

# Dagstuhl Seminar 07401: Deduction and Decision Procedures

Schloss Dagstuhl, 30th September - 5th October 2007

# Delayed Theory Combination

Roberto Sebastiani

DIT, Università di Trento, Italy.

joint work with

Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Anders Franzen,  
Alberto Griggio, Tommi Junntila, Silvio Ranise, Peter van Rossum

# Content

- ⇒ Lazy Satisfiability Modulo Theories (Lazy SMT) .
- The Nelson-Oppen combination procedure (N.O.)
- Delayed Theory Combination (DTC) . . . . .
- DTC vs. N.O. procedure in lazy SMT . . . . .
- Conclusions & open research issues . . . . .

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

**Satisfiability Modulo Theories ( $SMT(\mathcal{T})$ )** is the problem of deciding the satisfiability of (typically quantifier-free) formulas in some decidable first-order theory  $\mathcal{T}$ .

- ▷ 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 ( $\mathcal{DL}$ ):**  $((x = y) \wedge (y - z \leq 4)) \rightarrow (x - z \leq 6)$
  - **Linear arithmetic over the rationals ( $\mathcal{LA}(\text{Rat})$ ):**  
 $(T_\delta \rightarrow (s_1 = s_0 + 3.4 \cdot t - 3.4 \cdot t_0)) \wedge (\neg T_\delta \rightarrow (s_1 = s_0))$
  - **Linear arithmetic over the integers ( $\mathcal{LA}(\text{Int})$ ):**  
 $(x := x_l + 2^{16}x_h) \wedge (x \geq 0) \wedge (x \leq 2^{16} - 1)$
  - **Bit vectors:**  $(x_{[16]}[15 : 0] := x_{[16]}[15 : 8] \cdot x_{[16]}[7 : 0])$
  - ...

# The Lazy SMT approach

[Armando et al. ECP99; Wolfman&Weld IJCAI99; ...]

- ▷ A prominent “lazy” approach:
  - a **SAT solver** (DPLL) is used to enumerate truth assignments  $\mu_i$  for (the boolean abstraction of) the input formula  $\varphi$
  - a theory-specific solver  **$\mathcal{T}$ -solver** is used to check the  $\mathcal{T}$ -consistency of the **set of  $\mathcal{T}$ -literals** corresponding to each assignment
- ▷ Many lazy SMT tools available ( [Ario](#), [Barcellogic](#), [CVClite/CVC3](#), [MathSAT](#), [Sateen](#), [TSAT++](#), [Yices](#), [VeriFUN](#), [Zapato](#), [Z3](#), ... )
- ▷ many optimization techniques

# Example

 $\varphi =$ 

$$c_1 : \neg(2x_2 - x_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (x_1 - x_5 \leq 1)$$

$$c_3 : (3x_1 - 2x_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3x_1 - 2x_2 \leq 3)$$

$$c_6 : (x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$$

blue = true, red = false

 $\varphi^p =$ 

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

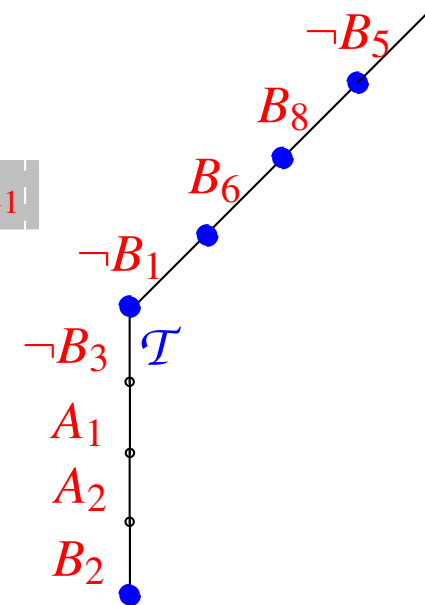
$$B_3 \vee A_2$$

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

$$A_1 \vee B_3$$

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

$$A_1 \vee B_8 \vee A_2$$



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

$$\mu = \{\neg(3x_1 - x_3 \leq 6), (x_3 = 3x_5 + 4), (x_2 - x_4 \leq 6), \neg(2x_2 - x_3 > 2), \neg(3x_1 - 2x_2 \leq 3), (x_1 - x_5 \leq 1)\}$$

$\implies$  inconsistent in  $\mathcal{L}\mathcal{A}(Rat) \implies$  backtrack

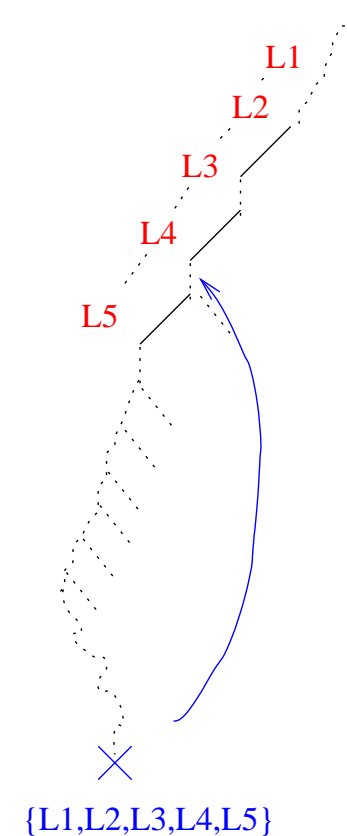
## Lazy SMT: optimization techniques

- ▷  $\mathcal{T}$ -Backjumping
- ▷  $\mathcal{T}$ -learning
- ▷ Early pruning
- ▷  $\mathcal{T}$ -Propagation
- ▷ Normalization of Atoms
- ▷ Static Learning
- ▷ Layering
- ▷ Pure-literal filtering
- ▷  $\mathcal{T}$ -deduced-literals filtering
- ▷ clustering
- ▷ reduction to prime-implicants
- ▷ ...

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

[Horrocks et al. Tableau'98; Wolfman&Weld, AAAI99]

- ▷ Similar to boolean backjumping & learning
- ▷ important property of  $\mathcal{T}$ -solver: when  $\mu$  is  $\mathcal{T}$ -unsatisfiable,  $\mathcal{T}$ -SOLVER( $\mu$ ) returns the subset  $\eta$  of  $\mu$  causing the  $\mathcal{T}$ -inconsistency of  $\mu$  ( $\mathcal{T}$ -conflict set)
- ▷ If so:
  - the conflict clause  $C := \neg\eta$  is learned, and
  - $C$  drives the backjumping mechanism of DPLL
  - $C$  prevents generating the conflict  $\eta$  in future branches.
- ⇒ lots of search saved
- ▷ mixed boolean/theory conflict clauses can be used
- ▷ the less redundant is  $\eta$ , the more search is saved



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

$\varphi =$

$$c_1 : \neg(2x_2 - x_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (x_1 - x_5 \leq 1)$$

$$c_3 : (3x_1 - 2x_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3x_1 - 2x_2 \leq 3)$$

$$c_6 : (x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$$

blue = true, red = false

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

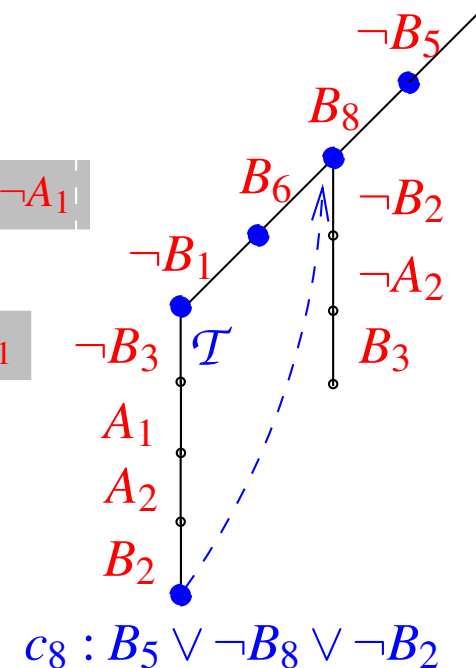
$$B_3 \vee A_2$$

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

$$A_1 \vee B_3$$

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

$$A_1 \vee B_8 \vee A_2$$



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

$$\mu = \{\neg(3x_1 - x_3 \leq 6), (x_3 = 3x_5 + 4), (x_2 - x_4 \leq 6), \neg(2x_2 - x_3 > 2), \neg(3x_1 - 2x_2 \leq 3), (x_1 - x_5 \leq 1)\}$$

$$\eta = \{\neg(3x_1 - x_3 \leq 6), (x_3 = 3x_5 + 4), (x_1 - x_5 \leq 1)\}$$

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

## Early pruning

[Giunchiglia & Sebastiani CADE 96; Armando et al. ECP99; Wolfman&Weld, AAAI99]

- ▷ Introduce a  $\mathcal{T}$ -satisfiability test on **intermediate assignments** :  
if  $\mathcal{T}$ -SOLVER returns Unsat, the procedure backtracks.
- ▷ many different strategies for interleaving DPLL steps with  $\mathcal{T}$ -solver calls
- ▷ important properties of  $\mathcal{T}$ -solvers:
  - **incrementality**:  $\mathcal{T}$ -SOLVER( $\mu_1 \cup \mu_2$ ) reuses computation of  $\mathcal{T}$ -SOLVER( $\mu_1$ ) without restarting from scratch
  - **backtrackability (resettability)**:  $\mathcal{T}$ -SOLVER can efficiently undo steps and return to a previous status on the stack

# $\mathcal{T}$ -Propagation

[Armando et al. ECP'99; Audemard et al. CADE'02; Ganzinger et al CAV04]

- ▷ strictly related to early-pruning
- ▷ important property of  $\mathcal{T}$ -solver: when a partial assignment  $\mu$  is  $\mathcal{T}$ -satisfiable,  $\mathcal{T}$ -SOLVER( $\mu$ ) may be able to return also an assignment  $\eta$  to some unassigned atom occurring in  $\varphi$  s.t.  $\mu \models_{\mathcal{T}} \eta$ .

E.g., if  $(v_1 - v_3 \geq 2), (v_2 = v_3) \in \mu$  and  $(v_1 - v_2 < 1) \notin \mu$  and occurs in  $\varphi$ , then  $\mathcal{T}$ -SOLVER can derive  $\neg(v_1 - v_2 < 1)$  from  $\mu$ .

- ▷ If so:
    - the literal  $\eta$  is then unit-propagated;
    - optionally, a deduction clause  $C := \neg\mu' \vee \eta$  can be learned,  $\mu'$  being the subset of  $\mu$  which caused the deduction ( $\mu' \models_{\mathcal{T}} \eta$ )
- E.g.,  $\neg(v_1 - v_3 \geq 2) \vee \neg(v_2 = v_3) \vee \neg(v_1 - v_2 < 1)$

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

$\varphi =$

$$c_1 : \neg(2x_2 - x_3 > 2) \vee A_1$$

$$c_2 : \neg A_2 \vee (x_1 - x_5 \leq 1)$$

$$c_3 : (3x_1 - 2x_2 \leq 3) \vee A_2$$

$$c_4 : \neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$$

$$c_5 : A_1 \vee (3x_1 - 2x_2 \leq 3)$$

$$c_6 : (x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$$

$$c_7 : A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$$

$\varphi^p =$

$$\neg B_1 \vee A_1$$

$$\neg A_2 \vee B_2$$

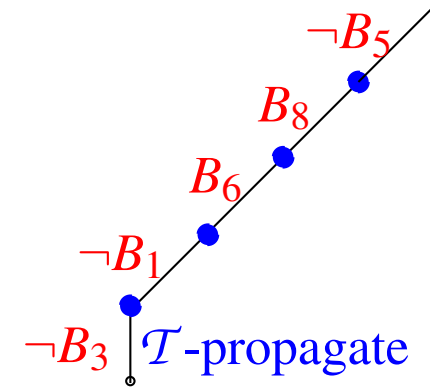
$$B_3 \vee A_2$$

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

$$A_1 \vee B_3$$

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

$$A_1 \vee B_8 \vee A_2$$



blue = true, red = false

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

$$\mu = \{\neg(3x_1 - x_3 \leq 6), (x_3 = 3x_5 + 4), (x_2 - x_4 \leq 6), \neg(2x_2 - x_3 > 2)\} \models_{\mathcal{L}\mathcal{A}(Rat)} \neg(3x_1 - 2x_2 \leq 3)$$

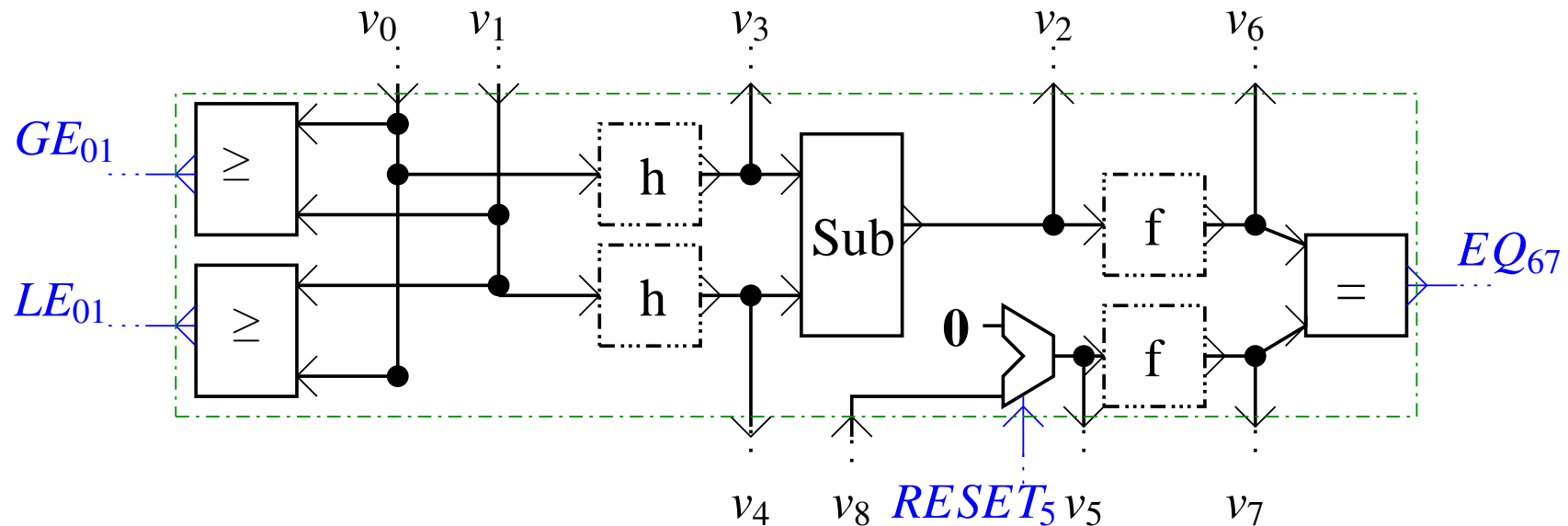
$\implies$  propagate  $\neg B_3$  [and learn the deduction clause  $B_5 \vee B_1 \vee \neg B_3$ ]

# Content

- ✓ Lazy Satisfiability Modulo Theories (Lazy SMT) .
- ⇒ The Nelson-Oppen combination procedure (N.O.)
  - Delayed Theory Combination (DTC) . . . . .
  - DTC vs. N.O. procedure in lazy SMT . . . . .
  - Conclusions & open research issues . . . . .

