VTSA summer school 2015

Exploiting SMT for Verification of Infinite-State Systems

Alberto Griggio
Fondazione Bruno Kessler – Trento, Italy

Agenda

Part 1: Introduction to SMT

Part 2: Interpolation in SMT and in Verification

Part 3: SMT-based Verification with IC3

VTSA summer school 2015

Exploiting SMT for Verification of Infinite-State Systems

1. Introduction to SMT

Alberto Griggio
Fondazione Bruno Kessler – Trento, Italy

Some material courtesy of R. Sebastiani

Outline

Introduction

CDCL-based SAT solvers

The DPLL(T) architecture

Some relevant T-solvers

Combination of theories

Boolean Satisfiability (SAT)

- Given a formula φ in propositional logic, with predicates (aka variables) $A, B, C, \ldots$, find an assignment to the variables

$$\{A \mapsto \top, B \mapsto \bot, \ldots\}$$

that makes the formula true, or prove that none exists

Boolean Satisfiability (SAT)

- Given a formula φ in propositional logic, with predicates (aka variables) $A, B, C, \ldots$, find an assignment to the variables

$$\{A \mapsto \top, B \mapsto \bot, \ldots\}$$

that makes the formula true, or prove that none exists

- Example

$$\varphi := (A \vee (B \wedge C)) \wedge (A \implies \neg C)$$

SAT, with solution (model)

$$\begin{aligned} \mu := \{A &\mapsto \bot, \\ B &\mapsto \top, \\ C &\mapsto \top\} \end{aligned}$$

The SMT problem

- **Satisfiability Modulo Theories**
  - Given a (quantifier-free) FOL formula and a (decidable) combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$, is there an assignment to the free variables $x_1, \ldots, x_n$ that makes the formula true?

- Example:

$$\varphi \stackrel{\text{def}}{=} (x_1 \geq 0) \wedge (x_1 < 1) \wedge$$
$$((f(x_1) = f(0)) \rightarrow (\mathsf{rd}(\mathsf{wr}(P, x_2, x_3), x_2 + x_1) = x_3 + 1))$$

The SMT problem

- **Satisfiability Modulo Theories**
  - Given a (quantifier-free) FOL formula and a (decidable) combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$, is there an assignment to the free variables $x_1, \ldots, x_n$ that makes the formula true?

- Example:

$$\varphi \stackrel{\text{def}}{=} \boxed{(x_1 \geq 0) \wedge (x_1 < 1)} \wedge$$
$$((f(x_1) = f(0)) \rightarrow (\mathsf{rd}(\mathsf{wr}(P, x_2, x_3), \boxed{x_2 + x_1}) = \boxed{x_3 + 1}))$$

Linear Integer Arithmetic (LIA)

The SMT problem

- **Satisfiability Modulo Theories**
  - Given a (quantifier-free) FOL formula and a (decidable) combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$, is there an assignment to the free variables $x_1, \ldots, x_n$ that makes the formula true?

- Example:

$$\varphi \stackrel{\text{def}}{=} \boxed{(x_1 \geq 0) \wedge (x_1 < 1)} \wedge$$
$$\boxed{((f(x_1) = f(0))} \rightarrow (\mathsf{rd}(\mathsf{wr}(P, x_2, x_3), \boxed{x_2 + x_1}) = \boxed{x_3 + 1}))$$

Linear Integer Arithmetic (LIA)

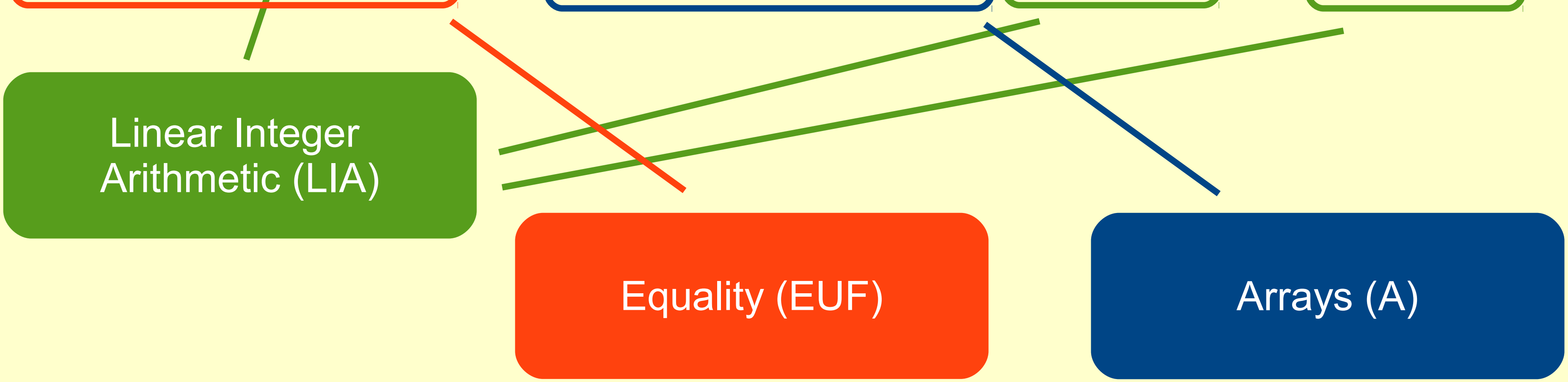
Equality (EUF)

The SMT problem

- **Satisfiability Modulo Theories**
  - Given a (quantifier-free) FOL formula and a (decidable) combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$, is there an assignment to the free variables $x_1, \ldots, x_n$ that makes the formula true?

