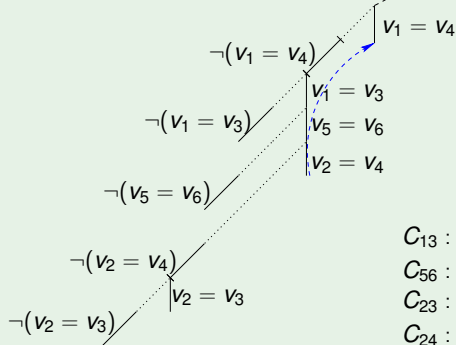
$$C_{23} : (\mu''_{\mathcal{LIA}} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{\mathcal{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

$$C_{14} : (\mu'''_{\mathcal{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{\mathcal{EUF}}$ :	$\mu_{\mathcal{LIA}}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	



$$C_{13} : (\mu'_{\mathcal{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

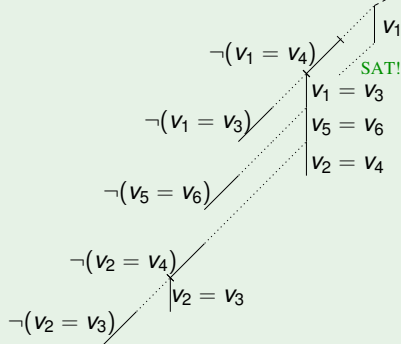
$$C_{23} : (\mu''_{\mathcal{LIA}} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{\mathcal{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

$$C_{14} : (\mu'''_{\mathcal{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$$\begin{array}{l}
 \mu_{\mathcal{EUF}}: \\
 \neg(f(v_1) = f(v_2)) \\
 \neg(f(v_2) = f(v_4)) \\
 f(v_3) = v_5 \\
 f(v_1) = v_6 \\
 \mu_{\mathcal{LIA}}: \\
 v_1 \geq 0 \\
 v_1 \leq 1 \\
 v_2 \geq v_6 \\
 v_2 \leq v_6 + 1 \\
 v_5 = v_4 - 1 \\
 v_3 = 0 \\
 v_4 = 1
 \end{array}$$



$$C_{13} : (\mu'_{\mathcal{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

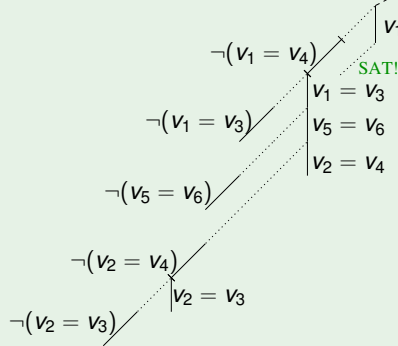
$$C_{23} : (\mu''_{\mathcal{LIA}} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{\mathcal{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

$$C_{14} : (\mu'''_{\mathcal{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$

# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{EUF}$ :	$\mu_{LIA}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	



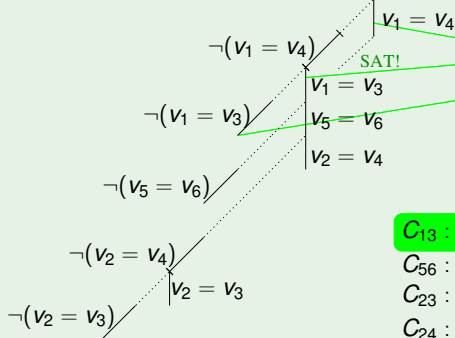
SAT! 6 branches

- $C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$
- $C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$
- $C_{23} : (\mu''_{LIA} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$
- $C_{24} : (\mu''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$
- $C_{14} : (\mu'''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$



# DTC: example w.out $\mathcal{T}$ -prop. (non-convex theory)

$$\begin{array}{l}
 \mu_{EUF}: \\
 \neg(f(v_1) = f(v_2)) \\
 \neg(f(v_2) = f(v_4)) \\
 f(v_3) = v_5 \\
 f(v_1) = v_6 \\
 \hline
 \mu_{LIA}: \\
 v_1 \geq 0 \\
 v_1 \leq 1 \\
 v_2 \geq v_6 \\
 v_2 \leq v_6 + 1 \\
 \hline
 v_5 = v_4 - 1 \\
 v_3 = 0 \\
 v_4 = 1
 \end{array}$$



Mimics the  $\theta_{ij}$ -deduction

$\mu'_{LIA} \models_{LIA} ((v_1 = v_3) \vee (v_1 = v_4))$   
 and the two branches  $(v_1 = v_3), (v_1 = v_4)$

$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

$$C_{23} : (\mu''_{LIA} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

$$C_{14} : (\mu'''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$

## DTC: example with $\mathcal{T}$ -prop. (non-convex theory)

$$\begin{array}{l} \mu_{EUF}: \\ \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array} \quad \begin{array}{l} \mu_{LIA}: \\ v_1 \geq 0 \\ v_1 \leq 1 \\ v_2 \geq v_6 \\ v_2 \leq v_6 + 1 \end{array} \quad \begin{array}{l} v_5 = v_4 - 1 \\ v_3 = 0 \\ v_4 = 1 \end{array}$$

# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)

$$\begin{array}{l} \mu_{EUF}: \\ \neg(f(v_1) = f(v_2)) \\ \neg(f(v_2) = f(v_4)) \\ f(v_3) = v_5 \\ f(v_1) = v_6 \end{array} \quad \begin{array}{l} \mu_{LIA}: \\ v_1 \geq 0 \\ v_1 \leq 1 \\ v_2 \geq v_6 \\ v_2 \leq v_6 + 1 \end{array} \quad \begin{array}{l} v_5 = v_4 - 1 \\ v_3 = 0 \\ v_4 = 1 \end{array}$$

$\mathcal{LIA}$ -deduce  $(v_1 = v_4) \vee (v_1 = v_3)$ ,  $C_{13}$

$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{EUF}$ :	$\mu_{LIA}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 > 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	

$\neg(v_1 = v_4)$	$v_1 = v_3$
-------------------	-------------

$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{\mathcal{EUF}}:$	$\mu_{\mathcal{LIA}}:$	
$\neg(f(v_1) = f(v_2))$	$v_1 > 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	

$$\neg(v_1 = v_4) \left| \begin{array}{l} v_1 = v_3 \\ v_5 = v_6 \end{array} \right. \mathcal{EUF}\text{-deduce } (v_5 = v_6), C_{56}$$

$$C_{13} : (\mu'_{\mathcal{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)

$\mu_{EUF}$ :	$\mu_{LIA}$ :	
$\neg(f(v_1) = f(v_2))$	$v_1 > 0$	$v_5 = v_4 - 1$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$	$v_3 = 0$
$f(v_3) = v_5$	$v_2 \geq v_6$	$v_4 = 1$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$	

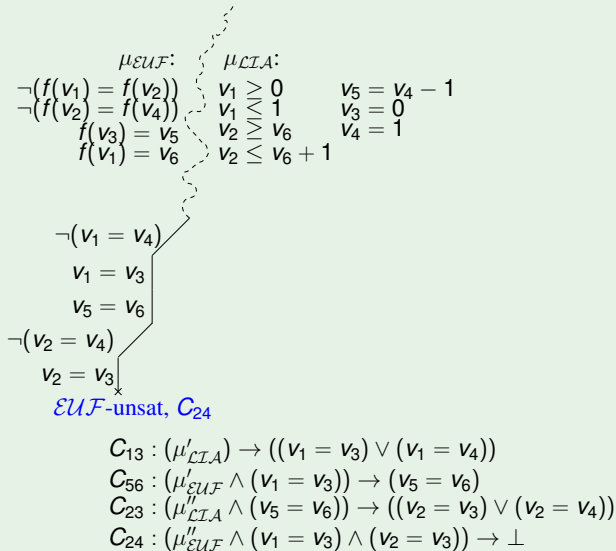
$\neg(v_1 = v_4)$	$LIA$ -deduce $(v_2 = v_4) \vee (v_2 = v_3), C_{23}$
$v_1 = v_3$	
$v_5 = v_6$	

$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

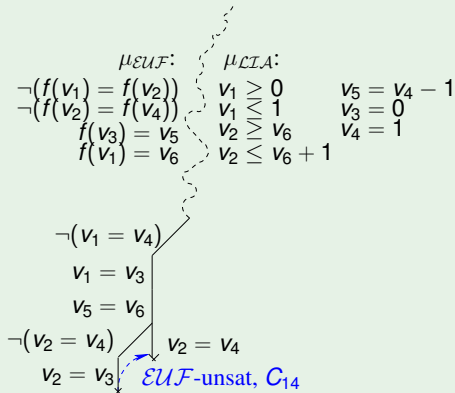
$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

$$C_{23} : (\mu''_{LIA} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)



# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)



$$C_{13} : (\mu'_{\text{LIA}}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{\text{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

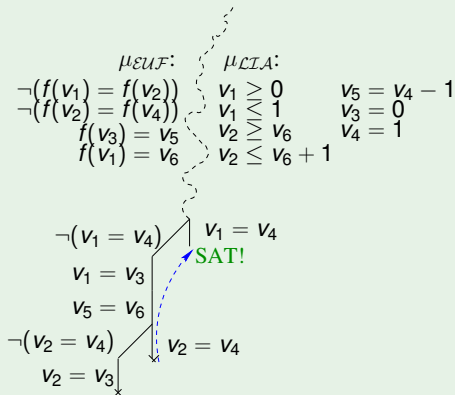
$$C_{23} : (\mu''_{\text{LIA}} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{\text{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

$$C_{14} : (\mu'''_{\text{EUF}} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$



# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)



$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

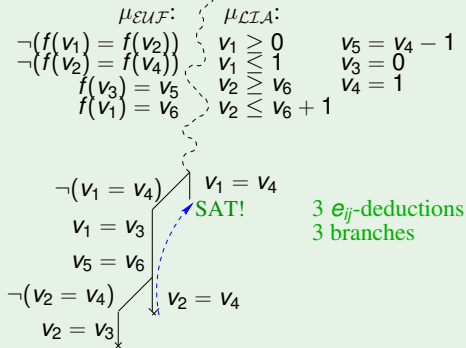
$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

$$C_{23} : (\mu''_{LIA} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

$$C_{14} : (\mu'''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$

# DTC: example with $\mathcal{T}$ -prop. (non-convex theory)



$$C_{13} : (\mu'_{LIA}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

$$C_{23} : (\mu''_{LIA} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$$

$$C_{24} : (\mu''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_3)) \rightarrow \perp$$