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

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



$$\mathcal{L}\mathcal{A}(\text{Int}) : \quad (GE_{01} \leftrightarrow (v_0 \geq v_1)) \wedge (LE_{01} \leftrightarrow (v_0 \leq v_1)) \wedge$$

$$\mathcal{E}\mathcal{U}\mathcal{F} : \quad (v_3 = h(v_0)) \wedge (v_4 = h(v_1)) \wedge$$

$$\mathcal{L}\mathcal{A}(\text{Int}) : \quad (v_2 = v_3 - v_4) \wedge (RESET_5 \rightarrow (v_5 = 0)) \wedge$$

$$\mathcal{E}\mathcal{U}\mathcal{F} \text{ or } \mathcal{L}\mathcal{A}(\text{Int}) : \quad (\neg RESET_5 \rightarrow (v_5 = v_8)) \wedge$$

$$\mathcal{E}\mathcal{U}\mathcal{F} : \quad (v_6 = f(v_2)) \wedge (v_7 = f(v_5)) \wedge$$

$$\mathcal{E}\mathcal{U}\mathcal{F} \text{ or } \mathcal{L}\mathcal{A}(\text{Int}) : \quad (EQ_{67} \leftrightarrow (v_6 = v_7)) \wedge \dots$$

# Assumptions & Terminology

[Nelson&Oppen TOCL 79]

We consider “Nelson-Oppen theories”: stably-infinite theories  $\mathcal{T}_i$  with equality and disjoint signatures  $\Sigma_i$

We assume all  $\mathcal{T}_i$ -solvers are incremental and backtrackable

To simplify the explanation, we also make w.l.o.g. the following assumptions:

- ▷ only two theories  $\mathcal{T}_1$  and  $\mathcal{T}_2$ :  $SMT(\mathcal{T}_1 \cup \mathcal{T}_2)$
- ▷ all formulas are in CNF
- ▷ all formulas are **pure**
  - ⇒ interface equalities between **pairs of variables**  $v_i = v_j$   
(hereafter denoted as “ $e_{ij}$ ”)

# The Nelson-Oppen logical framework

[Nelson&Oppen TOCL 79; Oppen, Theor.Comp.Sci.80, Tinelli&Harandi FroCos96]

GOAL:

Determine the  $\mathcal{T}_1 \cup \mathcal{T}_2$ -satisfiability of  $\mu_1 \cup \mu_2$ ,  $\mu_i$  set of  $i$ -pure literals,  $\mathcal{T}_1$  and  $\mathcal{T}_2$  being signature-disjoint and stably-infinite theories with equality

THEOREM:

$\mu_1 \cup \mu_2$  is  $\mathcal{T}_1 \cup \mathcal{T}_2$ -satisfiable iff there exists some equivalence relation  $e(.,.)$  over  $Vars(\mu_1) \cap Vars(\mu_2)$  s.t.  $\mu_i \cup \mu_e$  is  $\mathcal{T}_i$ -satisfiable for every  $i$ , where  $\mu_e$  is:

$$\mu_e =_{def} \{v_i = v_j \mid e(v_i, v_j)\} \cup \{\neg(v_i = v_j) \mid not\ e(v_i, v_j)\}.$$



The problem of  $\mathcal{T}_1 \cup \mathcal{T}_2$ -satisfiability of sets of pure literals  $\mu_1 \cup \mu_2$  in N.O. theories reduces to that of nondeterministically guessing an equivalence relation  $e(.,.)$  over shared variables s.t.  $\mu_i \cup \mu_e$  is  $\mathcal{T}_i$ -satisfiable for every  $i$ .

## Lazy $SMT(\bigcup_i \mathcal{T}_i)$ : approaches

- ▷ Traditional approach: combine  $\mathcal{T}_i$ -solvers into one  $\mathcal{T}_1 \cup \mathcal{T}_2$ -solver via [Nelson-Oppen/Shostak \(N.O.\) combination procedure](#)

[Nelson&Oppen TOCL 79;Shostak JACM 84]

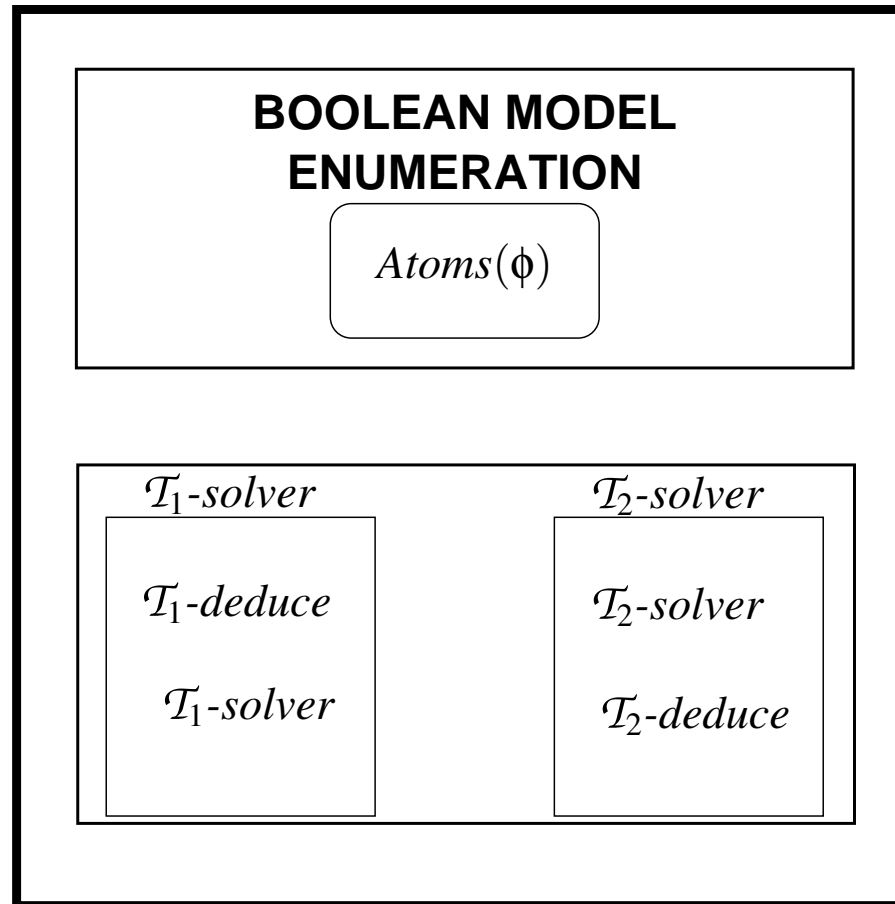
- based on the deduction and exchange of equalities between shared variables/terms (“interface equalities”)
- evolutions of N.O. adopted by many lazy SMT tools (SVC/CVC/CVCLITE/CVC 3, ICS, SIMPLIFY, VERIFUN,...)

- ▷ Recent approach: [Delayed Theory Combination \(DTC\)](#)

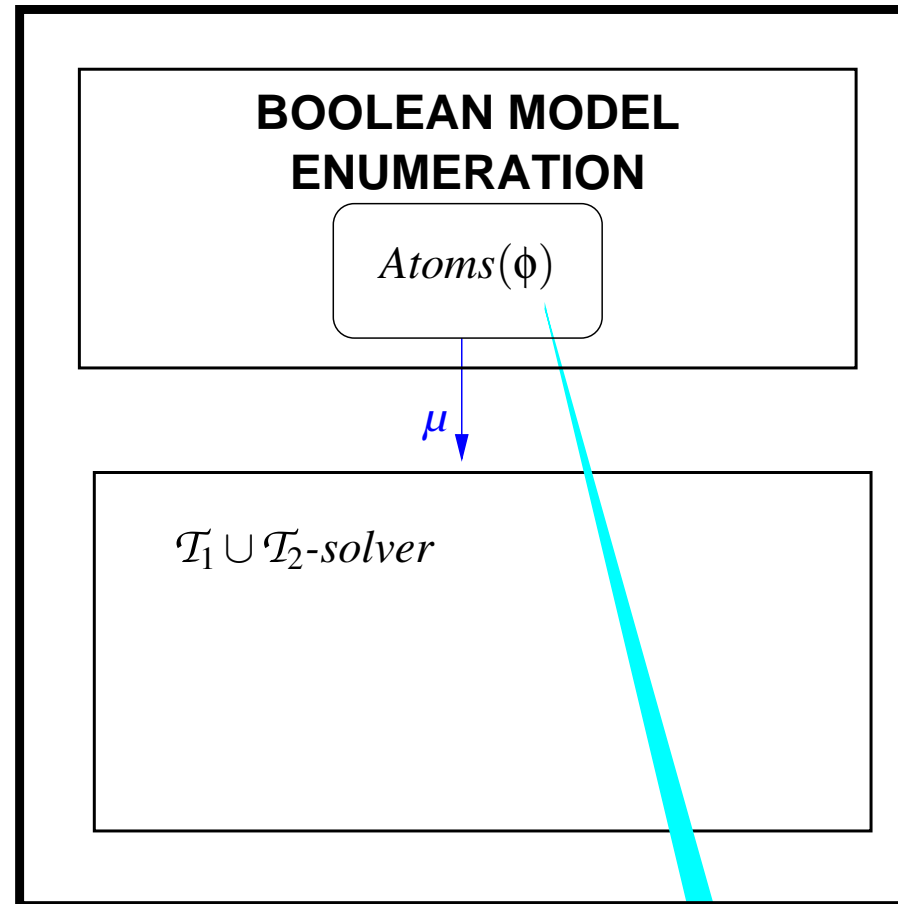
[Bozzano et al. CAV05, Information & Computation 06]

- based on boolean reasoning on interface equalities via DPLL (+  $\mathcal{T}$ -propagation)
- evolutions of DTC adopted by MATHSAT, YICES and Z3