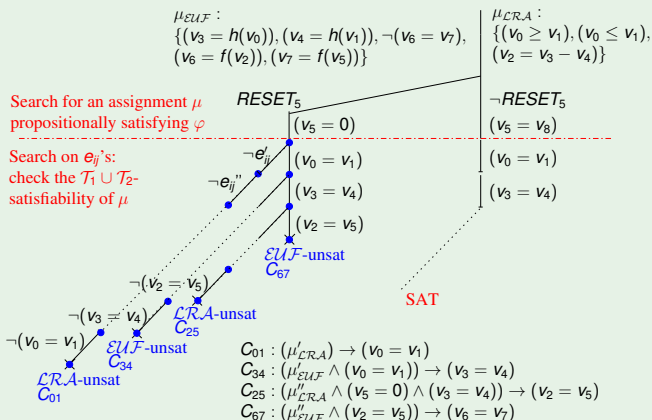
$$C_{14} : (\mu'''_{EUF} \wedge (v_1 = v_3) \wedge (v_2 = v_4)) \rightarrow \perp$$

# DTC: example without $\mathcal{T}$ -propagation (convex theory)

$$\mathcal{EUF} : (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{LRA} : (v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (\text{RESET}_5 \rightarrow (v_5 = 0)) \wedge$$

$$\text{Both} : (\neg \text{RESET}_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$$



# DTC: example with $\mathcal{T}$ -propagation (convex theory)

$\mathcal{EUF}$ :  $(v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$

$\mathcal{LRA}$ :  $(v_0 \geq v_1) \wedge (v_0 \leq v_1) \wedge (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$

$Both$ :  $(\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge \neg(v_6 = v_7).$

$\mu_{\mathcal{EUF}}$ :

$\{(v_3 = h(v_0)), (v_4 = h(v_1)), \neg(v_6 = v_7),$   
 $(v_6 = f(v_2)), (v_7 = f(v_5))\}$

$RESET_5$

$\mu_{\mathcal{LRA}}$ :

$\{(v_0 \geq v_1), (v_0 \leq v_1),$   
 $(v_2 = v_3 - v_4)\}$

$\neg RESET_5$

$(v_5 = 0)$

$(v_5 = v_8)$

$\mathcal{LRA}$ -deduce  $(v_0 = v_1)$

learn  $C_{01}$

$(v_0 = v_1)$

$(v_0 = v_1)$

$\mathcal{EUF}$ -deduce  $(v_3 = v_4)$

learn  $C_{34}$

$(v_3 = v_4)$

$(v_3 = v_4)$

$\mathcal{LRA}$ -deduce  $(v_2 = v_5)$

learn  $C_{25}$

$(v_2 = v_5)$

SAT

$\mathcal{EUF}$ -unsat

$C_{67}$

$C_{01} : (\mu'_{\mathcal{LRA}}) \rightarrow (v_0 = v_1)$

$C_{34} : (\mu'_{\mathcal{EUF}} \wedge (v_0 = v_1)) \rightarrow (v_3 = v_4)$

$C_{25} : (\mu''_{\mathcal{LRA}} \wedge (v_5 = 0) \wedge (v_3 = v_4)) \rightarrow (v_2 = v_5)$

$C_{67} : (\mu''_{\mathcal{EUF}} \wedge (v_2 = v_5)) \rightarrow (v_6 = v_7)$

## DTC + Model-based heuristic (aka Model-Based Theory Combination) [34]

- Initially, no interface equalities generated
- When a model is found, check against all the possible interface equalities
  - If  $\mathcal{T}_1$  and  $\mathcal{T}_2$  agree on the implied equalities, then return SAT
  - Otherwise, branch on equalities implied by  $\mathcal{T}_1$ -model but not by  $\mathcal{T}_2$ -model
- “Optimistic” approach, similar to axiom instantiation

# Exercises

For each of the previous DTC examples:

- write the (minimal) clauses corresponding to each  $e_{ij}$ -deduction (as clauses rather than as implications)
- compute the conflict-analysis steps leading to the backjumping steps in the figures.

## Exercise

Let  $\mathcal{LRA}$  be the logic of linear arithmetic over the rationals and  $\mathcal{EUF}$  be the logic of equality and uninterpreted functions. Consider the following pure formula  $\varphi$  in the combined logic  $\mathcal{LRA} \cup \mathcal{EUF}$ :

$$(x = 1.0) \wedge (h = 1.0) \wedge (k = 1.0) \wedge (y = 2h - k) \wedge (z < w) \\ (z = f(x)) \wedge (w = f(y))$$

- 1 Say which variables are interface variables,
- 2 list the interface equalities for this formula (modulo symmetry),
- 3 decide whether this formula is  $\mathcal{LRA} \cup \mathcal{EUF}$ -satisfiable or not, using both Nelson-Oppen or Delayed Theory Combination.

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 **Beyond Solving: Advanced SMT Functionalities**
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)



# Beyond Solving: advanced SAT & SMT functionalities

Advanced SMT functionalities (very important in FV):

- Building proofs of  $\mathcal{T}$ -unsatisfiability
- Extracting  $\mathcal{T}$ -unsatisfiable Cores
- Computing Craig interpolants
- Performing All-SMT and Predicate Abstraction
- Deciding/optimizing SMT problems with costs

# Outline

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 **Beyond Solving: Advanced SMT Functionalities**
  - **Proofs and Unsatisfiable Cores**
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

# Building (Resolution) Proofs of $\mathcal{T}$ -Unsatisfiability

## Resolution proof of $\mathcal{T}$ -unsatisfiability

Very similar to building proofs with plain SAT:

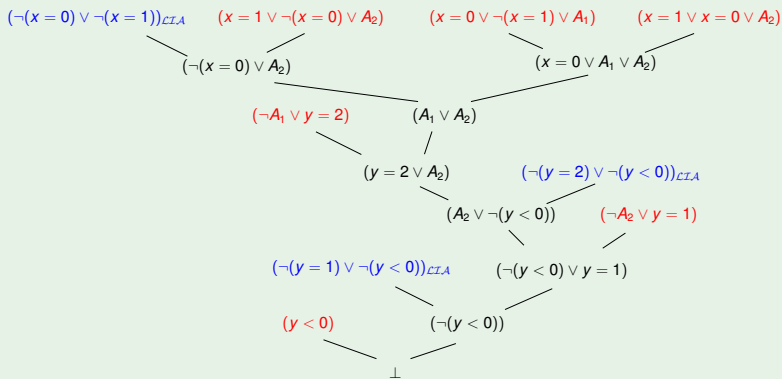
- resolution proofs whose leaves are original clauses and  $\mathcal{T}$ -lemmas returned by the  $\mathcal{T}$ -solver (i.e.,  $\mathcal{T}$ -conflict and  $\mathcal{T}$ -deduction clauses)
- built by backward traversal of implication graphs, as in CDCL SAT
- Sub-proofs of  $\mathcal{T}$ -lemmas can be built in some  $\mathcal{T}$ -specific deduction framework if requested

Important for:

- certifying  $\mathcal{T}$ -unsatisfiability results
- computing unsatisfiable cores
- computing interpolants

# Building Proofs of $\mathcal{T}$ -Unsatisfiability: example

$$(x = 0 \vee \neg(x = 1) \vee A_1) \wedge (x = 0 \vee x = 1 \vee A_2) \wedge (\neg(x = 0) \vee x = 1 \vee A_2) \wedge \\ (\neg A_2 \vee y = 1) \wedge (\neg A_1 \vee x + y > 3) \wedge (y < 0) \wedge (A_2 \vee x - y = 4) \wedge (y = 2 \vee \neg A_1) \wedge (x \geq 0),$$



relevant original clauses, irrelevant original clauses,  $\mathcal{T}$ -lemmas

## Example: proof on non-strict $\mathcal{LRA}$ inequalities

- A proof of unsatisfiability for a set of non-strict  $\mathcal{LRA}$  inequalities can be obtained by building a linear combination of such inequalities, each time eliminating one or more variables, until you get a contradictory inequality on constant values.
- Example:

$$\varphi \stackrel{\text{def}}{=} (0 \leq x_1 - 3x_2 + 1), (0 \leq x_1 + x_2), (0 \leq x_3 - 2x_1 - 3), (0 \leq 1 - 2x_3).$$

A proof of unsatisfiability  $P$  for  $\varphi$  is the following:

$$\frac{\begin{array}{l} (0 \leq x_1 - 3x_2 + 1) \quad (0 \leq x_1 + x_2) \\ \text{COMB } (0 \leq 4x_1 + 1) \text{ with coeffs } 1 \text{ and } 3 \end{array}}{\text{COMB } (0 \leq -4) \text{ with coeffs } 1 \text{ and } 1} \quad \frac{\begin{array}{l} (0 \leq x_3 - 2x_1 - 3) \quad (0 \leq 1 - 2x_3) \\ \text{COMB } (0 \leq -4x_1 - 5) \text{ with coeffs } 2 \text{ and } 1 \end{array}}{\text{COMB } (0 \leq -4) \text{ with coeffs } 1 \text{ and } 1}$$

- It is possible to produce such proof from an unsatisfiable tableau in Simplex procedure for  $\mathcal{LRA}$  [27, 29]
- It is straightforward to produce such proof from a negative cycle in the graph-based procedure for  $\mathcal{DL}$  [27, 29]

# Extraction of $\mathcal{T}$ -unsatisfiable cores

## The problem

Given a  $\mathcal{T}$ -unsatisfiable set of clauses, extract from it a (possibly small/minimal/minimum)  $\mathcal{T}$ -unsatisfiable subset ( $\mathcal{T}$ -unsatisfiable core)

- Wide literature in SAT
- Some implementations, very few literature for SMT [26, 51]
- We recognize three approaches:
  - **Proof-based** approach (CVC4, MathSAT):  
byproduct of finding a resolution proof
  - **Assumption-based** approach (Yices):  
use extra variables labeling clauses, as in the plain Boolean case
  - **Lemma-Lifting** approach [26] :  
use an external (possibly-optimized) Boolean unsat-core extractor

# The proof-based approach to $\mathcal{T}$ -unsat cores

Idea (adapted from [74])