# Lazy SMT with N.O. procedure: basic schema

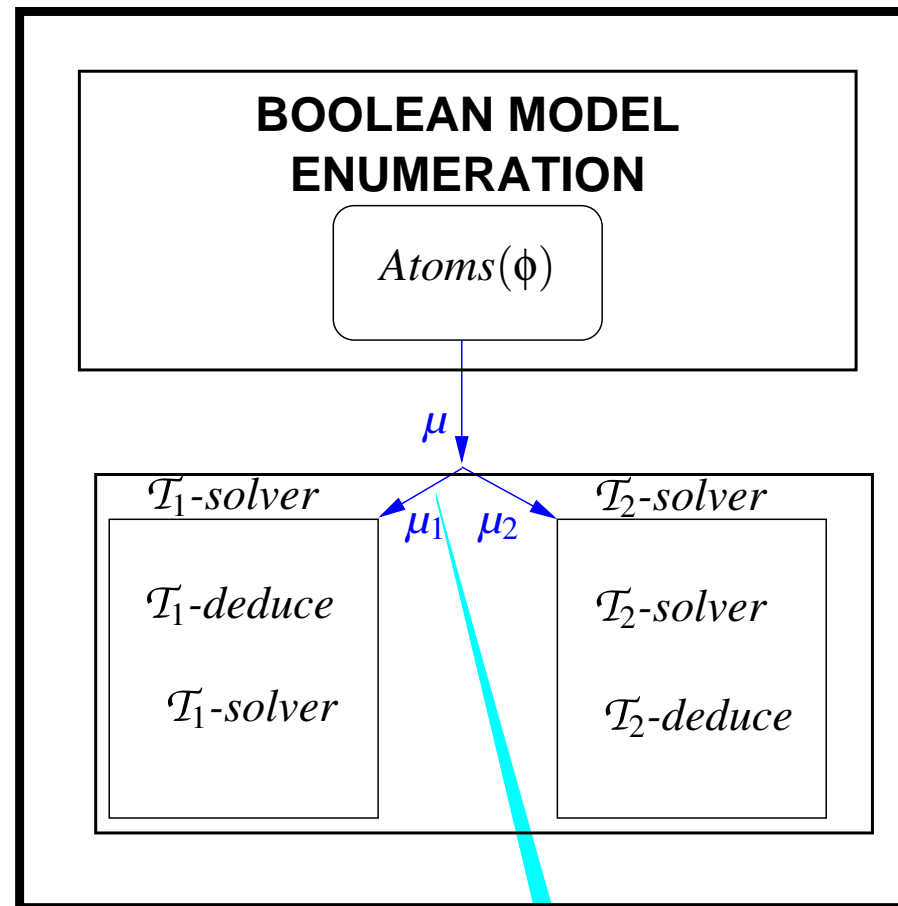


# Lazy SMT with N.O. procedure: basic schema



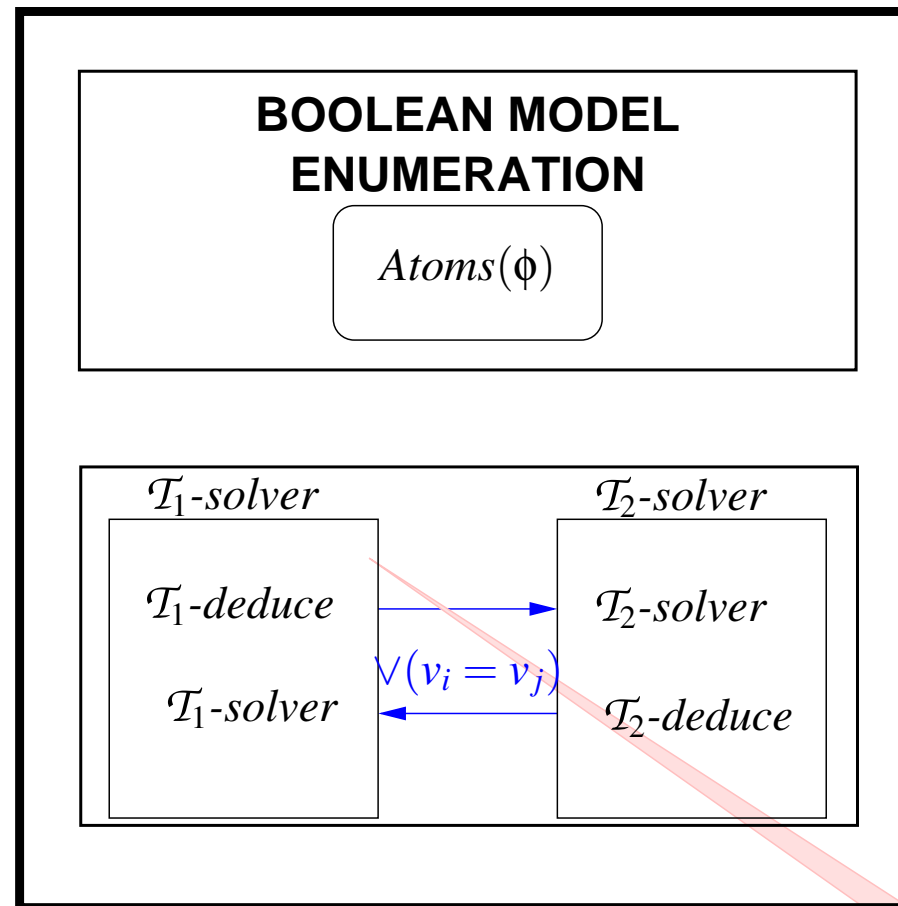
The SAT solver assigns truth values to atoms in  $Atoms(\phi)$ , and passes the corresponding set of literal  $\mu$  to the combined  $\mathcal{T}_1 \cup \mathcal{T}_2$ -solver.

# Lazy SMT with N.O. procedure: basic schema



$\mu$  is partitioned into  $\mu_1 \cup \mu_2$ , s.t.  $\mu_i$  is  $i$ -pure.  
Each  $\mu_i$  is fed to the respective  $\mathcal{T}_i$ -solver.

# Lazy SMT with N.O. procedure: basic schema

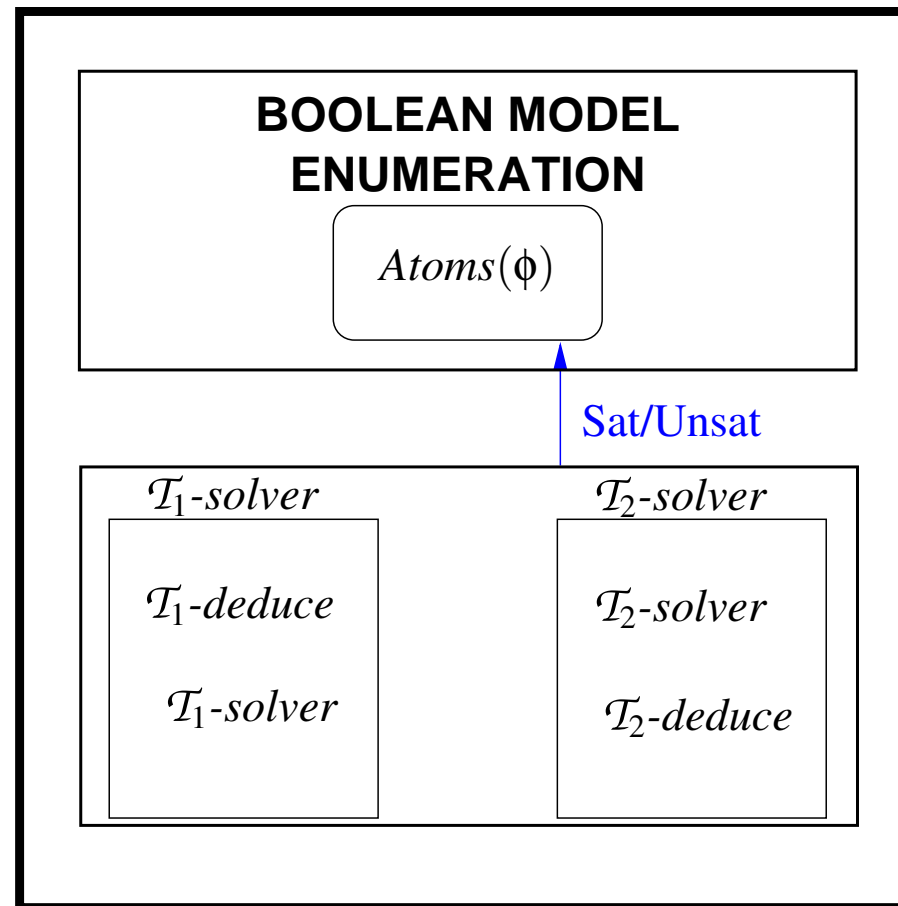


each  $\mathcal{T}_i$ -solver, in turn:

- checks the  $\mathcal{T}_i$ -satisfiability of  $\mu_i$ ,
- deduces all the (disjunctions of) interface equalities deriving from  $\mu_i$ ,
- passes them to the other  $\mathcal{T}$ -solver,

Remark: Disjunctions of literals (due to non-convexity) force case-splitting!

# Lazy SMT with N.O. procedure: basic schema



until either:

- one  $\mathcal{T}_i$ -solver detects inconsistency ( $\mu_1 \cup \mu_2$  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).

## Lazy SMT with N.O. procedure: enhanced schema

- ▷ benefits from all modern lazy SMT techniques
  - DPLL-based assignment enumeration on  $Atoms(\phi)$
  - Theory-Backjumping & Learning
  - Early pruning
  - Theory-propagation
  - ...
- ▷ some important improvements [Barrett et al., FroCoS'02]
  - uses canonizers as in Shostak's method
  - uses interface equalities between **shared terms**  
     $\implies$  allows for avoiding purification
  - ...
- ▷ ...

# Lazy SMT with N.O.: example

$\mu_{\mathcal{L}\mathcal{A}}(\text{Int})$

$$v_1 \geq 0$$

$$v_1 \leq 1$$

$$v_2 \geq v_6$$

$$v_2 \leq v_6 + 1$$

$$v_5 = v_4 - 1$$

$$v_3 = 0$$

$$v_4 = 1$$

$\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$

$$\neg(f(v_1) = f(v_2))$$

$$\neg(f(v_2) = f(v_4))$$

$$f(v_3) = v_5$$

$$f(v_1) = v_6$$

# Lazy SMT with N.O.: example

 $\mu_{\mathcal{L}\mathcal{A}}(\text{Int})$ 

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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

 $\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$ 

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

# Lazy SMT with N.O.: example

 $\mu_{\mathcal{L}\mathcal{A}}(\text{Int})$ 

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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


 $\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$ 

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

$$v_1 = v_3$$

# Lazy SMT with N.O.: example

 $\mu_{\mathcal{L}\mathcal{A}}(Int)$ 

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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


 $\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$ 

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

$$v_1 = v_3$$

*$\langle e_{ij}\text{-deduction} \rangle$*

$$v_5 = v_6$$

# Lazy SMT with N.O.: example

 $\mu_{\mathcal{L}\mathcal{A}}(Int)$ 

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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

$$v_5 = v_6$$

*$\langle e_{ij}\text{-deduction} \rangle$*

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

 $\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$ 

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

$$v_1 = v_3$$

*$\langle e_{ij}\text{-deduction} \rangle$*

$$v_5 = v_6$$

# Lazy SMT with N.O.: example

$\mu_{\mathcal{L}\mathcal{A}}(Int)$

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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

$$v_5 = v_6$$

*$\langle e_{ij}\text{-deduction} \rangle$*

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

$\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$

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



~~$$v_1 = v_3$$~~

*$\langle e_{ij}\text{-deduction} \rangle$*



$$v_5 = v_6$$



~~$$v_2 = v_3$$~~

$\perp$

# Lazy SMT with N.O.: example

$\mu_{\mathcal{L}\mathcal{A}}(Int)$

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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