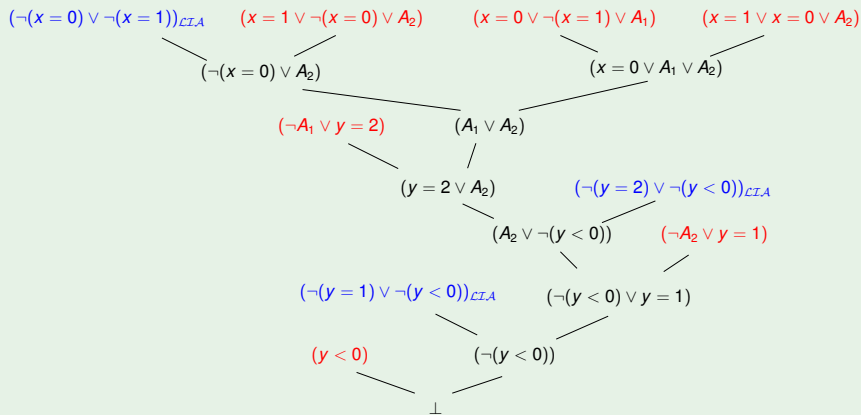
Unsatisfiable core of  $\varphi$ :

- in SAT: the set of leaf clauses of a resolution proof of unsatisfiability of  $\varphi$
- in SMT( $\mathcal{T}$ ): the set of leaf clauses of a resolution proof of  $\mathcal{T}$ -unsatisfiability of  $\varphi$ , minus the  $\mathcal{T}$ -lemmas

# The proof-based approach to $\mathcal{T}$ -unsat cores: example

$$(x = 0 \vee \neg(x = 1) \vee A_1) \wedge (x = 0 \vee x = 1 \vee A_2) \wedge (\neg(x = 0) \vee x = 1 \vee A_2) \wedge$$

$$(\neg A_2 \vee y = 1) \wedge (\neg A_1 \vee x + y > 3) \wedge (y < 0) \wedge (A_2 \vee x - y = 4) \wedge (y = 2 \vee \neg A_1) \wedge (x \geq 0),$$





# The Assumption-based approach to $\mathcal{T}$ -unsat cores

Idea (adapted from [52])

Let  $\varphi$  be  $\bigwedge_{i=1}^n C_i$  s.t.  $\varphi$  unsatisfiable.

- 1 each clause  $C_i$  in  $\varphi$  is substituted by  $\neg S_i \vee C_i$ , s.t.  $S_i$  fresh “selector” variable
- 2 the resulting formula is checked for **satisfiability under the assumption of all  $S_i$ 's**
- 3 final conflict clause at dec. level 0:  $\bigvee_j \neg S_j$   
 $\implies \{C_j\}_j$  is the unsat core

Extends straightforwardly to  $\text{SMT}(\mathcal{T})$ .

# The assumption-based approach to $\mathcal{T}$ -unsat cores: Example

$$\begin{aligned} & (\mathcal{S}_1 \rightarrow (x = 0 \vee \neg(x = 1) \vee A_1)) \wedge (\mathcal{S}_2 \rightarrow (x = 0 \vee x = 1 \vee A_2)) \wedge \\ & (\mathcal{S}_3 \rightarrow (\neg(x = 0) \vee x = 1 \vee A_2)) \wedge (\mathcal{S}_4 \rightarrow (\neg A_2 \vee y = 1)) \wedge \\ & (\mathcal{S}_5 \rightarrow (\neg A_1 \vee x + y > 3)) \wedge (\mathcal{S}_6 \rightarrow y < 0) \wedge \\ & (\mathcal{S}_7 \rightarrow (A_2 \vee x - y = 4)) \wedge (\mathcal{S}_8 \rightarrow (y = 2 \vee \neg A_1)) \wedge (\mathcal{S}_9 \rightarrow x \geq 0) \end{aligned}$$

Conflict analysis (Yices 1.0.6) returns:

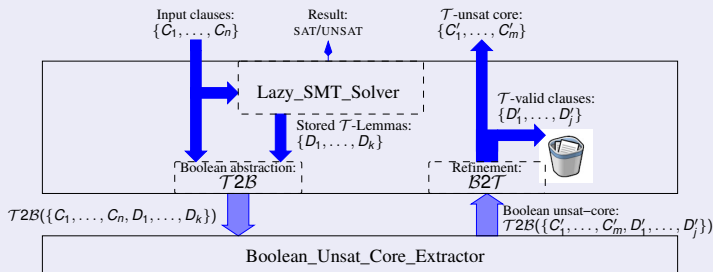
$$\neg \mathcal{S}_1 \vee \neg \mathcal{S}_2 \vee \neg \mathcal{S}_3 \vee \neg \mathcal{S}_4 \vee \neg \mathcal{S}_6 \vee \neg \mathcal{S}_7 \vee \neg \mathcal{S}_8,$$

corresponding to the unsat core in red.

# The lemma-lifting approach to $\mathcal{T}$ -unsat cores

Idea [26, 30]

- (i) The  $\mathcal{T}$ -lemmas  $D_i$  are valid in  $\mathcal{T}$
- (ii) The conjunction of  $\varphi$  with all the  $\mathcal{T}$ -lemmas  $D_1, \dots, D_k$  is propositionally unsatisfiable:  $\mathcal{T}2\mathcal{B}(\varphi \wedge \bigwedge_{i=1}^n D_i) \models \perp$ .



- interfaces with an external Boolean Unsat-core Extractor

⇒ benefits for free of all state-of-the-art size-reduction techniques

## The lemma-lifting approach to $\mathcal{T}$ -unsat cores (cont.)

```
<SatValue, Clause_set>  $\mathcal{T}$ -Unsat_Core(Clause_set  $\varphi$ ) {  
  //  $\varphi$  is  $\{C_1, \dots, C_n\}$   
  if (Lazy_SMT_Solver( $\varphi$ ) == SAT)  
    then return  $\langle \text{SAT}, \emptyset \rangle$ ;  
  //  $D_1, \dots, D_k$  are the  $\mathcal{T}$ -lemmas stored by Lazy_SMT_Solver  
   $\psi^P = \text{Boolean\_Core\_Extractor}(\mathcal{T}2\mathcal{B}(\{C_1, \dots, C_n, D_1, \dots, D_k\}));$   
  //  $\psi^P$  is  $\mathcal{T}2\mathcal{B}(\{C'_1, \dots, C'_m, D'_1, \dots, D'_j\});$   
  return  $\langle \text{UNSAT}, \{C'_1, \dots, C'_m\} \rangle$ ;  
}
```

# The lemma-lifting approach to $\mathcal{T}$ -unsat cores: example

$$(x = 0 \vee \neg(x = 1) \vee A_1) \wedge (x = 0 \vee x = 1 \vee A_2) \wedge (\neg(x = 0) \vee x = 1 \vee A_2) \wedge \\ (\neg A_2 \vee y = 1) \wedge (\neg A_1 \vee x + y > 3) \wedge (y < 0) \wedge (A_2 \vee x - y = 4) \wedge (y = 2 \vee \neg A_1) \wedge (x \geq 0),$$

- 1 The SMT solver generates the following set of  $\mathcal{LIA}$ -lemmas:

$$\{(\neg(x = 1) \vee \neg(x = 0)), (\neg(y = 2) \vee \neg(y < 0)), (\neg(y = 1) \vee \neg(y < 0))\}.$$

- 2 The following formula is passed to the external Boolean core extractor

$$(B_0 \vee \neg B_1 \vee A_1) \wedge (B_0 \vee B_1 \vee A_2) \wedge (\neg B_0 \vee B_1 \vee A_2) \wedge \\ (\neg A_2 \vee B_2) \wedge (\neg A_1 \vee B_3) \wedge B_4 \wedge (A_2 \vee B_5) \wedge (B_6 \vee \neg A_1) \wedge B_7 \wedge \\ (\neg B_1 \vee \neg B_0) \wedge (\neg B_6 \vee \neg B_4) \wedge (\neg B_2 \vee \neg B_4)$$

which returns the unsat core in red.

- 3 The unsat-core is mapped back, the three  $\mathcal{T}$ -lemmas are removed  
 $\implies$  the final  $\mathcal{T}$ -unsat core (in red above).

## Exercise

Consider the following set of clauses  $\varphi$  in  $\mathcal{EUF}$ .

$$\left\{ \begin{array}{l} (\neg(x = y) \vee (f(x) = f(y))), \\ (\neg(x = y) \vee \neg(f(x) = f(y))), \\ ((x = y) \vee (f(x) = f(y))), \\ ((x = y) \vee \neg(f(x) = f(y))) \end{array} \right\}$$

Find a minimal  $\mathcal{EUF}$ -unsatisfiable core.

# Outline

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 **Beyond Solving: Advanced SMT Functionalities**
  - Proofs and Unsatisfiable Cores
  - **Interpolants**
  - All-SMT & Predicate Abstraction (hints)
  - SMT with Optimization (Optimization Modulo Theories)

# Computing (Craig) Interpolants in SMT

## Craig Interpolant

Given an ordered pair  $(A, B)$  of formulas such that  $A \wedge B \models_{\mathcal{T}} \perp$ , a *Craig interpolant* is a formula  $I$  s.t.:

- a)  $A \models_{\mathcal{T}} I$ ,
- b)  $I \wedge B \models_{\mathcal{T}} \perp$ ,
- c)  $I \preceq A$  and  $I \preceq B$ .

“ $I \preceq A$ ” meaning that all non-interpreted (in  $\mathcal{T}$ ) symbols in  $I$  occur in  $A$  (including variables)

- Important in some FV applications
- A few works presented for various theories:
  - *EFUF* [54, 63], *DL* [27, 29], *UTVPI* [28, 29], *LRA* [54, 63, 27, 29], *LIA* [48, 17, 45], *BV* [49], ...



# A General Algorithm

Algorithm: Interpolant generation for  $\text{SMT}(\mathcal{T})$  [61, 54]

- (i) Generate a resolution proof of  $\mathcal{T}$ -unsatisfiability  $\mathcal{P}$  for  $A \wedge B$ .
  - (ii) ...
  - (iii) Foreach  $\mathcal{T}$ -lemma  $\neg\eta$  in  $\mathcal{P}$ , generate an interpolant  $I_\eta$  for  $(\eta \setminus B, \eta \downarrow B)$ .
  - (iv) For every original leaf clause  $C$  in  $\mathcal{P}$ , set  $I_C \stackrel{\text{def}}{=} C \downarrow B$  if  $C \in A$ , and  $I_C \stackrel{\text{def}}{=} \top$  if  $C \in B$ .
  - (v) For every inner node  $C$  of  $\mathcal{P}$  obtained by resolution from  $C_1 \stackrel{\text{def}}{=} p \vee \phi_1$  and  $C_2 \stackrel{\text{def}}{=} \neg p \vee \phi_2$ , set  $I_C \stackrel{\text{def}}{=} I_{C_1} \vee I_{C_2}$  if  $p$  does not occur in  $B$ , and  $I_C \stackrel{\text{def}}{=} I_{C_1} \wedge I_{C_2}$  otherwise.
  - (vi) Output  $I_\perp$  as an interpolant for  $(A, B)$ .
- “ $\eta \setminus B$ ” [resp. “ $\eta \downarrow B$ ”] is the set of literals in  $\eta$  whose atoms do not [resp. do] occur in  $B$ .

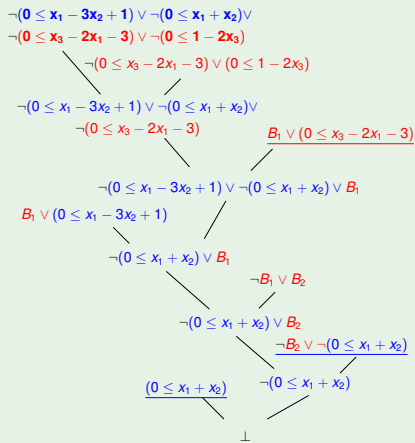
• row 2. only takes place where  $\mathcal{T}$  comes in to play

⇒ Reduced to the problem of finding an interpolant for two sets of  $\mathcal{T}$ -literals (Boolean and  $\mathcal{T}$ -specific component decoupled)

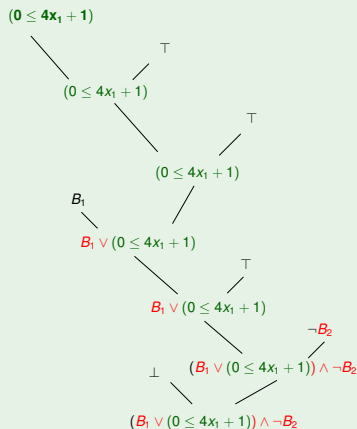
# Computing Craig Interpolants in SMT: example

$$A \stackrel{\text{def}}{=} (B_1 \vee (0 \leq x_1 - 3x_2 + 1)) \wedge (0 \leq x_1 + x_2) \wedge (\neg B_2 \vee \neg(0 \leq x_1 + x_2))$$

$$B \stackrel{\text{def}}{=} (\neg(0 \leq x_3 - 2x_1 - 3) \vee (0 \leq 1 - 2x_3)) \wedge (\neg B_1 \vee B_2) \wedge (B_1 \vee (0 \leq x_3 - 2x_1 - 3))$$



original proof



interpolant proof

# McMillan's algorithm for non-strict $\mathcal{LRA}$ inequalities

$$A \stackrel{\text{def}}{=} \{(0 \leq x_1 - 3x_2 + 1), (0 \leq x_1 + x_2)\}$$

$$B \stackrel{\text{def}}{=} \{(0 \leq x_3 - 2x_1 - 3), (0 \leq 1 - 2x_3)\}.$$

A proof of unsatisfiability  $P$  for  $A \wedge B$  is the following:

$$\frac{\frac{(0 \leq x_1 - 3x_2 + 1) \quad (0 \leq x_1 + x_2)}{\text{COMB } (0 \leq 4x_1 + 1) \text{ with c. 1 and 3}} \quad \frac{(0 \leq x_3 - 2x_1 - 3) \quad (0 \leq 1 - 2x_3)}{\text{COMB } (0 \leq -4x_1 - 5) \text{ with c. 2 and 1}}}{\text{COMB } (0 \leq -4) \text{ with c. 1 and 1}}$$

By replacing inequalities in  $B$  with  $(0 \leq 0)$ , we obtain the proof  $P'$ :

$$\frac{\frac{(0 \leq x_1 - 3x_2 + 1) \quad (0 \leq x_1 + x_2)}{\text{COMB } (0 \leq 4x_1 + 1)} \quad \frac{(0 \leq 0) \quad (0 \leq 0)}{\text{COMB } (0 \leq 0)}}{\text{COMB } (0 \leq 4x_1 + 1)}$$

Thus, the interpolant obtained is  $(0 \leq 4x_1 + 1)$ .

# Example: Interpolation Algorithms for Difference Logic

An inference-based algorithm [54]

$$A \stackrel{\text{def}}{=} \{(0 \leq x_1 - x_2 + 1), (0 \leq x_2 - x_3), (0 \leq x_4 - x_5 - 1)\}$$

$$B \stackrel{\text{def}}{=} \{(0 \leq x_5 - x_1), (0 \leq x_3 - x_4 - 1)\}.$$

$$\frac{\frac{(0 \leq x_1 - x_2 + 1) \quad (0 \leq x_2 - x_3)}{\text{COMB} \quad (0 \leq x_1 - x_3 + 1)} \quad (0 \leq x_4 - x_5 - 1)}{\text{COMB} \quad (0 \leq x_1 - x_3 + x_4 - x_5)} \quad (0 \leq x_5 - x_1)}{\text{COMB} \quad (0 \leq -x_3 + x_4)} \quad (0 \leq x_3 - x_4 - 1)$$

$$\frac{\frac{(0 \leq x_1 - x_2 + 1) \quad (0 \leq x_2 - x_3)}{\text{COMB} \quad (0 \leq x_1 - x_3 + 1)} \quad (0 \leq x_4 - x_5 - 1)}{\text{COMB} \quad (0 \leq x_1 - x_3 + x_4 - x_5)} \quad (0 \leq 0)}{\text{COMB} \quad (0 \leq x_1 - x_3 + x_4 - x_5)} \quad (0 \leq 0)$$

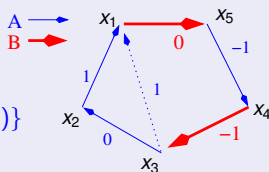
$\Rightarrow$  Interpolant:  $(0 \leq x_1 - x_3 + x_4 - x_5)$  (not in  $\mathcal{DL}$ , and weaker).

# Example: Interpolation Algorithms for Difference Logic

A graph-based algorithm [27, 29]

$$A \stackrel{\text{def}}{=} \overbrace{\{(0 \leq x_1 - x_2 + 1), (0 \leq x_2 - x_3), (0 \leq x_4 - x_5 - 1)\}}^{\text{Chord: } (0 \leq x_1 - x_3 + 1)}$$

$$B \stackrel{\text{def}}{=} \{(0 \leq x_5 - x_1), (0 \leq x_3 - x_4 - 1)\}.$$



$\implies$  Interpolant:  $(0 \leq x_1 - x_3 + 1) \wedge (0 \leq x_4 - x_5 - 1)$  (still in  $\mathcal{DL}$ )

## Exercise

Consider the following formulas in difference logic ( $\mathcal{DL}$ ):

$$\begin{aligned}\varphi_1 \stackrel{\text{def}}{=} & (x_2 - x_3 \leq -4) \wedge \\ & (x_3 - x_4 \leq -6) \wedge \\ & (x_5 - x_6 \leq 4) \wedge \\ & (x_6 - x_1 \leq 2) \wedge \\ & (x_6 - x_7 \leq -2) \wedge \\ & (x_7 - x_8 \leq 1)\end{aligned}$$

$$\begin{aligned}\varphi_2 \stackrel{\text{def}}{=} & (x_4 - x_9 \leq 2) \wedge \\ & (x_9 - x_5 \leq 0) \wedge \\ & (x_1 - x_2 \leq 1)\end{aligned}$$

which are such that  $\varphi_1 \wedge \varphi_2 \models_{\mathcal{DL}} \perp$ . Compute an interpolant for  $\langle \varphi_1, \varphi_2 \rangle$ , using both methods presented in previous slides.

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 **Beyond Solving: Advanced SMT Functionalities**
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - **All-SMT & Predicate Abstraction (hints)**
  - SMT with Optimization (Optimization Modulo Theories)

## All-SAT/All-SMT (hints)

- **All-SAT**: enumerate all truth assignments satisfying  $\varphi$
- **All-SMT**: enumerate all  $\mathcal{T}$ -satisfiable truth assignments propositionally satisfying  $\varphi$
- **All-SMT over an “important” subset of atoms  $\Gamma \stackrel{\text{def}}{=} \{\gamma_i\}_i$** :  
enumerate all assignments over  $\Gamma$  which can be extended to  $\mathcal{T}$ -satisfiable truth assignments propositionally satisfying  $\varphi$   
 $\implies$  can compute **predicate abstraction**
- **Algorithms**:
  - **BCLT** [50]  
each time a  $\mathcal{T}$ -satisfiable assignment  $\{l_1, \dots, l_n\}$  is found, perform conflict-driven backjumping as if the restricted clause  $(\bigvee_i \neg l_i) \downarrow \Gamma$  belonged to the clause set
  - **MathSAT/NuSMV** [25]  
As above, plus the Boolean search of the SMT solver is driven by an OBDD.



# Predicate Abstraction

## Predicate abstraction

if  $\varphi(\mathbf{v})$  is a SMT formula over the domain variables  $\mathbf{v} \stackrel{\text{def}}{=} \{v_j\}_j$ ,  $\{\gamma_i\}_i$  is a set of “relevant” predicates over  $\mathbf{v}$ , and  $\mathbf{P} \stackrel{\text{def}}{=} \{P_i\}_i$  a set of fresh Boolean labels, then:

$$\begin{aligned} & \text{PredAbs}_{\mathbf{P}}(\varphi) \\ \stackrel{\text{def}}{=} & \exists \mathbf{v}. ( \varphi(\mathbf{v}) \wedge \bigwedge_i P_i \leftrightarrow \gamma_i(\mathbf{v}) ) \\ = & \bigvee \left\{ \mu \mid \begin{array}{l} \mu \text{ truth assignment on } \mathbf{P} \\ \text{s.t. } \mu \wedge \varphi \wedge \bigwedge_i (P_i \leftrightarrow \gamma_i) \text{ is } \mathcal{T}\text{-satisfiable} \end{array} \right\} \end{aligned}$$

- projection of  $\varphi$  over (the Boolean abstraction of) the set  $\{\gamma_i\}_i$ .
- important step in FV: extracts finite-state abstractions from a infinite state space

## Predicate Abstraction: example

$$\varphi \stackrel{\text{def}}{=} (v_1 + v_2 > 12)$$

$$\gamma_1 \stackrel{\text{def}}{=} (v_1 + v_2 = 2)$$

$$\gamma_2 \stackrel{\text{def}}{=} (v_1 - v_2 < 10)$$

↓

$$\begin{aligned} \text{PreAbs}(\varphi)_{\{P_1, P_2\}} &\stackrel{\text{def}}{=} \exists v_1 v_2 . \left( \begin{array}{l} (v_1 + v_2 > 12) \quad \wedge \\ (P_1 \leftrightarrow (v_1 + v_2 = 2)) \quad \wedge \\ (P_2 \leftrightarrow (v_1 - v_2 < 10)) \end{array} \right) \\ &= (\neg P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2) \\ &= \neg P_1. \end{aligned}$$