$$v_5 = v_6$$

*$\langle e_{ij}\text{-deduction} \rangle$*

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

$\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$

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



$$v_1 = v_3$$

*$\langle e_{ij}\text{-deduction} \rangle$*

$$v_5 = v_6$$



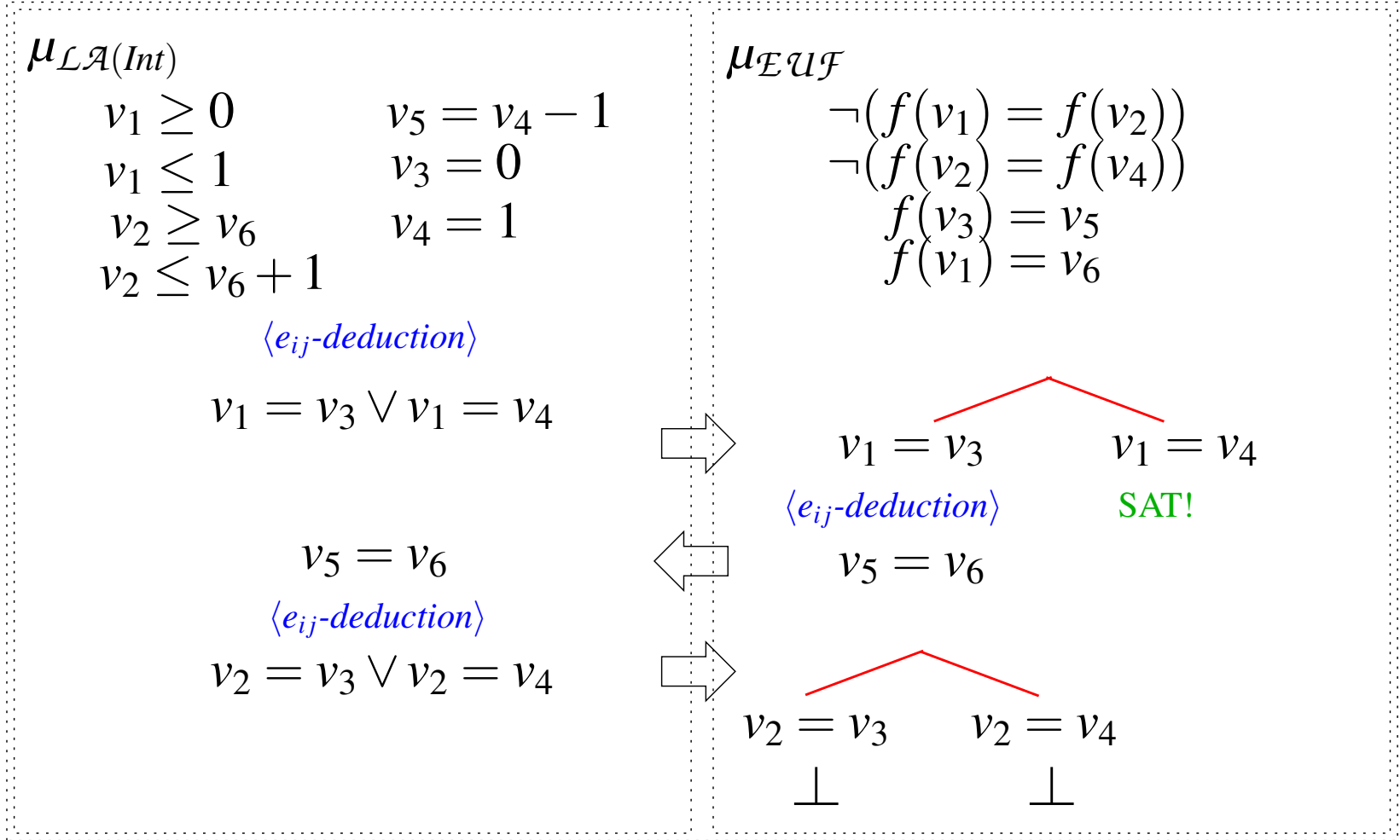
$$v_2 = v_3$$

$\perp$

$$v_2 = v_4$$

$\perp$

# Lazy SMT with N.O.: example



# Content

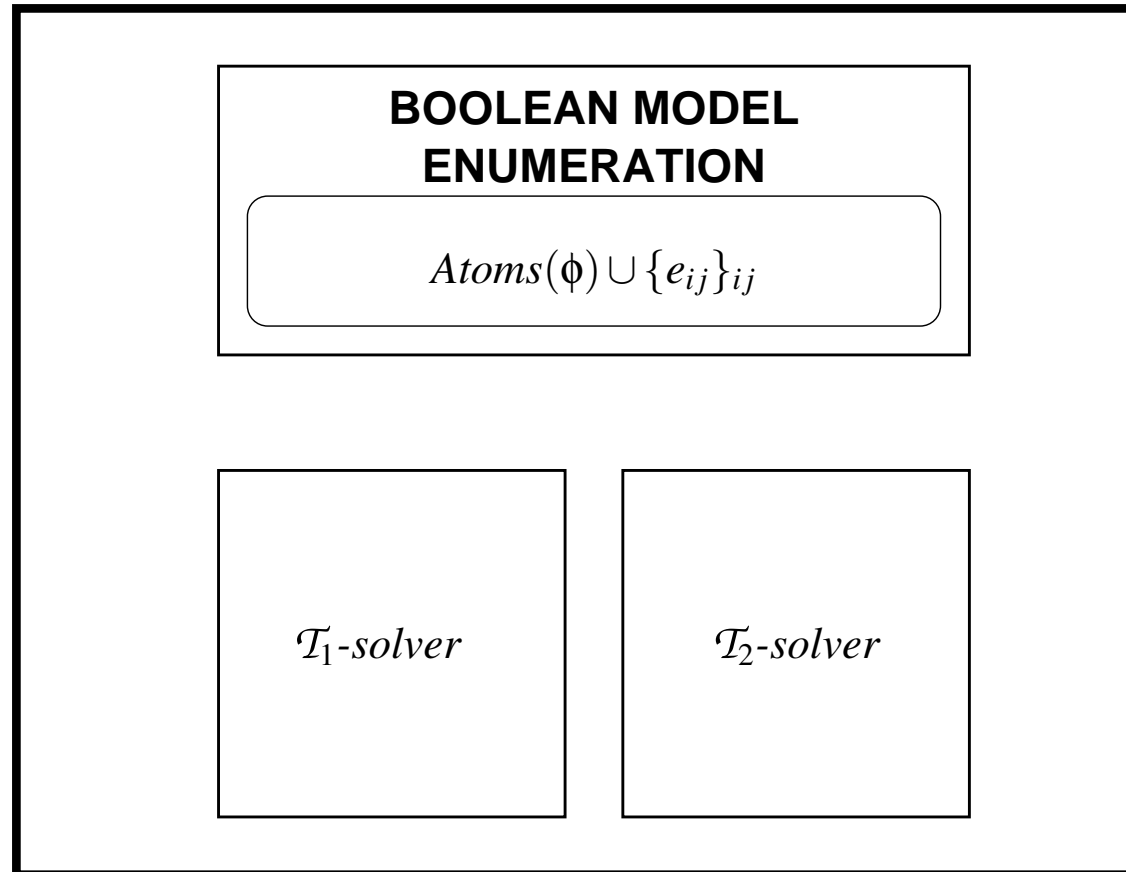
- ✓ Lazy Satisfiability Modulo Theories (Lazy SMT) .
- ✓ The Nelson-Oppen combination procedure (N.O.)
- ⇒ Delayed Theory Combination (DTC) . . . . .
- DTC vs. N.O. procedure in lazy SMT . . . . .
- Conclusions & open research issues . . . . .

## DTC: main ideas

- ▷ Idea: **the interface equalities  $e_{ij}$  take part to the top-level boolean search**
- ▷ The DPLL engine performs three tasks:
  1. enumerates truth assignments (as in standard lazy SMT)
  2. **assigns truth values also to the  $e_{ij}$ 's** the  $\mathcal{T}_i$ -solvers are not able to infer (like the “nondeterministic guesses” of N.O. logical framework in [Oppen, Theor.Comp.Sci.80, Tinelli&Harandi FroCoS96])
  3. [with non-convex theories] **handles the case-splits forced by the deduced disjunctions of  $e_{ij}$ 's**
- ▷ Rationale:

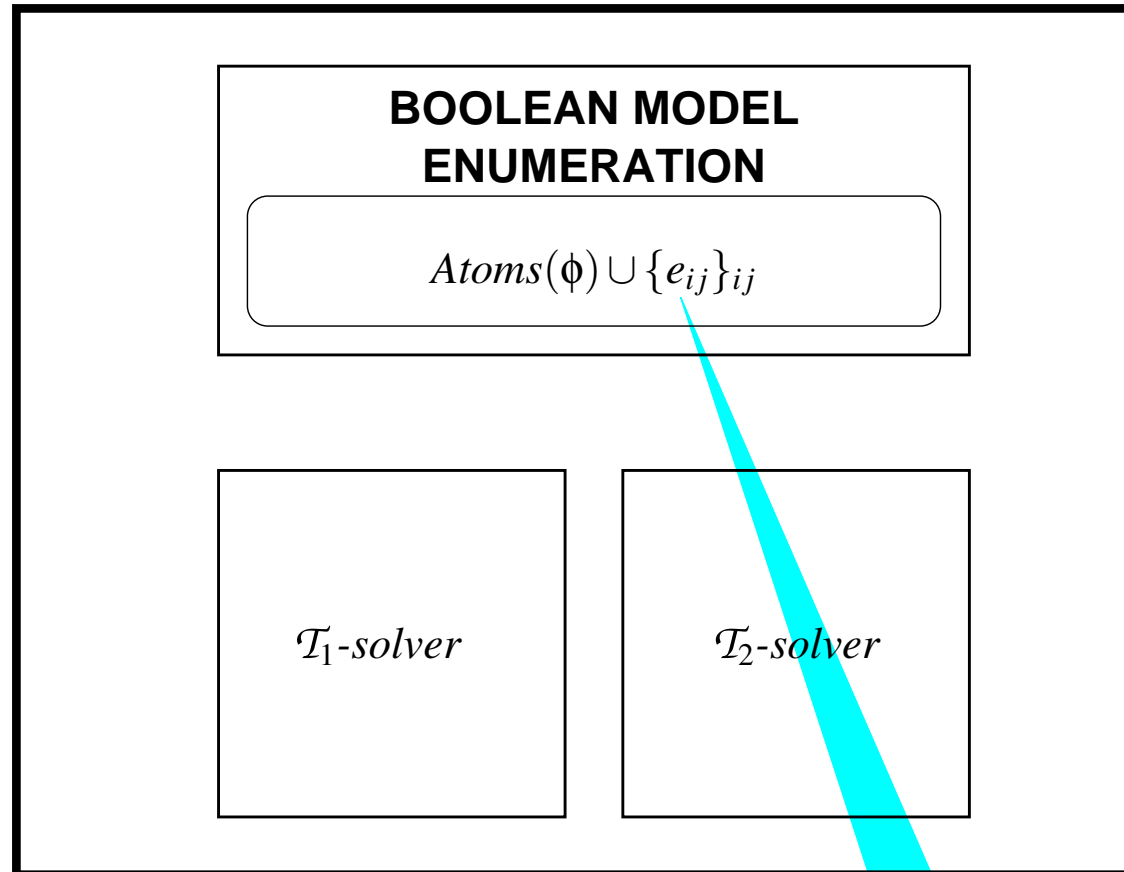
delegate to DPLL part/most of the (possibly very expensive) reasoning effort on  $e'_{ij}$ s previously due to the  $\mathcal{T}_i$ -solvers ( $e_{ij}$ -deduction, case-split).

# DTC: Basic schema (no $e_{ij}$ -deduction)



1st case:  $\mathcal{T}_i$ -solvers have no  $e_{ij}$ -deduction capability

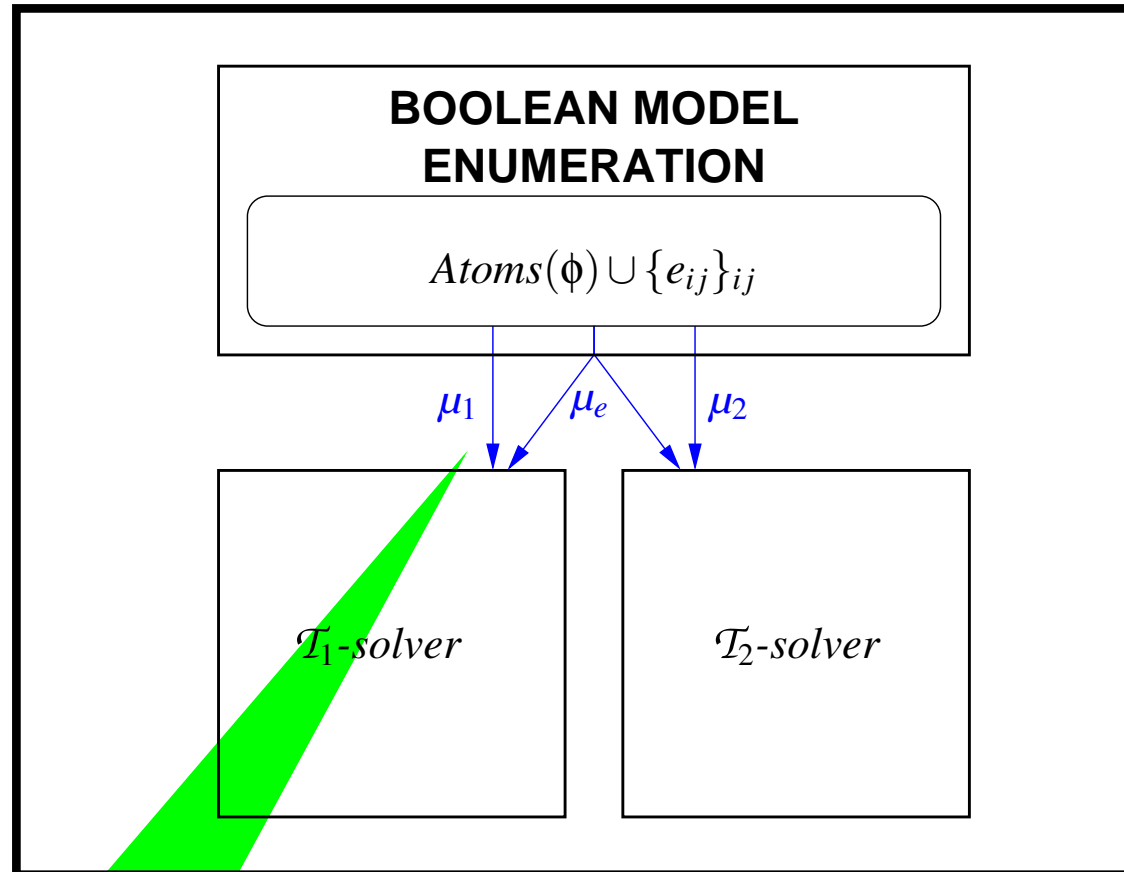
# DTC: Basic schema (no $e_{ij}$ -deduction)



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

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

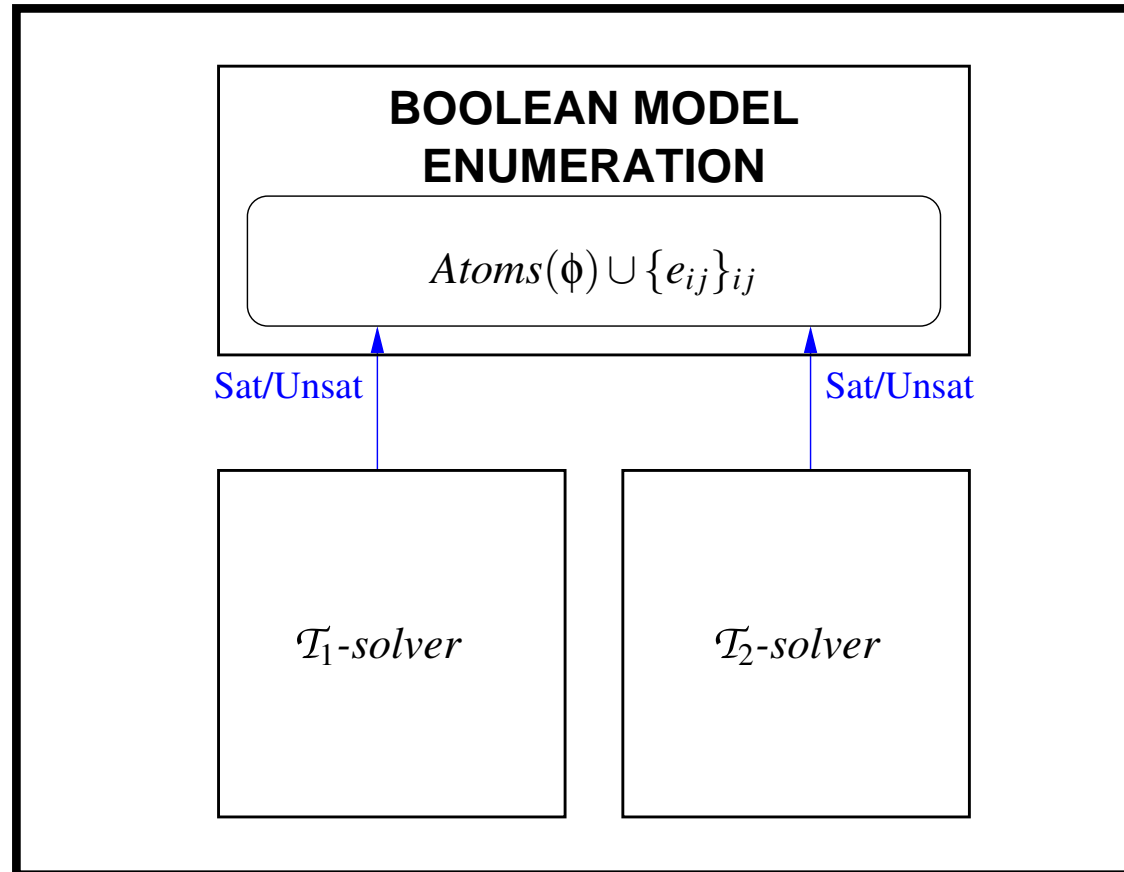
# DTC: Basic schema (no $e_{ij}$ -deduction)



Each  $\mathcal{T}_i$ -solver interacts only with the boolean enumerator (DPLL)

- receives  $\mu_i \cup \mu_e$  from the enumerator
- checks the  $\mathcal{T}_i$ -satisfiability of  $\mu_i \cup \mu_e$

# DTC: Basic schema (no $e_{ij}$ -deduction)



...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 (no $e_{ij}$ -deduction)

- ▷ **Branching** : selection of new  $e_{ij}$ s postponed
  - ⇒ boolean search on  $e_{ij}$ s performed only when strictly necessary (“Delayed Theory Combination”)
- ▷  **$\mathcal{T}$ -Backjumping &  $\mathcal{T}$ -Learning on  $e_{ij}$ s**:  $e_{ij}$ s are involved in conflicts clauses
  - ⇒  $e_{ij}$ s can drive backjumping
  - ⇒ search on  $e_{ij}$ s reused from branch to branch
- ▷  $e_{ij}$ s generated on-demand [Demoura & Dutertre SMT06]
- ▷ **Ad-hoc heuristics** can drive the boolean search over  $e_{ij}$ s [Bruttomesso et al. LPAR 06, Bjorner&Demoura SMT07]
- ▷ ...