- 1 Introduction
  - What is a Theory?
  - Satisfiability Modulo Theories
  - Motivations and Goals of SMT
- 2 Efficient SMT solving
  - Combining SAT with Theory Solvers
  - Theory Solvers for Theories of Interest (hints)
  - SMT for Combinations of Theories
- 3 **Beyond Solving: Advanced SMT Functionalities**
  - Proofs and Unsatisfiable Cores
  - Interpolants
  - All-SMT & Predicate Abstraction (hints)
  - **SMT with Optimization (Optimization Modulo Theories)**

# Optimization Modulo Theories: General Case

Ingredients:  $\langle \varphi, cost \rangle$

- a **SMT formula**  $\varphi$  in some background theory  $\mathcal{T} = \mathcal{T}_{\preceq} \cup \bigcup_i \mathcal{T}_i$ 
  - $\bigcup_i \mathcal{T}_i$  may be empty
  - $\mathcal{T}_{\preceq}$  has a predicate  $\preceq$  representing a **total order**
- a  $\mathcal{T}_{\preceq}$ -**variable/term** “*cost*” occurring in  $\varphi$

Optimization Modulo  $\mathcal{T}_{\preceq} \cup \bigcup_i \mathcal{T}_i$  ( $OMT(\mathcal{T}_{\preceq} \cup \bigcup_i \mathcal{T}_i)$ )

The problem of finding a model  $\mathcal{M}$  for  $\varphi$  whose value of *cost* is minimum according to  $\preceq$ .

- maximization is dual

## Note

The cost term can be rewritten as a variable

$$\langle \varphi, term \rangle \implies \langle \varphi \wedge (cost = term), cost \rangle, \quad cost \text{ fresh}$$

# Optimization Modulo Theories with $\mathcal{L}\mathcal{A}$ $\mathcal{L}\mathcal{R}\mathcal{A}$ costs

## Ingredients

- an **SMT formula**  $\varphi$  on  $\mathcal{L}\mathcal{A} \cup \mathcal{T}$   $\mathcal{L}\mathcal{R}\mathcal{A} \cup \mathcal{T}$ 
  - $\mathcal{L}\mathcal{A}$  can be  $\mathcal{L}\mathcal{R}\mathcal{A}$ ,  $\mathcal{L}\mathcal{I}\mathcal{A}$  or a combination of both
  - $\mathcal{T} \stackrel{\text{def}}{=} \bigcup_i \mathcal{T}_i$ , possibly empty
  - $\mathcal{L}\mathcal{A}\mathcal{L}\mathcal{R}\mathcal{A}$  and  $\mathcal{T}_i$  Nelson-Oppen theories (i.e. signature-disjoint infinite-domain theories)
- a  $\mathcal{L}\mathcal{A}$   $\mathcal{L}\mathcal{R}\mathcal{A}$  variable [term] “**cost**” occurring in  $\varphi$
- (optionally) two constant numbers lb (**lower bound**) and ub (**upper bound**) s.t.  $\text{lb} \leq \text{cost} < \text{ub}$  (lb, ub may be  $\mp\infty$ )

## Optimization Modulo Theories with $\mathcal{L}\mathcal{A}$ $\mathcal{L}\mathcal{R}\mathcal{A}$ costs (OMT( $\mathcal{L}\mathcal{A} \cup \mathcal{T}$ ) OMT( $\mathcal{L}\mathcal{R}\mathcal{A} \cup \mathcal{T}$ ))

Find a model for  $\varphi$  whose value of **cost** is minimum.

- maximization dual

We first restrict to the case  $\mathcal{L}\mathcal{A} = \mathcal{L}\mathcal{R}\mathcal{A}$  and  $\bigcup_i \mathcal{T}_i = \{\}$  (OMT( $\mathcal{L}\mathcal{R}\mathcal{A}$ )).

# Solving OMT( $\mathcal{LRA}$ ) [65, 66]

## General idea

Combine standard SMT and LP minimization techniques.

## Offline Schema

- Minimizer: based on the Simplex  $\mathcal{LRA}$ -solver by [37]
  - Handles strict inequalities
- Search Strategies:
  - Linear-Search strategy
  - Mixed Linear/Binary strategy

# A toy example (linear search)

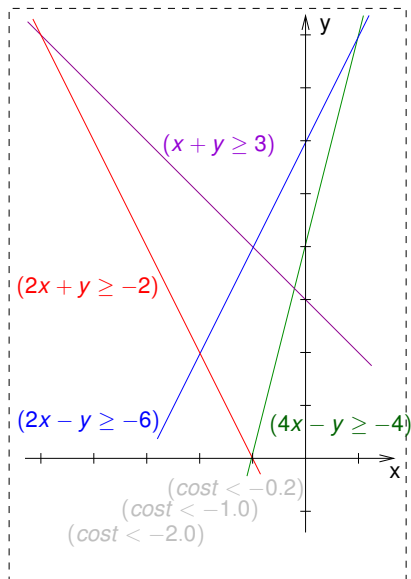
[w. pure-literal filt.  $\implies$  partial assignments]

- OMT( $\mathcal{LRA}$ ) problem:

$$\begin{aligned} \varphi &\stackrel{\text{def}}{=} (\neg A_1 \vee (2x + y \geq -2)) \\ &\wedge (A_1 \vee (x + y \geq 3)) \\ &\wedge (\neg A_2 \vee (4x - y \geq -4)) \\ &\wedge (A_2 \vee (2x - y \geq -6)) \\ &\wedge (\text{cost} < -0.2) \\ &\wedge (\text{cost} < -1.0) \\ &\wedge (\text{cost} < -2.0) \end{aligned}$$

$$\text{cost} \stackrel{\text{def}}{=} x$$

- $\mu = \left\{ \begin{array}{l} A_1, \neg A_1, A_2, \neg A_2, \\ (4x - y \geq -4), \\ (x + y \geq 3), \\ (2x + y \geq -2), \\ (2x - y \geq -6) \\ (\text{cost} < -0.2) \\ (\text{cost} < -1.0) \\ (\text{cost} < -2.0) \end{array} \right\}$



# A toy example (linear search)

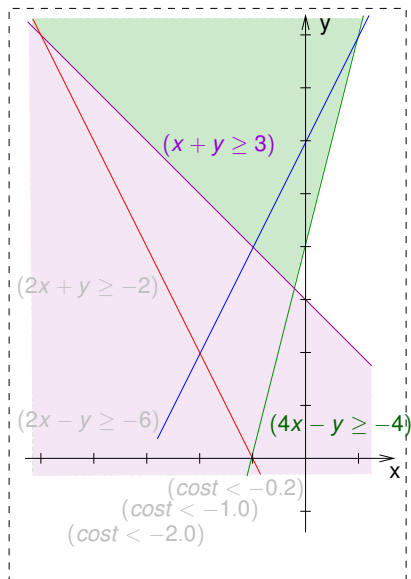
[w. pure-literal filt.  $\implies$  partial assignments]

- OMT( $\mathcal{LRA}$ ) problem:

$$\begin{aligned}\varphi &\stackrel{\text{def}}{=} (\neg A_1 \vee (2x + y \geq -2)) \\ &\wedge (A_1 \vee (x + y \geq 3)) \\ &\wedge (\neg A_2 \vee (4x - y \geq -4)) \\ &\wedge (A_2 \vee (2x - y \geq -6)) \\ &\wedge (\text{cost} < -0.2) \\ &\wedge (\text{cost} < -1.0) \\ &\wedge (\text{cost} < -2.0)\end{aligned}$$

$$\text{cost} \stackrel{\text{def}}{=} x$$

- $\mu = \left\{ \begin{array}{l} A_1, \neg A_1, A_2, \neg A_2, \\ (4x - y \geq -4), \\ (x + y \geq 3), \\ (2x + y \geq -2), \\ (2x - y \geq -6) \\ (\text{cost} < -0.2) \\ (\text{cost} < -1.0) \\ (\text{cost} < -2.0) \end{array} \right\}$   
 $\implies \text{SAT}, \min = -0.2$





# A toy example (linear search)

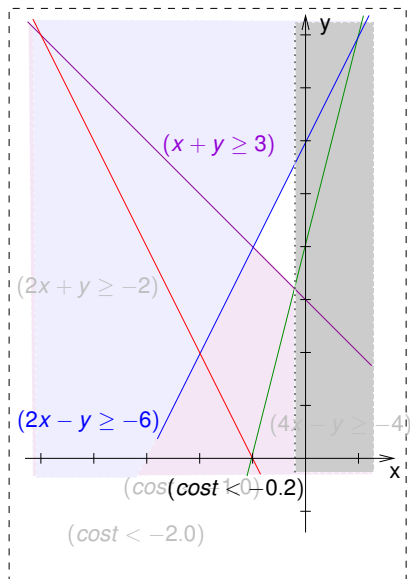
[w. pure-literal filt.  $\implies$  partial assignments]

- OMT( $\mathcal{LRA}$ ) problem:

$$\begin{aligned}\varphi &\stackrel{\text{def}}{=} (\neg A_1 \vee (2x + y \geq -2)) \\ &\wedge (A_1 \vee (x + y \geq 3)) \\ &\wedge (\neg A_2 \vee (4x - y \geq -4)) \\ &\wedge (A_2 \vee (2x - y \geq -6)) \\ &\wedge (\text{cost} < -0.2) \\ &\wedge (\text{cost} < -1.0) \\ &\wedge (\text{cost} < -2.0)\end{aligned}$$

$$\text{cost} \stackrel{\text{def}}{=} x$$

- $\mu = \left\{ \begin{array}{l} A_1, \neg A_1, A_2, \neg A_2, \\ (4x - y \geq -4), \\ (x + y \geq 3), \\ (2x + y \geq -2), \\ (2x - y \geq -6) \\ (\text{cost} < -0.2) \\ (\text{cost} < -1.0) \\ (\text{cost} < -2.0) \end{array} \right\}$   
 $\implies$  SAT,  $\min = -1.0$



# A toy example (linear search)

[w. pure-literal filt.  $\implies$  partial assignments]