# DTC with no $e_{ij}$ -deduction: example

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

# DTC with no $e_{ij}$ -deduction: example

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

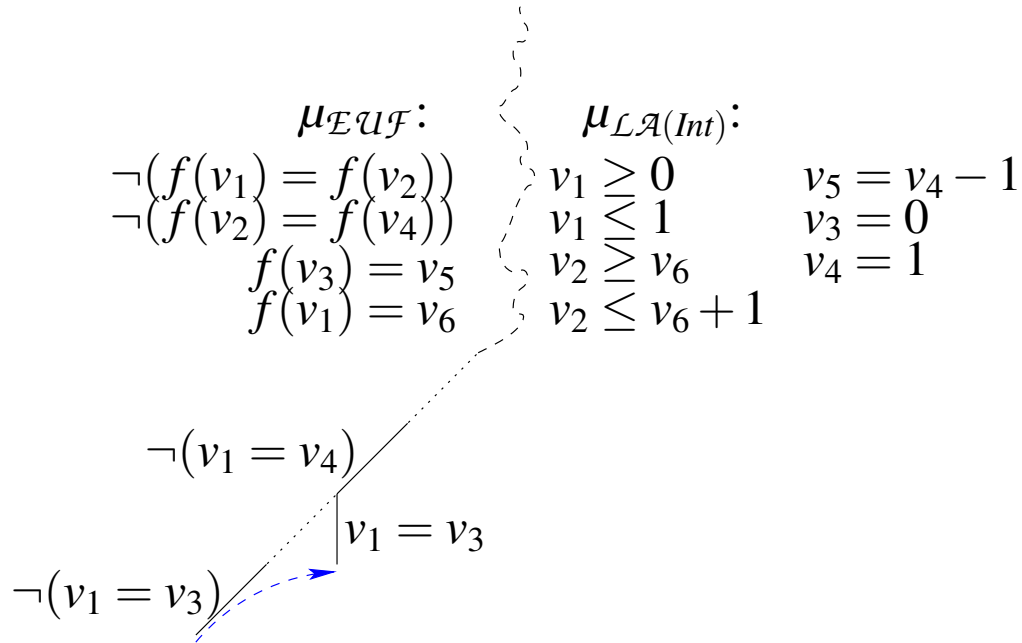
$$\neg(v_1 = v_4)$$

$$\neg(v_1 = v_3)$$

$\mathcal{L}\mathcal{A}(Int)$ -unsat,  $C_{13}$

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

# DTC with no $e_{ij}$ -deduction: example



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

# DTC with no $e_{ij}$ -deduction: example

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

$$\neg(v_1 = v_4)$$

$$\neg(v_1 = v_3)$$

$$\neg(v_5 = v_6)$$

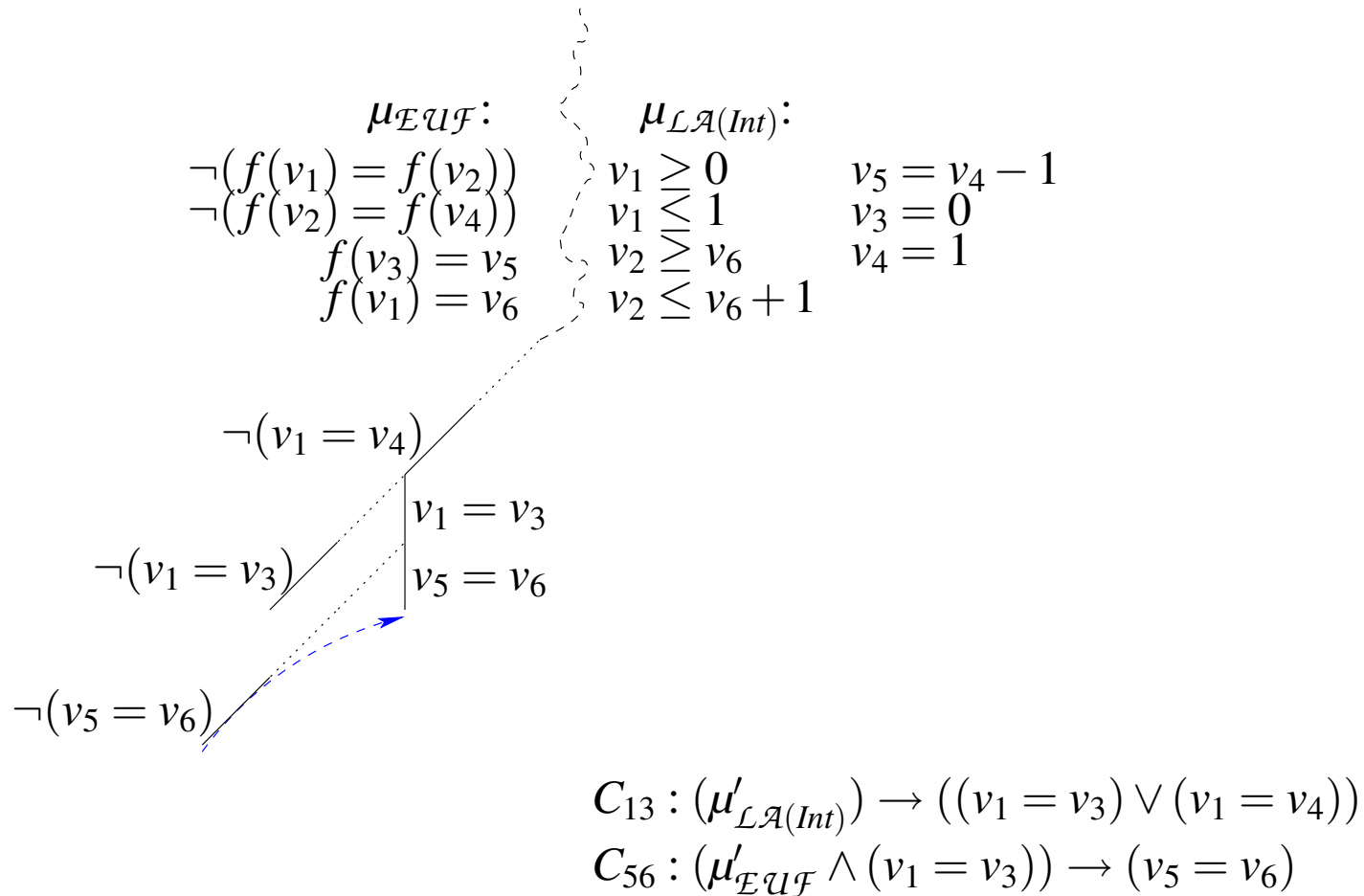
$$v_1 = v_3$$

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

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

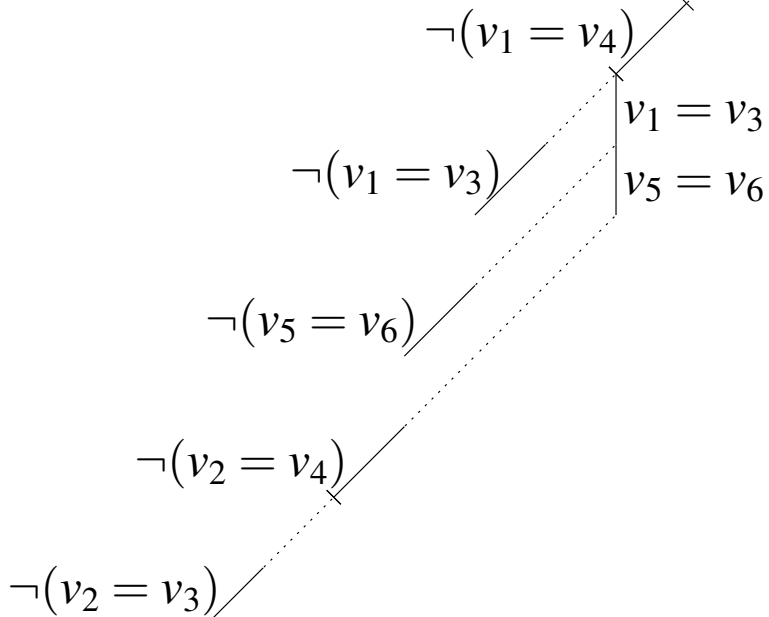
$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

# DTC with no $e_{ij}$ -deduction: example



# DTC with no $e_{ij}$ -deduction: example

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



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

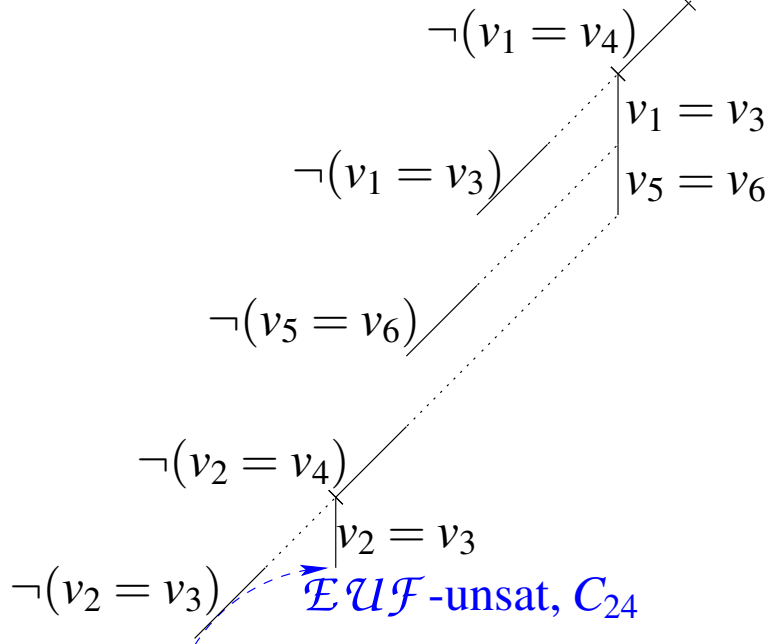
$$C_{56} : (\mu'_{EUF} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

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

$\mathcal{LA}(Int)$ -unsat,  $C_{23}$

# DTC with no $e_{ij}$ -deduction: example

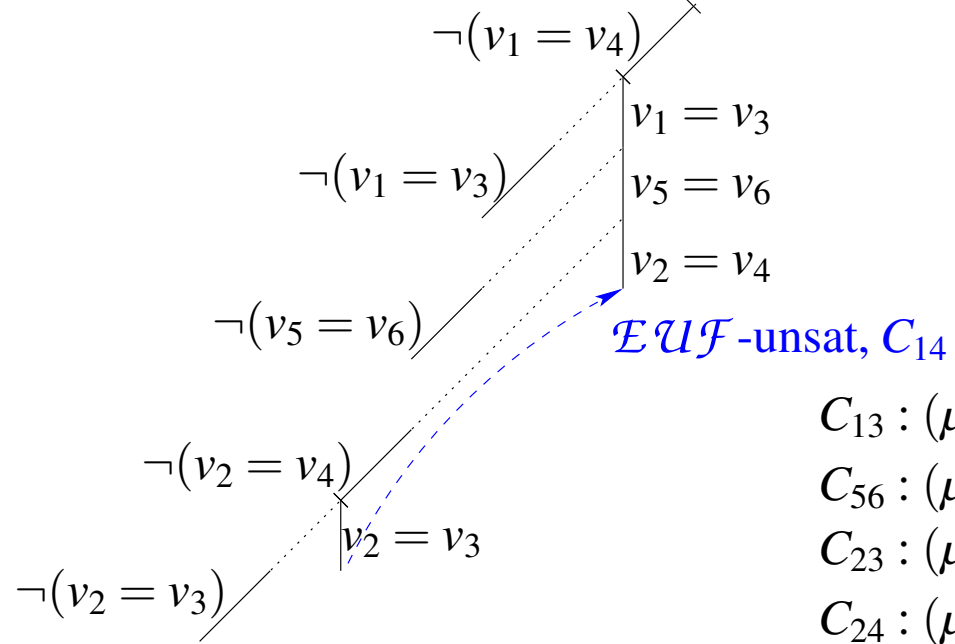
$\mu_{\mathcal{EUF}}:$	$\mu_{\mathcal{LA}(Int)}:$
$\neg(f(v_1) = f(v_2))$	$v_1 \geq 0$
$\neg(f(v_2) = f(v_4))$	$v_1 \leq 1$
$f(v_3) = v_5$	$v_2 \geq v_6$
$f(v_1) = v_6$	$v_2 \leq v_6 + 1$
	$v_5 = v_4 - 1$
	$v_3 = 0$
	$v_4 = 1$