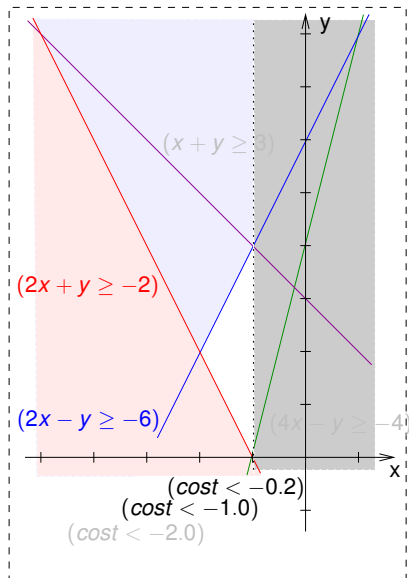
- OMT( $\mathcal{LRA}$ ) problem:

$$\begin{aligned} \varphi &\stackrel{\text{def}}{=} (\neg A_1 \vee (2x + y \geq -2)) \\ &\wedge (A_1 \vee (x + y \geq 3)) \\ &\wedge (\neg A_2 \vee (4x - y \geq -4)) \\ &\wedge (A_2 \vee (2x - y \geq -6)) \\ &\wedge (\text{cost} < -0.2) \\ &\wedge (\text{cost} < -1.0) \\ &\wedge (\text{cost} < -2.0) \end{aligned}$$

$$\text{cost} \stackrel{\text{def}}{=} x$$

$$\bullet \mu = \left\{ \begin{array}{l} A_1, \neg A_1, A_2, \neg A_2, \\ (4x - y \geq -4), \\ (x + y \geq 3), \\ (2x + y \geq -2), \\ (2x - y \geq -6) \\ (\text{cost} < -0.2) \\ (\text{cost} < -1.0) \\ (\text{cost} < -2.0) \end{array} \right\}$$

$\implies$  SAT,  $\min = -2.0$



# A toy example (linear search)

[w. pure-literal filt.  $\implies$  partial assignments]

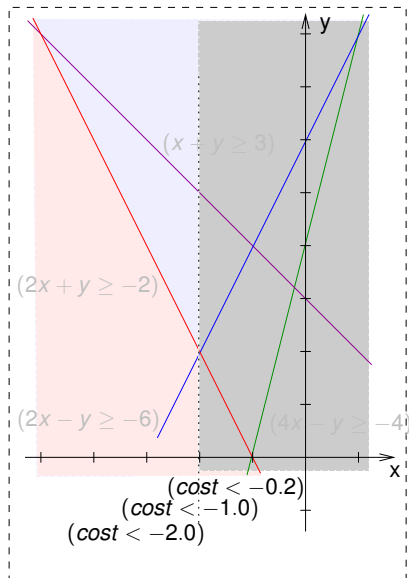
- OMT( $\mathcal{LRA}$ ) problem:

$$\begin{aligned} \varphi &\stackrel{\text{def}}{=} (\neg A_1 \vee (2x + y \geq -2)) \\ &\wedge (A_1 \vee (x + y \geq 3)) \\ &\wedge (\neg A_2 \vee (4x - y \geq -4)) \\ &\wedge (A_2 \vee (2x - y \geq -6)) \\ &\wedge (\text{cost} < -0.2) \\ &\wedge (\text{cost} < -1.0) \\ &\wedge (\text{cost} < -2.0) \end{aligned}$$

$$\text{cost} \stackrel{\text{def}}{=} x$$

- $\mu = \left\{ \begin{array}{l} A_1, \neg A_1, A_2, \neg A_2, \\ (4x - y \geq -4), \\ (x + y \geq 3), \\ (2x + y \geq -2), \\ (2x - y \geq -6) \\ (\text{cost} < -0.2) \\ (\text{cost} < -1.0) \\ (\text{cost} < -2.0) \end{array} \right\}$

$\implies$  UNSAT,  $\min = -2.0$

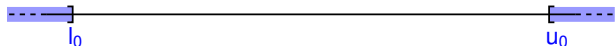


# Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:**  $\langle \varphi, cost, lb, ub \rangle$  //  $lb$  can be  $-\infty$ ,  $ub$  can be  $+\infty$

$l \leftarrow lb; u \leftarrow ub; \mathcal{M} \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{-(cost < lb), (cost < ub)\};$

**while** ( $l < u$ ) **do**

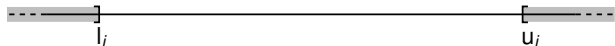


# Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:**  $\langle \varphi, cost, lb, ub \rangle$  //  $lb$  can be  $-\infty$ ,  $ub$  can be  $+\infty$   
 $l \leftarrow lb$ ;  $u \leftarrow ub$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  $\varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\}$ ;  
**while** ( $l < u$ ) **do**

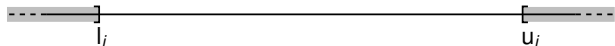
**if** (BinSearchMode()) **then** // Binary-search Mode

**else** // Linear-search Mode



# Offline Schema: Mixed Linear/Binary-Search Strategy

```
Input:  $\langle \varphi, cost, lb, ub \rangle$  // lb can be  $-\infty$ , ub can be  $+\infty$   
 $l \leftarrow lb; u \leftarrow ub; \mathcal{M} \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\};$   
while ( $l < u$ ) do  
  if (BinSearchMode()) then // Binary-search Mode  
  else // Linear-search Mode  
     $\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi);$ 
```



# Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:**  $\langle \varphi, cost, lb, ub \rangle$  //  $lb$  can be  $-\infty$ ,  $ub$  can be  $+\infty$   
 $l \leftarrow lb$ ;  $u \leftarrow ub$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  $\varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\}$ ;

**while** ( $l < u$ ) **do**

**if** (BinSearchMode()) **then** // Binary-search Mode

**else** // Linear-search Mode

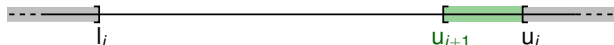
$\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi)$ ;

**if** ( $res = \text{SAT}$ ) **then**

$\langle \mathcal{M}, u \rangle \leftarrow \text{LRA-Solver.Minimize}(cost, \mu)$ ;

$\varphi \leftarrow \varphi \cup \{(cost < u)\}$ ;

**else**  $\{res = \text{UNSAT}\}$



# Offline Schema: Mixed Linear/Binary-Search Strategy

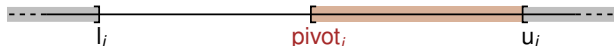
```
Input:  $\langle \varphi, cost, lb, ub \rangle$  //  $lb$  can be  $-\infty$ ,  $ub$  can be  $+\infty$   
 $l \leftarrow lb$ ;  $u \leftarrow ub$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  $\varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\}$ ;  
while ( $l < u$ ) do  
  if (BinSearchMode()) then // Binary-search Mode  
  else // Linear-search Mode  
     $\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi)$ ;  
    if ( $res = \text{SAT}$ ) then  
      else { $res = \text{UNSAT}$ }  
         $l \leftarrow u$ ;  
return  $\langle \mathcal{M}, u \rangle$ 
```





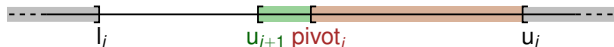
# Offline Schema: Mixed Linear/Binary-Search Strategy

```
Input:  $\langle \varphi, cost, lb, ub \rangle$  // lb can be  $-\infty$ , ub can be  $+\infty$   
 $l \leftarrow lb; u \leftarrow ub; \mathcal{M} \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\};$   
while ( $l < u$ ) do  
  if (BinSearchMode()) then // Binary-search Mode  
     $pivot \leftarrow \text{ComputePivot}(l, u);$   
     $\varphi \leftarrow \varphi \cup \{(cost < pivot)\};$   
     $\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi);$   
  else // Linear-search Mode  
    [
```



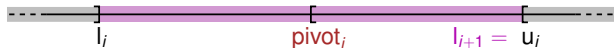
# Offline Schema: Mixed Linear/Binary-Search Strategy

```
Input:  $\langle \varphi, cost, lb, ub \rangle$  //  $lb$  can be  $-\infty$ ,  $ub$  can be  $+\infty$   
 $l \leftarrow lb$ ;  $u \leftarrow ub$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  $\varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\}$ ;  
while ( $l < u$ ) do  
  if (BinSearchMode()) then // Binary-search Mode  
     $pivot \leftarrow \text{ComputePivot}(l, u)$ ;  
     $\varphi \leftarrow \varphi \cup \{(cost < pivot)\}$ ;  
     $\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi)$ ;  
  else // Linear-search Mode  
    if ( $res = \text{SAT}$ ) then  
       $\langle \mathcal{M}, u \rangle \leftarrow \text{LRA-Solver.Minimize}(cost, \mu)$ ;  
       $\varphi \leftarrow \varphi \cup \{(cost < u)\}$ ;  
    else  $\{res = \text{UNSAT}\}$ 
```



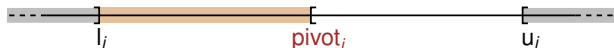
# Offline Schema: Mixed Linear/Binary-Search Strategy

```
Input:  $\langle \varphi, cost, lb, ub \rangle$  // lb can be  $-\infty$ , ub can be  $+\infty$   
 $l \leftarrow lb$ ;  $u \leftarrow ub$ ;  $\mathcal{M} \leftarrow \emptyset$ ;  $\varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\}$ ;  
while ( $l < u$ ) do  
  if (BinSearchMode()) then // Binary-search Mode  
     $pivot \leftarrow \text{ComputePivot}(l, u)$ ;  
     $\varphi \leftarrow \varphi \cup \{(cost < pivot)\}$ ;  
     $\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi)$ ;  
  else // Linear-search Mode  
    if ( $res = \text{SAT}$ ) then  
      return  $\langle \mathcal{M}, u \rangle$   
    else { $res = \text{UNSAT}$ }  
      if ( $(cost < pivot) \notin \text{SMT.ExtractUnsatCore}(\varphi)$ ) then  
         $l \leftarrow u$ ;  
      else  
        return  $\langle \mathcal{M}, u \rangle$ 
```



# Offline Schema: Mixed Linear/Binary-Search Strategy

```
Input:  $\langle \varphi, cost, lb, ub \rangle$  // lb can be  $-\infty$ , ub can be  $+\infty$   
 $l \leftarrow lb; u \leftarrow ub; \mathcal{M} \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(cost < lb), (cost < ub)\};$   
while ( $l < u$ ) do  
  if (BinSearchMode()) then // Binary-search Mode  
     $pivot \leftarrow \text{ComputePivot}(l, u);$   
     $\varphi \leftarrow \varphi \cup \{(cost < pivot)\};$   
     $\langle res, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi);$   
  else // Linear-search Mode  
    if ( $res = \text{SAT}$ ) then  
    else { $res = \text{UNSAT}$ }  
      if ( $(cost < pivot) \notin \text{SMT.ExtractUnsatCore}(\varphi)$ ) then  
        else  
           $l \leftarrow pivot;$   
           $\varphi \leftarrow (\varphi \setminus \{(cost < pivot)\}) \cup \{\neg(cost < pivot)\};$ 
```



# OMT with Lexicographic Combination of Objectives [12]

## The problem

Find one optimal model  $\mathcal{M}$  minimizing  $\underline{c} \stackrel{\text{def}}{=} cost_1, cost_2, \dots, cost_k$  lexicographically.

## Solution

- Intuition:

*{ minimize  $cost_1$  }*

*when UNSAT*

*{ substitute unit clause ( $cost_1 < min_1$ ) with ( $cost_1 = min_1$ ) }*

*{ minimize  $cost_2$  }*

...

- improvement:

- each time UNSAT is found, add  $\bigwedge_i (cost_i \leq \mathcal{M}_i(cost_i))$  to  $\varphi$

# Optimization problems encoded into OMT( $\mathcal{L}\mathcal{A} \cup \mathcal{T}$ ) I

## SMT with Pseudo-Boolean Constraints & Weighted MaxSMT

$$OMT + PB : \quad \sum_j w_j \cdot A_j, \quad w_i > 0 \quad // (\sum_j \text{ite}(A_j, w_j, 0))$$

$\Downarrow$

$$\begin{array}{l} \sum_j x_j, \quad x_j \text{ fresh} \\ \text{s.t.} \quad \dots \wedge \bigwedge_j (A_j \rightarrow (x_j = w_j)) \wedge (\neg A_j \rightarrow (x_j = 0)) \\ \quad \quad \quad \wedge (x_j \geq 0) \wedge (x_j \leq w_j) \end{array}$$

$$MaxSMT : \quad \langle \varphi_h, \bigwedge_j \psi_j \rangle \quad \text{s.t. } \psi_j \text{ soft}, \quad w_j = \text{weight}(\psi_j), \quad w_i > 0$$

$\Downarrow$

$$\begin{array}{l} \text{minimize } \sum_j x_j, \quad x_j, A_j \text{ fresh} \\ \varphi_h \wedge \bigwedge_j (A_j \vee \psi_j) \wedge \bigwedge_j (\neg A_j \vee (x_j = w_j)) \wedge (A_j \vee (x_j = 0)) \\ \quad \quad \quad \wedge (x_j \geq 0) \wedge (x_j \leq w_j) \end{array}$$

## Remark: range constraints “ $(x_j \geq 0) \wedge (x_j \leq w_j)$ ”

$$\begin{aligned} OMT + PB : \quad & \sum_j w_j \cdot A_j, w_i > 0 \quad //(\sum_j \text{ite}(A_j, w_j, 0)) \\ & \downarrow \\ & \sum_j x_j, x_j \text{ fresh} \\ \text{s.t.} \quad & \dots \wedge \bigwedge_j (A_j \rightarrow (x_j = w_j)) \wedge (\neg A_j \rightarrow (x_j = 0)) \\ & \wedge (x_j \geq 0) \wedge (x_j \leq w_j) \end{aligned}$$

Range constraints “ $(x_j \geq 0) \wedge (x_j \leq w_j)$ ” logically redundant, but essential for efficiency:

- Without range constraints, the SMT solver can detect the violation of a bound **only after all  $A_i$ 's are assigned** :  
Ex:  $w_1 = 4, w_2 = 7, \sum_{i=1} x_i < 10, A_1 = A_2 = \top, A_i = * \forall i > 2$ .
- With range constraints, the SMT solver detects the violation as soon as the assigned  $A_i$ 's violate a bound  
 $\implies$  drastic pruning of the search
- same for weighted MaxSMT

# Optimization problems encoded into $\text{OMT}(\mathcal{L}\mathcal{A} \cup \mathcal{T})$ II

## OMT with Min-Max [Max-Min] optimization

Given  $\langle \varphi, \{cost_1, \dots, cost_k\} \rangle$ , find a solution which minimizes the maximum value among  $\{cost_1, \dots, cost_k\}$ . (Max-Min dual.)

- Frequent in some applications (e.g. [66, 71])

$\Rightarrow$  encode into  $\text{OMT}(\mathcal{L}\mathcal{A} \cup \mathcal{T})$  problem  $\{\varphi \wedge \bigwedge_i (cost_i \leq cost), cost\}$   
s.t.  $cost$  fresh.

## OMT with linear combinations of costs

Given  $\langle \varphi, \{cost_1, \dots, cost_k\} \rangle$  and a set of weights  $\{w_1, \dots, w_k\}$ , find a solution which minimizes  $\sum_i w_i \cdot cost_i$ .

$\Rightarrow$  encode into  $\text{OMT}(\mathcal{L}\mathcal{A} \cup \mathcal{T})$  problem  
 $\{\varphi \wedge (cost = \sum_i w_i \cdot cost_i), cost\}$  s.t.  $cost$  fresh.

These objectives can be composed with other  $\text{OMT}(\mathcal{L}\mathcal{A})$  objectives.



- survey papers:
  - Roberto Sebastiani: "Lazy Satisfiability Modulo Theories". Journal on Satisfiability, Boolean Modeling and Computation, JSAT. Vol. 3, 2007. Pag 141–224, ©IOS Press.
  - Clark Barrett, Roberto Sebastiani, Sanjit Seshia, Cesare Tinelli "Satisfiability Modulo Theories". Part II, Chapter 26, The Handbook of Satisfiability. 2009. ©IOS press.
  - Leonardo de Moura and Nikolaj Bjørner. "Satisfiability modulo theories: introduction and applications". Communications of the ACM, 54 (9), 2011. ©ACM press.
- web links:
  - The SMT library SMT-LIB:  
<http://goedel.cs.uiowa.edu/smtlib/>
  - The SMT Competition SMT-COMP:  
<http://www.smtcomp.org/>
  - The SAT/SMT Schools  
<http://satassociation.org/sat-smt-school.html>

# References I

- [1] A. Armando.  
Simplifying OBDDs in Decidable Theories.  
In *Proc. PDPAR'03.*, 2003.
- [2] A. Armando, C. Castellini, and E. Giunchiglia.  
SAT-based procedures for temporal reasoning.  
In *Proc. European Conference on Planning, CP-99*, 1999.
- [3] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani.  
A SAT Based Approach for Solving Formulas over Boolean and Linear Mathematical Propositions.  
In *Proc. CADE'2002.*, volume 2392 of *LNAI*. Springer, July 2002.
- [4] G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani.  
Integrating Boolean and Mathematical Solving: Foundations, Basic Algorithms and Requirements.  
In *Proc. AIARSC'2002*, volume 2385 of *LNAI*. Springer, 2002.
- [5] G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani.  
Verifying Industrial Hybrid Systems with MathSAT.  
In *Proc. PDPAR'03*, 2003.
- [6] G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani.  
SAT-Based Bounded Model Checking for Timed Systems.  
In *Proc. FORTE'02.*, volume 2529 of *LNCS*. Springer, November 2002.
- [7] C. Barret, D. Dill, and A. Stump.  
A Generalization of Shostak's Method for Combining Decision Procedures.  
In *Proc. FRODOS'02*, 2002.
- [8] C. Barrett, D. Dill, and A. Stump.  
Checking Satisfiability of First-Order Formulas by Incremental Translation to SAT.  
In *14th International Conference on Computer-Aided Verification*, 2002.

# References II

- [9] C. Barrett, R. Nieuwenhuis, A. Oliveras, and C. Tinelli.  
Splitting on Demand in SAT Modulo Theories.  
In *Proc. LPAR'06*, volume 4246 of *LNAI*. Springer, 2006.
- [10] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli.  
Satisfiability Modulo Theories.  
In *Handbook of Satisfiability*, chapter 26, pages 825–885. IOS Press, 2009.
- [11] P. Baumgartner.  
FDPLL - A First Order Davis-Putnam-Longeman-Loveland Procedure.  
In *Proceedings of CADE-17*, pages 200–219. Springer-Verlag, 2000.
- [12] N. Bjorner and A.-D. Phan.  
 $\nu Z$  - Maximal Satisfaction with Z3.  
In *Proc International Symposium on Symbolic Computation in Software Science*, Gammart, Tunisia, December 2014. EasyChair Proceedings in Computing (EPIc).  
<http://www.easychair.org/publications/?page=862275542>.
- [13] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Ranise, and R. Sebastiani.  
Efficient Satisfiability Modulo Theories via Boolean Search.  
*Information and Computation*, 2005.
- [14] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Ranise, and R. Sebastiani.  
Efficient Satisfiability Modulo Theories via Delayed Theory Combination.  
In *Proc. CAV 2005*, volume 3576 of *LNCS*. Springer, 2005.
- [15] M. Bozzano, R. Bruttomesso, A. Cimatti, T. A. Junttila, S. Ranise, P. van Rossum, and R. Sebastiani.  
Efficient Theory Combination via Boolean Search.  
*Information and Computation*, 204(10):1493–1525, 2006.
- [16] M. Bozzano, R. Bruttomesso, A. Cimatti, T. A. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani.  
Mathsat: Tight integration of sat and mathematical decision procedures.  
*Journal of Automated Reasoning*, 35(1-3):265–293, 2005.

# References III

- [17] A. Brillout, D. Kroening, P. Rümmer, and T. Wahl.  
An Interpolating Sequent Calculus for Quantifier-Free Presburger Arithmetic.  
In *Proc. IJCAR*, volume 6173 of *LNCS*. Springer, 2010.
- [18] R. Brinkmann and R. Drechsler.  
RTL-datapath verification using integer linear programming.  
In *Proc. ASP-DAC 2002*, pages 741–746. IEEE, 2002.
- [19] R. Brummayer and A. Biere.  
Lemmas on Demand for the Extensional Theory of Arrays.  
*Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 6, 2009.
- [20] R. Brummayer and A. Biere.  
Boolector: An efficient smt solver for bit-vectors and arrays.  
In *TACAS*, volume 5505 of *LNCS*, pages 174–177. Springer, 2009.
- [21] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, Z. Hanna, A. Nadel, A. Palti, and R. Sebastiani.  
A Lazy and Layered SMT( $\mathcal{BV}$ ) Solver for Hard Industrial Verification Problems.  
In *CAV*, volume 4590 of *LNCS*, pages 547–560. Springer, 2007.
- [22] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani.  
Delayed Theory Combination vs. Nelson-Open for Satisfiability Modulo Theories: A Comparative Analysis.  
In *Proc. LPAR*, volume 4246 of *LNCS*. Springer, 2006.
- [23] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani.  
Delayed Theory Combination vs. Nelson-Open for Satisfiability Modulo Theories: A Comparative Analysis. .  
*Annals of Mathematics and Artificial Intelligence.*, 55(1-2), 2009.
- [24] J. R. Burch and D. L. Dill.  
Automatic Verification of Pipelined Microprocessor Control.  
In *Proc. CAV '94*, volume 818 of *LNCS*. Springer, 1994.