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

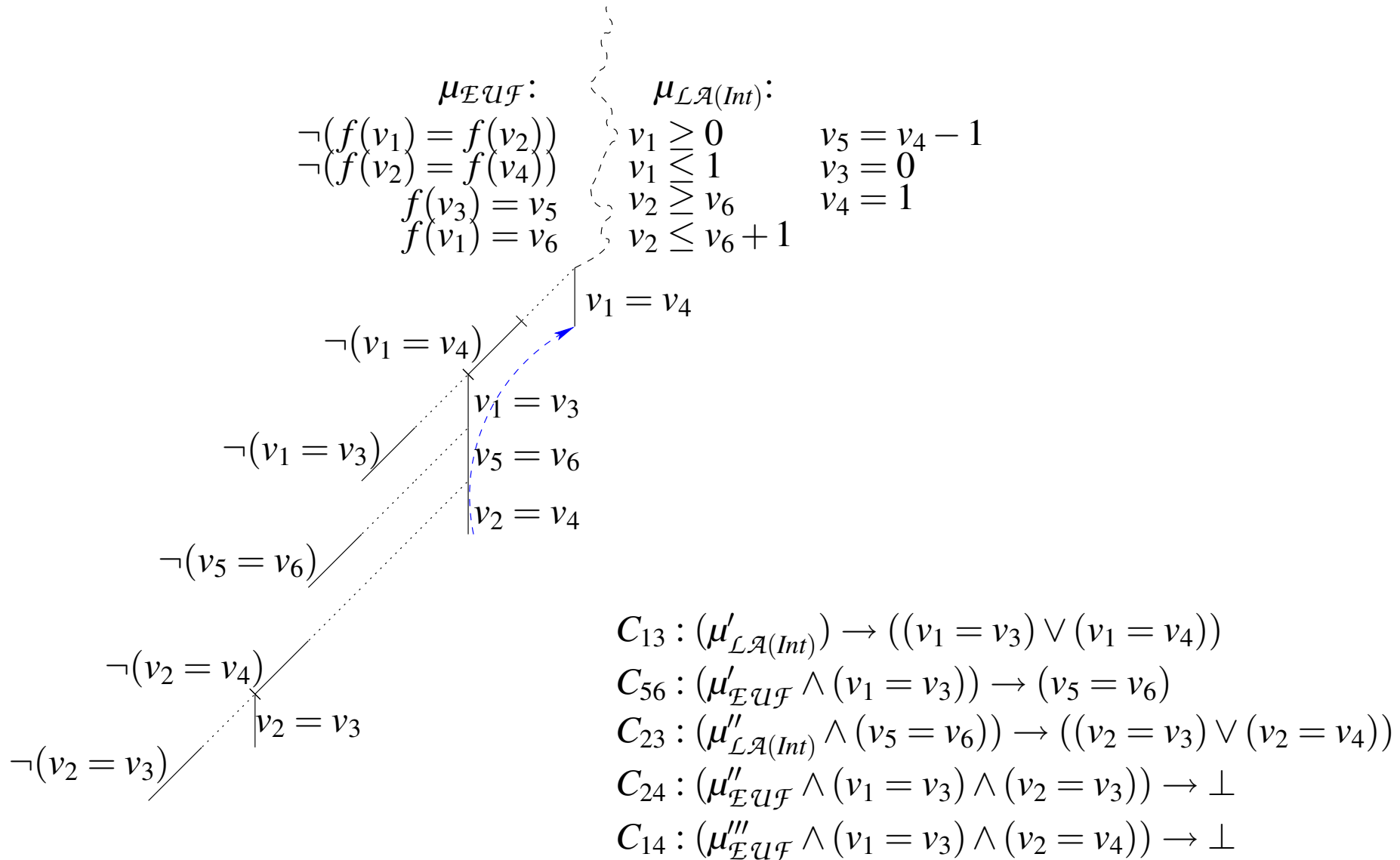
# DTC with no $e_{ij}$ -deduction: example

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

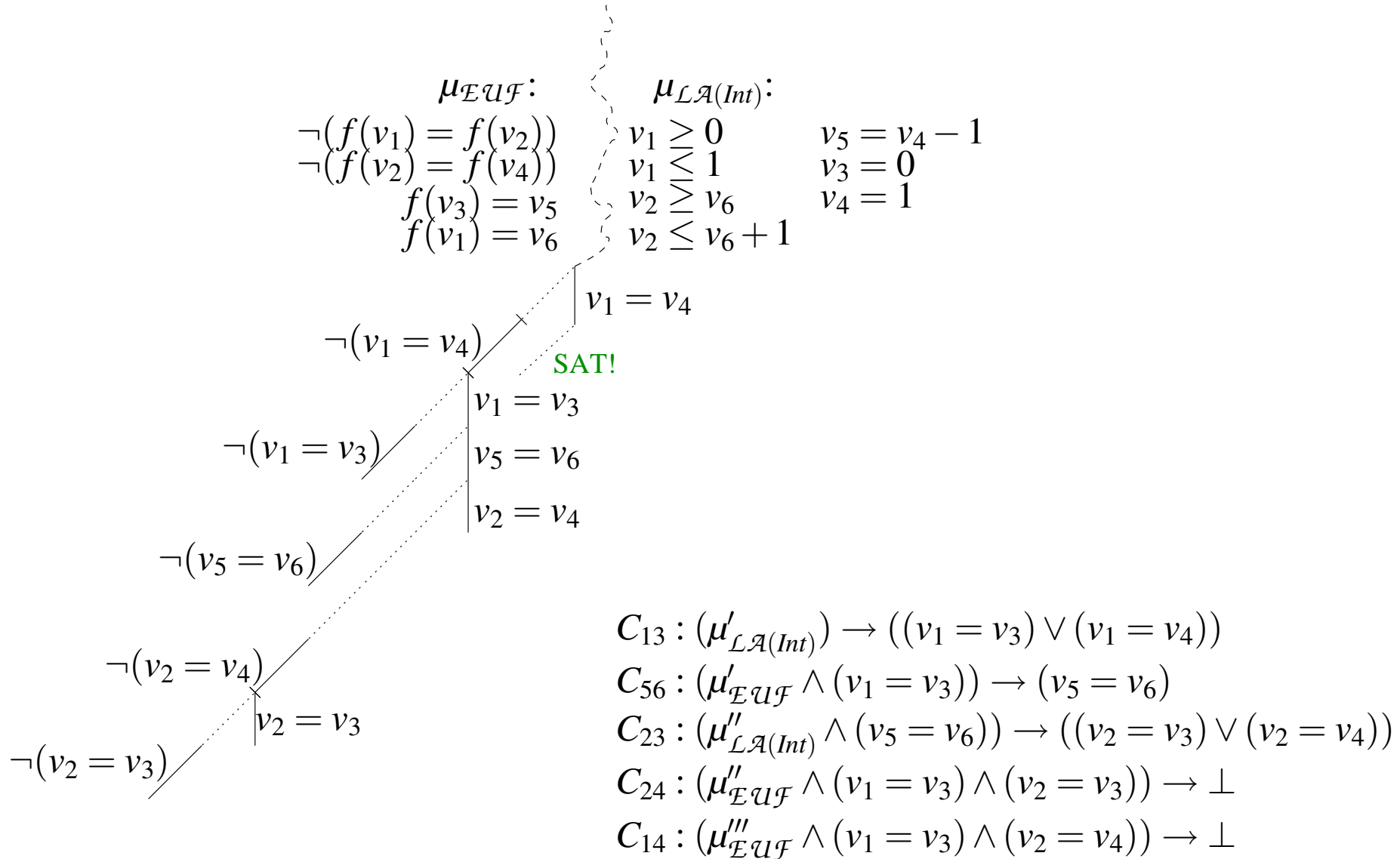


- $C_{13} : (\mu'_{\mathcal{LA}(Int)}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$
- $C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$
- $C_{23} : (\mu''_{\mathcal{LA}(Int)} \wedge (v_5 = v_6)) \rightarrow ((v_2 = v_3) \vee (v_2 = v_4))$
- $C_{24} : (\mu''_{\mathcal{EUF}} \wedge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

# DTC with no $e_{ij}$ -deduction: example



# DTC with no $e_{ij}$ -deduction: example



# Benefits of learned clauses containing $e_{ij}$ 's

$$\phi = \neg(f(x) = f(w_1)) \wedge (A \leftrightarrow \neg(f(x) = f(w_2)))$$

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

$$\wedge (\neg(w_1 = 1) \vee \neg(w_2 = 2) \vee \neg(w_1 = w_2))$$

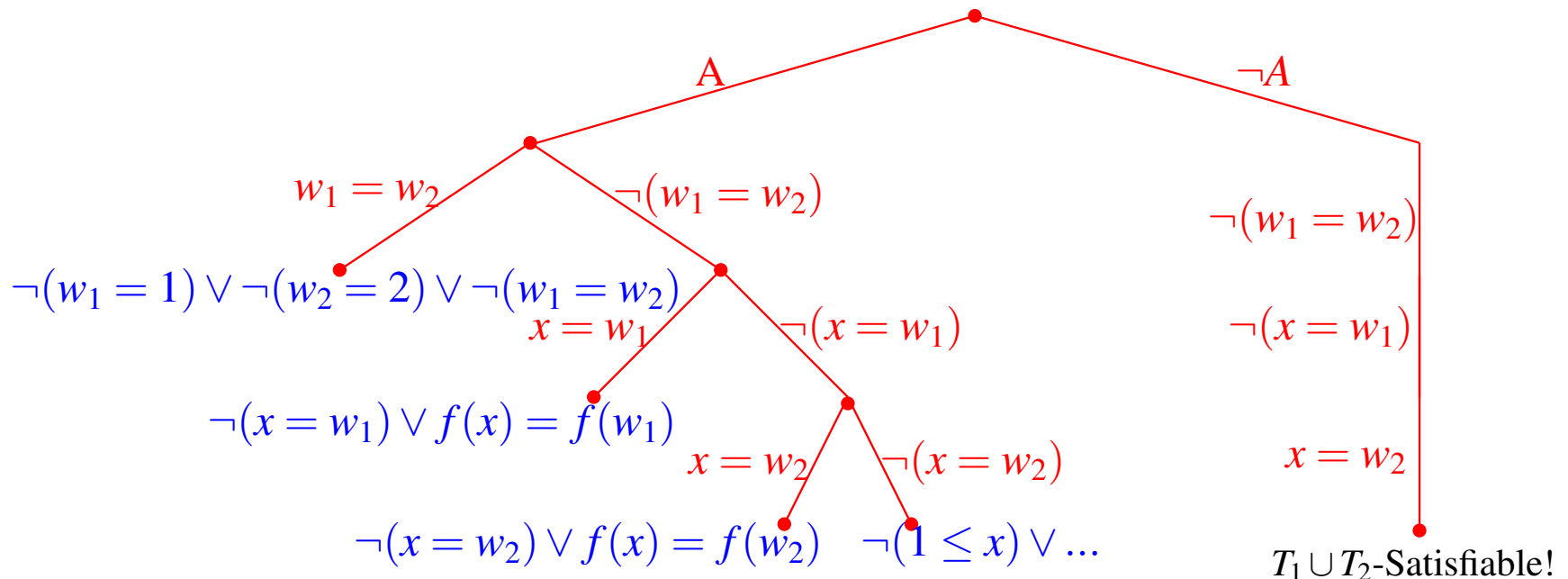
$$\wedge (\neg(x = w_1) \vee f(x) = f(w_1))$$

$$\wedge (\neg(x = w_2) \vee f(x) = f(w_2))$$

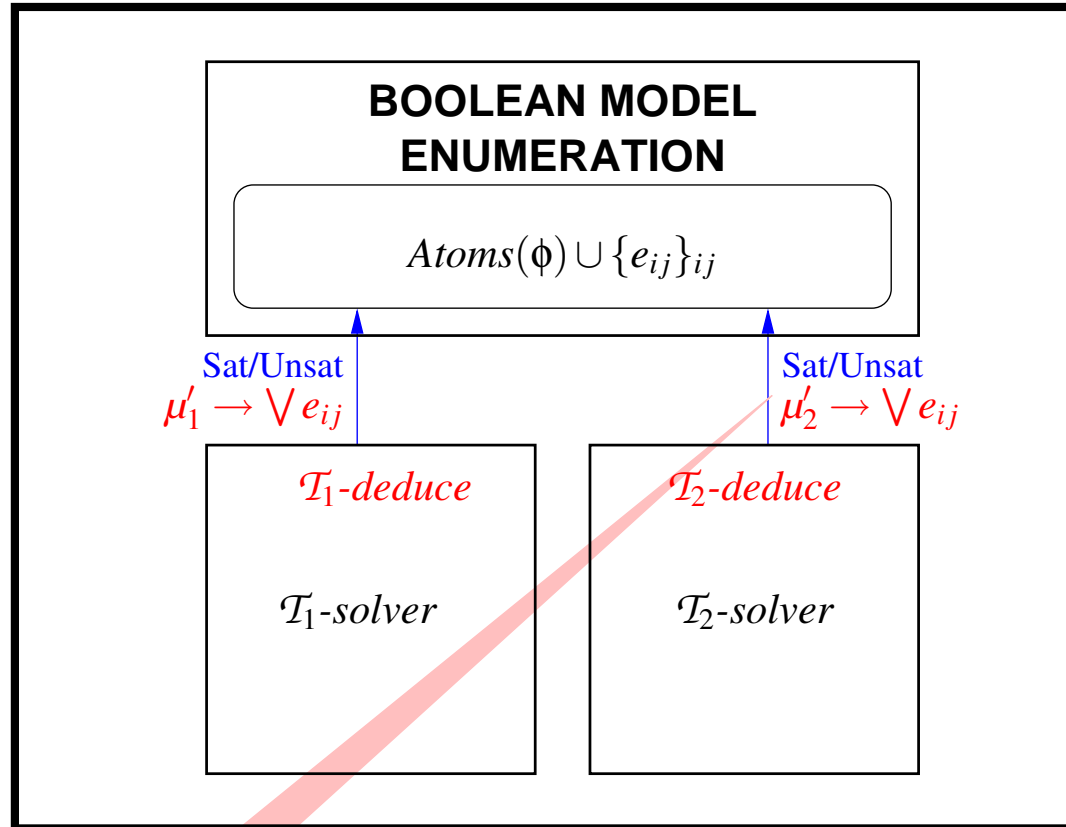
$$\wedge (\neg(1 \leq x) \vee \neg(x \leq 2) \vee \neg(w_1 = 1) \vee \neg(w_2 = 2) \vee x = w_1 \vee x = w_2)$$

$$\mu_1 : \{ \neg(f(x) = f(w_1)), (f(x) = f(w_2)), \neg(w_1 = w_2), \neg(x = w_1), x = w_2 \} \implies \mathcal{T}_1\text{-sat}$$

$$\mu_2 : \{ 1 \leq x, x \leq 2, w_1 = 1, w_2 = 2, \neg(w_1 = w_2), \neg(x = w_1), x = w_2 \} \implies \mathcal{T}_2\text{-sat}$$



# DTC: Schema with $e_{ij}$ -deduction



Returns also the deduction clause of the  $e_{ij}$ -deduction performed by the  $\mathcal{T}_i$ -solver, which is used by DPLL to drive and prune the boolean search

2nd case:  $\mathcal{T}_i$ -solvers have  $e_{ij}$ -deduction capabilities

## DTC with $e_{ij}$ -deduction

▷  $e_{ij}$ -deduction:  $\mathcal{T}_i$ -solver deduces (disjunctions of) unassigned  $e_{ij}$ s:

$$\mu' \models_{\mathcal{T}_i} \bigvee_{i=1}^k e_i, \text{ s.t. } \mu' \subseteq \mu$$

- the deduction clause  $\mu' \rightarrow \bigvee_{i=1}^k e_i$  is learned
- if  $k = 1$ ,  $e_1$  is unit-propagated ( $\mathcal{T}$ -propagation)
- if  $k > 1$ ,  $e_1 \dots e_k$  can be selected next

$\implies$  further reduces the boolean search on  $e_{ij}$ s

$$\text{EX: } \{1 \leq x, x \leq 2, w_1 = 1, w_2 = 2\} \models_{\mathcal{LA}(Int)} (x = w_1) \vee (x = w_2)$$

$$\implies \text{Learn } (1 \leq x \wedge x \leq 2 \wedge w_1 = 1 \wedge w_2 = 2) \rightarrow ((x = w_1) \vee (x = w_2))$$

# DTC with $e_{ij}$ -deduction: example

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

# DTC with $e_{ij}$ -deduction: example

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

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

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

# DTC with $e_{ij}$ -deduction: example

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

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

# DTC with $e_{ij}$ -deduction: example

$$\begin{array}{l}
 \mu_{\mathcal{EUF}}: \\
 \neg(f(v_1) = f(v_2)) \\
 \neg(f(v_2) = f(v_4)) \\
 f(v_3) = v_5 \\
 f(v_1) = v_6 \\
 \\
 \mu_{\mathcal{LA}(Int)}: \\
 v_1 \geq 0 \\
 v_1 \leq 1 \\
 v_2 \geq v_6 \\
 v_2 \leq v_6 + 1 \\
 \\
 v_5 = v_4 - 1 \\
 v_3 = 0 \\
 v_4 = 1 \\
 \\
 \neg(v_1 = v_4) \\
 \\
 \left. \begin{array}{l} v_1 = v_3 \\ v_5 = v_6 \end{array} \right\} \mathcal{EUF}\text{-deduce } (v_5 = v_6), C_{56}
 \end{array}$$

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

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

# DTC with $e_{ij}$ -deduction: example

$$\begin{array}{l}
 \mu_{\mathcal{EUF}}: \\
 \neg(f(v_1) = f(v_2)) \\
 \neg(f(v_2) = f(v_4)) \\
 f(v_3) = v_5 \\
 f(v_1) = v_6 \\
 \\
 \mu_{\mathcal{LA}(Int)}: \\
 v_1 \geq 0 \\
 v_1 \leq 1 \\
 v_2 \geq v_6 \\
 v_2 \leq v_6 + 1 \\
 \\
 v_5 = v_4 - 1 \\
 v_3 = 0 \\
 v_4 = 1 \\
 \\
 \neg(v_1 = v_4) \\
 \left. \begin{array}{l} v_1 = v_3 \\ v_5 = v_6 \end{array} \right\} \mathcal{LA}(Int)\text{-deduce } (v_2 = v_4) \vee (v_2 = v_3), C_{23}
 \end{array}$$

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

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

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

# DTC with $e_{ij}$ -deduction: example

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

$$\begin{array}{l}
 \neg(v_1 = v_4) \\
 v_1 = v_3 \\
 v_5 = v_6 \\
 \neg(v_2 = v_4) \\
 v_2 = v_3 \\
 \times \\
 \mathcal{EUF}\text{-unsat, } C_{24}
 \end{array}$$

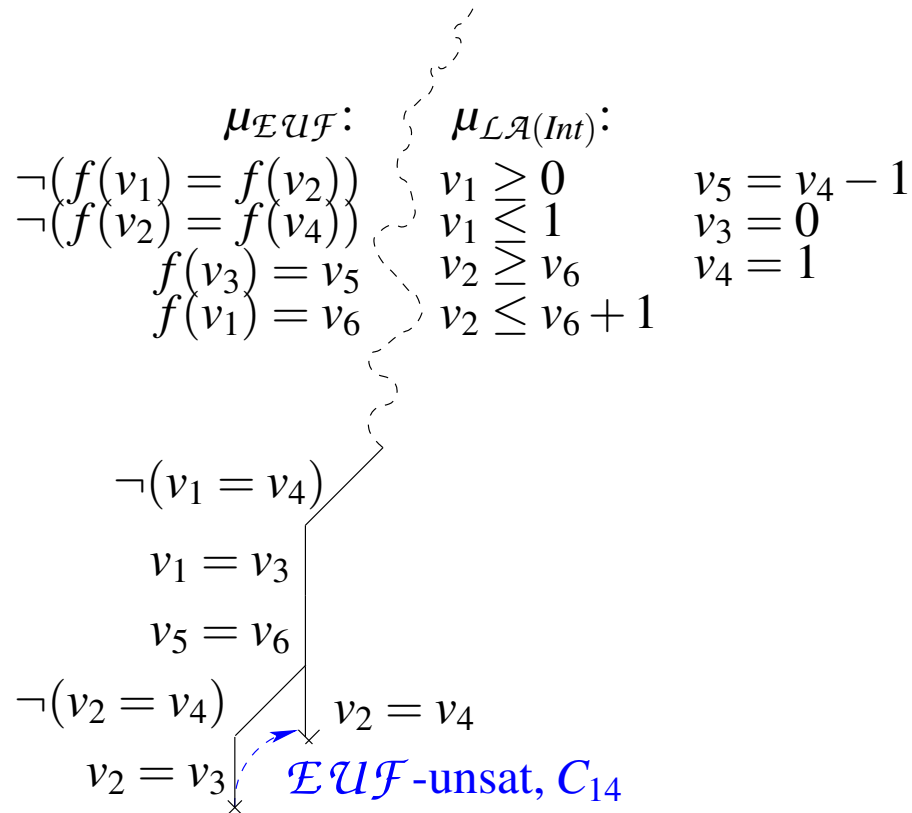
$$C_{13} : (\mu'_{\mathcal{LA}(Int)}) \rightarrow ((v_1 = v_3) \vee (v_1 = v_4))$$

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

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

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

# DTC with $e_{ij}$ -deduction: example



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

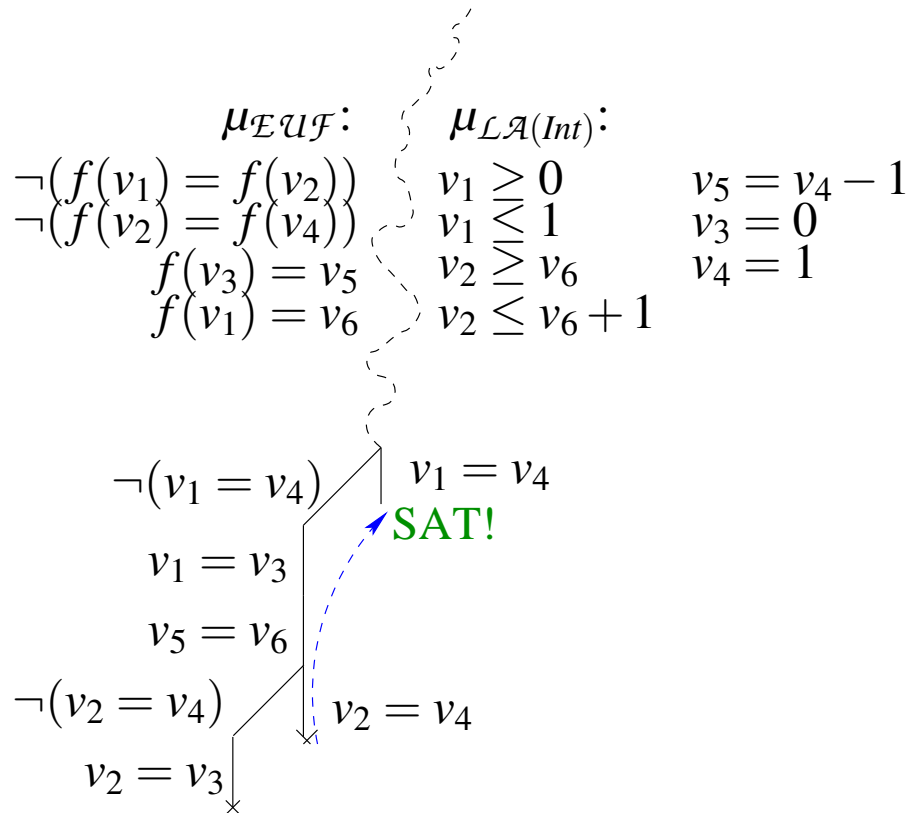
$$C_{56} : (\mu'_{\mathcal{EUF}} \wedge (v_1 = v_3)) \rightarrow (v_5 = v_6)$$

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

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

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

# DTC with $e_{ij}$ -deduction: example



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

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

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

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

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

# Content

- ✓ Lazy Satisfiability Modulo Theories (Lazy SMT) .
- ✓ The Nelson-Oppen combination procedure (N.O.)
- ✓ Delayed Theory Combination (DTC) . . . . .
- ⇒ DTC vs. N.O. procedure in lazy SMT . . . . .
- Conclusions & open research issues . . . . .

## DTC: features

[Bozzano et al et al. CAV05]:

- ▷ No direct interaction between the theory solvers
- ▷ Conflict-directed boolean search on interface equalities  $e_{ij}$ s
  - generation of conflict sets straightforward
  - $e_{ij}$ s can drive backjumping
  - results of reasoning on  $e_{ij}$ s reused from branch to branch
- ▷ Non-convexity handled naturally
  - $\mathcal{T}_i$ -solvers perform no case-split to handle input disjunctions of  $e_{ij}$ s
  - case-splits merged into the top-level boolean search
- ▷ Every  $\mathcal{T}$ -solver can be used, no matter its  $e_{ij}$ -deduction capabilities
  - $\mathcal{T}_i$ -solvers with partial or no  $e_{ij}$ -deduction capability can be used
  - benefit from  $e_{ij}$ -deduction capabilities

## The enlargement of the boolean search space: some results

Two facts: [Bruttomesso et al. LPAR06]:

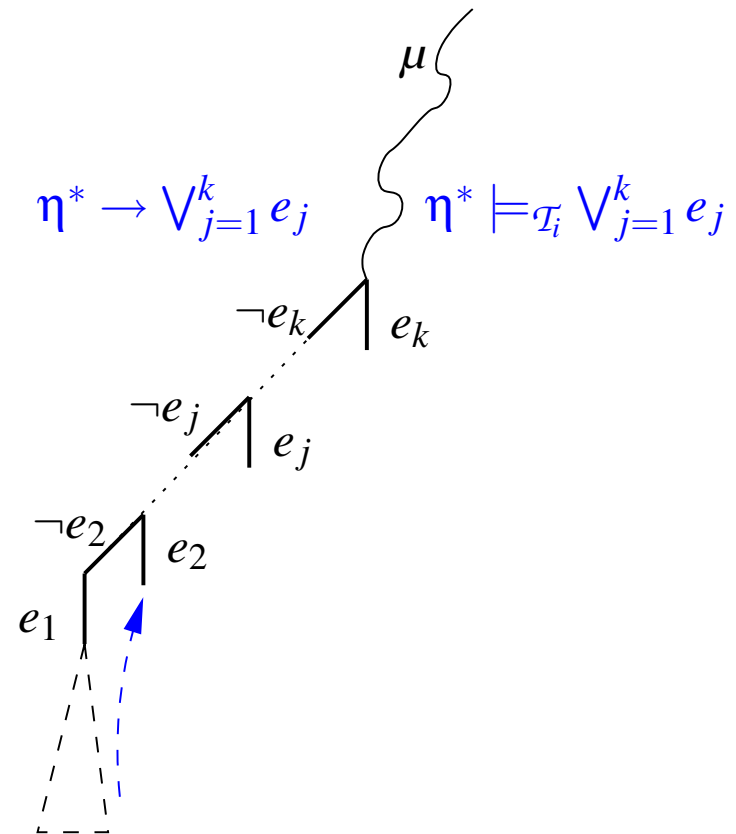
1. If both  $\mathcal{T}_i$ -solvers are  $e_{ij}$ -deduction complete (as with N.O. procedure):  
there is a DTC strategy s.t. the only extra boolean search on  $e_{ij}$ s  
performed by DPLL is that handling the deduced disjunctions of  $e_{ij}$ s  
 $\implies$  DTC “at least as good as N.O. procedure”
2. ...