# References IV

- [25] R. Cavada, A. Cimatti, A. Franzén, K. Kalyanasundaram, M. Roveri, and R. K. Shyamasundar. Computing Predicate Abstractions by Integrating BDDs and SMT Solvers. In *FMCAD*, pages 69–76. IEEE Computer Society, 2007.
- [26] A. Cimatti, A. Griggio, and R. Sebastiani. A Simple and Flexible Way of Computing Small Unsatisfiable Cores in SAT Modulo Theories. In *SAT*, volume 4501 of *LNCS*, pages 334–339. Springer, 2007.
- [27] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Interpolant Generation in Satisfiability Modulo Theories. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08.*, volume 4963 of *LNCS*. Springer, 2008.
- [28] A. Cimatti, A. Griggio, and R. Sebastiani. Interpolant Generation for UTVPI. In *CADE*, volume 5663 of *LNCS*, pages 167–182, 2009.
- [29] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Generation of Craig Interpolants in Satisfiability Modulo Theories. *ACM Transaction on Computational Logics – TOCL*, 12(1), October 2010.
- [30] A. Cimatti, A. Griggio, and R. Sebastiani. Computing Small Unsatisfiable Cores in SAT Modulo Theories. *Journal of Artificial Intelligence Research, JAIR*, 40:701–728, April 2011.
- [31] S. Cotton and O. Maler. Fast and Flexible Difference Logic Propagation for DPLL(T). In *Proc. SAT'06*, volume 4121 of *LNCS*. Springer, 2006.
- [32] L. de Moura and N. Bjørner. Efficient E-matching for SMT solvers. In *Proc. CADE-21, 21st International Conference on Automated Deduction*, volume 4603 of *LNCS*. Springer, 2007.

# References V

- [33] L. de Moura, H. Ruess, and M. Sorea.  
Lazy Theorem Proving for Bounded Model Checking over Infinite Domains.  
In *Proc. CADE'2002.*, volume 2392 of *LNAI*. Springer, July 2002.
- [34] L. M. de Moura and N. Bjørner.  
Model-based theory combination.  
*Electr. Notes Theor. Comput. Sci.*, 198(2):37–49, 2008.
- [35] L. M. de Moura and N. Bjørner.  
Generalized, efficient array decision procedures.  
In *FMCAD*, pages 45–52. IEEE, 2009.
- [36] D. Detlefs, G. Nelson, and J. Saxe.  
Simplify: a theorem prover for program checking.  
*Journal of the ACM*, 52(3):365–473, 2005.
- [37] B. Dutertre and L. de Moura.  
A Fast Linear-Arithmetic Solver for DPLL(T).  
In *CAV*, volume 4144 of *LNCS*, 2006.
- [38] B. Dutertre and L. de Moura.  
System Description: Yices 1.0.  
In *Proc. on 2nd SMT competition, SMT-COMP'06*, 2006.  
Available at [yices.csl.sri.com/yices-smtcomp06.pdf](http://yices.csl.sri.com/yices-smtcomp06.pdf).
- [39] A. Franzen, A. Cimatti, A. Nadel, R. Sebastiani, and J. Shalev.  
Applying SMT in Symbolic Execution of Microcode.  
In *Proc. Int. Conference on Formal Methods in Computer Aided Design (FMCAD'10)*. IEEE, 2010.
- [40] V. Ganesh and D. L. Dill.  
A Decision Procedure for Bit-Vectors and Arrays.  
In *CAV*, 2007.

# References VI

- [41] H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli.  
DPLL(T): Fast decision procedures.  
In R. Alur and D. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification, CAV'04 (Boston, Massachusetts)*, LNCS. Springer, 2004.
- [42] F. Giunchiglia and R. Sebastiani.  
Building decision procedures for modal logics from propositional decision procedures - the case study of modal K.  
In *Proc. CADE'13, LNAI*, New Brunswick, NJ, USA, August 1996. Springer.
- [43] A. Goel, S. Krstić, and A. Fuchs.  
Deciding array formulas with frugal axiom instantiation.  
In *Proceedings of SMT'08/BPR'08*, pages 12–17, New York, NY, USA, 2008. ACM.
- [44] A. Griggio.  
A Practical Approach to SMT(LA(Z)).  
In *Proc. SMT 2010*, 2010.
- [45] A. Griggio, T. T. H. Le, and R. Sebastiani.  
Efficient Interpolant Generation in Satisfiability Modulo Linear Integer Arithmetic.  
In *Proc. Tools and Algorithms for the Construction and Analysis of Systems, TACAS'11*, LNCS. Springer, 2011.
- [46] A. Griggio, Q. S. Phan, R. Sebastiani, and S. Tomasi.  
Stochastic Local Search for SMT: Combining Theory Solvers with WalkSAT.  
In *Frontiers of Combining Systems, FroCoS'11*, volume 6989 of *LNAI*. Springer, 2011.
- [47] I. Horrocks and P. F. Patel-Schneider.  
FaCT and DLP.  
In *Proc. Tableaux'98*, pages 27–30, 1998.
- [48] H. Jain, E. M. Clarke, and O. Grumberg.  
Efficient Craig Interpolation for Linear Diophantine (Dis)Equations and Linear Modular Equations.  
In A. Gupta and S. Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 254–267. Springer, 2008.

# References VII

- [49] D. Kroening and G. Weissenbacher.  
Lifting Propositional Interpolants to the Word-Level.  
In *FMCAD*, pages 85–89, Los Alamitos, CA, USA, 2007. IEEE Computer Society.
- [50] S. K. Lahiri, R. Nieuwenhuis, and A. Oliveras.  
SMT techniques for fast predicate abstraction.  
In *Proc. CAV*, LNCS 4144. Springer, 2006.
- [51] M. Liffiton and K. Sakallah.  
Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints.  
*Journal of Automated Reasoning*, 40(1), 2008.
- [52] I. Lynce and J. P. Marques-Silva.  
On computing minimum unsatisfiable cores.  
In *SAT*, 2004.
- [53] M. Mahfoudh, P. Niebert, E. Asarin, and O. Maler.  
A Satisfiability Checker for Difference Logic.  
In *Proceedings of SAT-02*, pages 222–230, 2002.
- [54] K. L. McMillan.  
An interpolating theorem prover.  
*Theor. Comput. Sci.*, 345(1):101–121, 2005.
- [55] J. Moeller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard.  
Fully symbolic model checking of timed systems using difference decision diagrams.  
In *Proc. Workshop on Symbolic Model Checking (SMC), FLoC'99*, Trento, Italy, July 1999.
- [56] C. G. Nelson and D. C. Oppen.  
Simplification by cooperating decision procedures.  
*TOPLAS*, 1(2):245–257, 1979.



# References VIII

- [57] G. Nelson and D. Oppen.  
Simplification by Cooperating Decision Procedures.  
*ACM Trans. on Programming Languages and Systems*, 1(2):245–257, 1979.
- [58] G. Nelson and D. Oppen.  
Fast Decision Procedures Based on Congruence Closure.  
*Journal of the ACM*, 27(2):356–364, 1980.
- [59] R. Nieuwenhuis and A. Oliveras.  
Congruence closure with integer offsets.  
In *In 10th Int. Conf. Logic for Programming, Artif. Intell. and Reasoning (LPAR)*, volume 2850 of *LNAI*, pages 78–90. Springer, 2003.
- [60] R. Nieuwenhuis and A. Oliveras.  
DPLL(T) with Exhaustive Theory Propagation and its Application to Difference Logic.  
In *Proc. CAV'05*, volume 3576 of *LNCS*. Springer, 2005.
- [61] P. Pudlák.  
Lower bounds for resolution and cutting planes proofs and monotone computations.  
*J. of Symb. Logic*, 62(3), 1997.
- [62] H. Rueß and N. Shankar.  
Deconstructing Shostak.  
In *Proc. LICS '01*. IEEE Computer Society, 2001.
- [63] A. Rybalchenko and V. Sofronie-Stokkermans.  
Constraint Solving for Interpolation.  
In *Proc. VMCAI*, volume 4349 of *LNCS*. Springer, 2007.
- [64] R. Sebastiani.  
Lazy Satisfiability Modulo Theories.  
*Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, 3(3-4):141–224, 2007.

# References IX

- [65] R. Sebastiani and S. Tomasi.  
Optimization in SMT with LA(Q) Cost Functions.  
In *IJCAR*, volume 7364 of *LNAI*, pages 484–498. Springer, July 2012.
- [66] R. Sebastiani and S. Tomasi.  
Optimization Modulo Theories with Linear Rational Costs.  
*ACM Transactions on Computational Logics*, 16(2), March 2015.
- [67] R. Shostak.  
A Practical Decision Procedure for Arithmetic with Function Symbols.  
*Journal of the ACM*, 26(2):351–360, 1979.
- [68] R. Shostak.  
Deciding Combinations of Theories.  
*Journal of the ACM*, 31:1–12, 1984.
- [69] K. Stergiou and M. Koubarakis.  
Backtracking algorithms for disjunctions of temporal constraints.  
In *Proc. AAAI*, pages 248–253, 1998.
- [70] A. Stump, C. W. Barrett, and D. L. Dill.  
CVC: A Cooperating Validity Checker.  
In *Proc. CAV'02*, number 2404 in *LNCS*. Springer Verlag, 2002.
- [71] S. Teso, R. Sebastiani, and A. Passerini.  
Structured learning modulo theories.  
*Artificial Intelligence*, 244:166–187, 2017.
- [72] S. Wolfman and D. Weld.  
The LPSAT Engine & its Application to Resource Planning.  
In *Proc. IJCAI*, 1999.

# References X

- [73] T. Yavuz-Kahveci, M. Tuncer, and T. Bultan.  
A Library for Composite Symbolic Representation.  
In *Proc. TACAS2001*, volume 2031 of *LNCS*. Springer Verlag, 2000.
- [74] L. Zhang and S. Malik.  
Extracting small unsatisfiable cores from unsatisfiable boolean formula.  
In *Proc. of SAT*, 2003.

## Disclaimer

The list of references above is by no means intended to be all-inclusive. I apologize both with the authors and with the readers for all the relevant works which are not cited here.