## DTC with $e_{ij}$ -deduction-complete $\mathcal{T}$ -solvers: intuitions

Hint: when DTC with NO-mimicing strategy performs an  $e_{ij}$ -deduction

$$\eta^* \models_{\mathcal{T}_i} \bigvee_{j=1}^k e_j, \eta^* \subseteq \mu:$$

- learns the clause  $\eta^* \rightarrow \bigvee_{j=1}^k e_j$ ,
- selects sequentially  $\neg e_k, \dots, \neg e_2$
- unit-propagates  $e_1$
- on failure, it backjumps to  $e_2$
- ...



⇒ corresponds to exploring the branches caused by  $e_1, e_2, \dots, e_k$

# Lazy SMT with N.O.: example

 $\mu_{\mathcal{L}\mathcal{A}}(\text{Int})$ 

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

*$\langle e_{ij}\text{-deduction} \rangle$*

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

$$v_5 = v_6$$

*$\langle e_{ij}\text{-deduction} \rangle$*

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

 $\mu_{\mathcal{E}\mathcal{U}\mathcal{F}}$ 

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



$$v_1 = v_3$$

*$\langle e_{ij}\text{-deduction} \rangle$*

$$v_5 = v_6$$



$$v_1 = v_4$$

**SAT!**

**3  $e_{ij}$ -deductions,**

**3 branches**



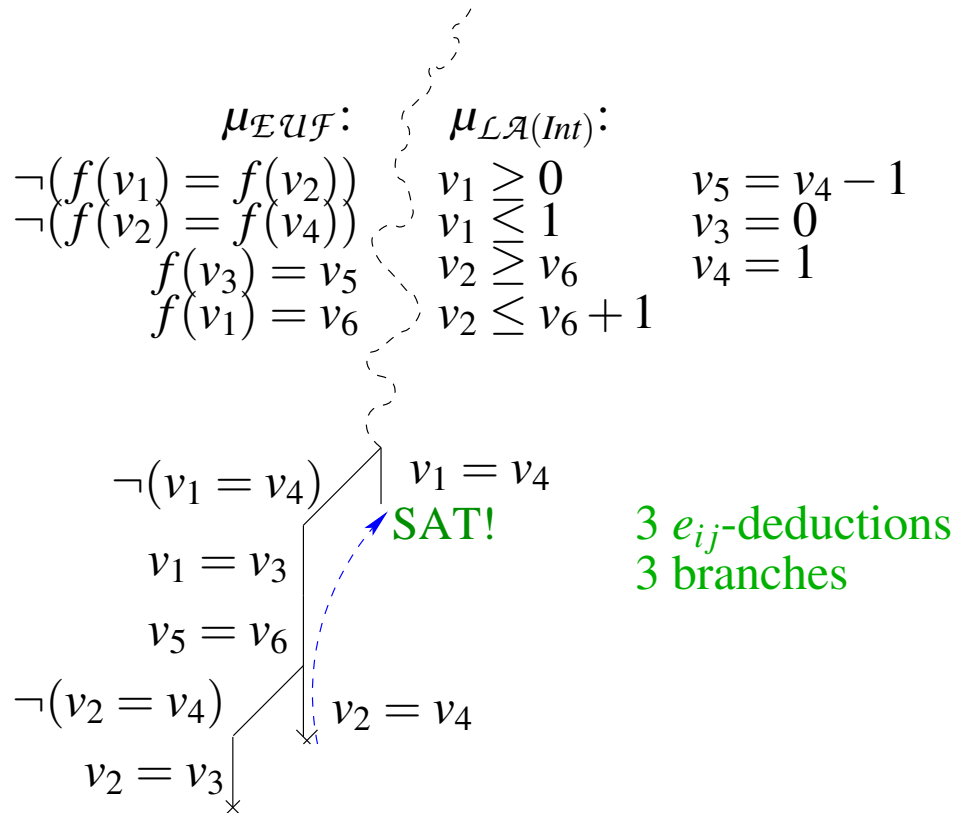
$$v_2 = v_3$$

$\perp$

$$v_2 = v_4$$

$\perp$

# DTC with $e_{ij}$ -deduction: example



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

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

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

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

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

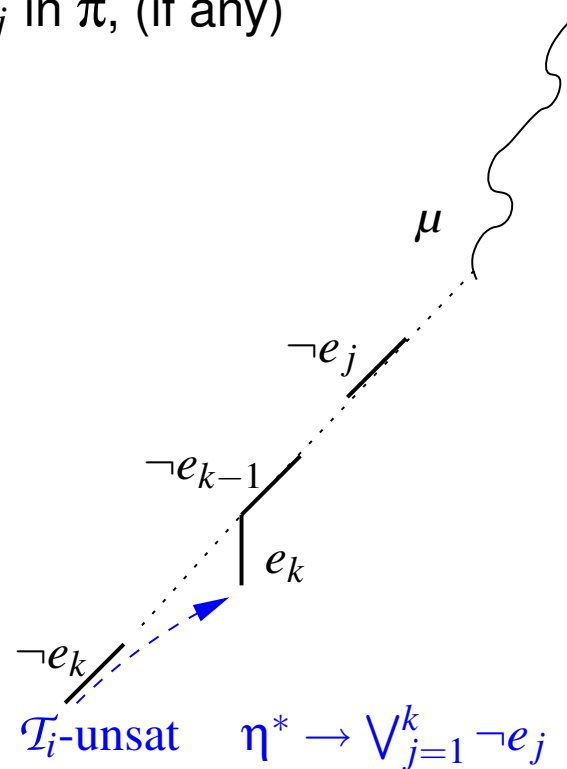
# The enlargement of the boolean search space: some results

Two facts: [Bruttomesso et al. LPAR06]:

1. If both  $\mathcal{T}_i$ -solvers are  $e_{ij}$ -deduction complete (as with N.O. procedure):  
there is a DTC strategy s.t. the only extra boolean search on  $e_{ij}$ s  
performed by DPLL is that handling the deduced disjunctions of  $e_{ij}$ s  
 $\implies$  DTC “at least as good as N.O. procedure”
2. if the  $\mathcal{T}_i$ -solvers are not  $e_{ij}$ -deduction complete (unlike N.O. procedure):  
there is a DTC strategy s.t.
  - if they return “good enough” conflict sets (which contain no  
redundant  $\neg e_{ij}$ 's), DTC can avoid the  $e_{ij}$ -deduction steps at the cost  
of one extra boolean branch explored each
  - in general: the more redundant  $\neg e_{ij}$ 's the  $\mathcal{T}_i$ -solvers remove from  
the conflict sets returned, the more boolean branches are pruned $\implies$  DTC can “emulate”  $e_{ij}$ -deduction by backjumping

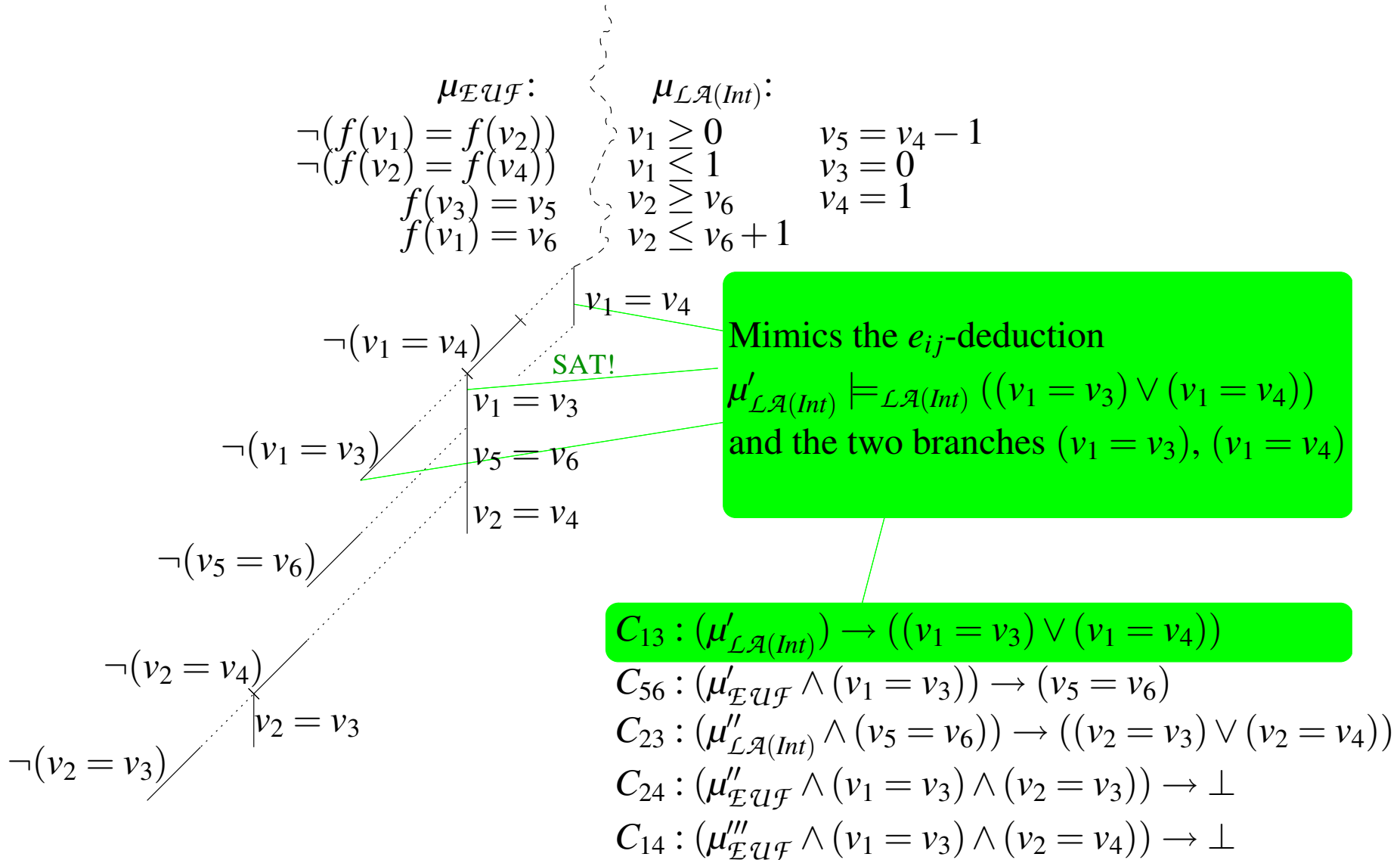
## DTC with partially or non- $e_{ij}$ -deductive $\mathcal{T}$ -solvers: intuitions

Hint: when DTC with NO-mimicing strategy fails and generates a conflict set  $\pi = \eta^* \cup \{\neg e_1, \dots, \neg e_k\}$ , it backjumps up to the second-most-recently-assigned  $\neg e_{ij}$  in  $\pi$ , (if any)



$\implies$  the more redundant  $\neg e_{ij}$ 's the  $\mathcal{T}$ -solver removes from the conflict set returned, the more boolean branches are skipped by backjumping

# DTC with no $e_{ij}$ -deduction: example



# Content

- ✓ Lazy Satisfiability Modulo Theories (Lazy SMT) .
- ✓ The Nelson-Oppen combination procedure (N.O.)
- ✓ Delayed Theory Combination (DTC) . . . . .
- ✓ DTC vs. N.O. procedure in lazy SMT . . . . .
- ⇒ Conclusions & open research issues . . . . .

## Conclusions

**DTC**: delegate to DPLL part/most of the reasoning on  $e'_{ij}$ s previously due to the  $\mathcal{T}_i$ -solvers ( $e_{ij}$ -deduction, case-split).

- ▷ many advantages wrt. N.O. combination procedure in a lazy SMT context
- ▷ DTC “at least as good as N.O. procedure”
- ▷  $\mathcal{T}$ -backjumping &  $\mathcal{T}$ -learning, early-pruning &  $\mathcal{T}$ -propagation essential

**We are sitting on very powerful DPLL tools: let's exploit their full power!**

## Ongoing work & open research problems

- ▷ Improve DTC for N.O. theories
  - novel  $e_{ij}$ -selection heuristics
  - novel  $\mathcal{T}_i$ -solvers able to eliminate redundant  $\neg e_{ij}$ 's from conflict sets
  - splitting-on-demand?
- ▷ DTC beyond N.O. theories

## References

- 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.
- M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Ranise, and R. Sebastiani. Efficient Theory Combination via Boolean Search. *Information and Computation*, 204(10), 2006.
- R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani. Delayed Theory Combination vs. Nelson-Oppen for Satisfiability Modulo Theories: a Comparative Analysis. In *Proc. LPAR'06*, volume 4246 of *LNAI*. Springer, 2006.



Thank you for your attention