- Example:

$$\varphi \stackrel{\text{def}}{=} \boxed{(x_1 \geq 0) \wedge (x_1 < 1)} \wedge$$
$$\left(\boxed{(f(x_1) = f(0))} \rightarrow \left(\boxed{\mathsf{rd}(\mathsf{wr}(P, x_2, x_3)}, \boxed{x_2 + x_1}) = \boxed{x_3 + 1}\right)\right)$$



The SMT problem

- **Satisfiability Modulo Theories**
  - Given a (quantifier-free) FOL formula and a (decidable) combination of theories $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$, is there an assignment to the free variables $x_1, \ldots, x_n$ that makes the formula true?

- Example:

$$\varphi \stackrel{\text{def}}{=} (x_1 \geq 0) \wedge (x_1 < 1) \wedge$$
$$((f(x_1) = f(0)) \rightarrow (\mathsf{rd}(\mathsf{wr}(P, x_2, x_3), x_2 + x_1) = x_3 + 1))$$

$$\text{LIA} \models (x_1 = 0)$$
$$\text{EUF} \models f(x_1) = f(0)$$
$$\text{A} \models \mathsf{rd}(\mathsf{wr}(P, x_2, x_3), x_2) = x_3$$
$$\text{Bool} \models \mathsf{rd}(\mathsf{wr}(P, x_2, x_3), x_2 + x_1) = x_3 + 1$$
$$\text{LIA} \models \bot$$

SMT: some history

The “early days”

- **The Simplify theorem prover [Detlefs, Nelson, Saxe]**
  - The grandfather of SMT solvers

- **Efficient decision procedures**
  - Equality logic + extensions (Congruence Closure)
  - Linear arithmetic (Simplex)
  - Theory combination (Nelson-Oppen method)
  - Quantifiers (E-matching with triggers)

- **Inefficient boolean search**

SMT: some history - 2

The SAT breakthrough

- late '90s - early 2000: major progress in SAT solvers

- **CDCL paradigm: Conflict-Driven Clause-Learning DPLL**
  - Grasp, (z)Chaff, Berkmin, MiniSat, ...

- combine strengths of **model search** and **proof search** in a single procedure
  - Model search: efficient BCP and variable selection heuristics
  - Proof search: conflict analysis, non-chronological backtracking, clause learning

- **Smart ideas + clever engineering “tricks”**

SMT: some history - 3

From SAT to SMT

- **exploit advances in SAT solving for richer logics**
  - Boolean combinations of constraints over (combinations of) background theories

- **The Eager approach** (a.k.a. “bit-blasting”)
  - Encode an SMT formula into propositional logic
  - Solve with an off-the-shelf efficient SAT solver
  - Pioneered by UCLID
  - Still the dominant approach for bit-vector arithmetic

SMT: some history - 4

The Lazy approach and DPLL(T) (2002 – 2004)

- **(non-trivial) combination of SAT (CDCL) and T-solvers**
  - SAT-solver enumerates models of boolean skeleton of formula
  - Theory solvers check consistency in the theory
  - Most popular approach (e.g. Barcelogic, CVC4, MathSAT, SMTInterpol, Yices, Z3, VeriT, ...)

- **Yices 1.0 (2006)**
  - The first efficient “general-purpose” SMT solver

- **Z3 1.0 (2008)**
  - > 1600 citations, most influential tool paper at TACAS

Some notation and definitions

- Signature Σ, functions $f, g, h, \ldots$ and predicates $P, Q, R, \ldots$
- Variables $x, y, z, \ldots$, quantifier-free formulas $\varphi, \psi, \ldots$
- **Structure** $\mathcal{A} = (\mathcal{D}, \mathcal{I})$, $\mathcal{I}(f) : \mathcal{D}^n \mapsto \mathcal{D} \quad \mathcal{I}(P) : \mathcal{D}^n \mapsto \{\top, \bot\}$
- **Assignment** $\xi : \xi(x) \mapsto \mathcal{D}$
- Evaluation $[\![\cdot]\!]_{\mathcal{A},\xi} \quad [\![x]\!]_{\mathcal{A},\xi} \stackrel{\text{def}}{=} \xi(x) \quad [\![f(x)]\!]_{\mathcal{A},\xi} \stackrel{\text{def}}{=} \mathcal{I}(f)([\![x]\!]_{\mathcal{A},\xi})$
- φ is **satisfiable** in $\mathcal{A}, \xi$ iff $[\![\varphi]\!]_{\mathcal{A},\xi} = \top$ ($\mathcal{A}, \xi$ is a **model** of φ)
- φ is **valid** in $\mathcal{A}$ ($\mathcal{A} \models \varphi$) iff satisfiable for all ξ ($\mathcal{A}$ is a **model**)
- A **theory** ***T*** is a set of Σ-structures $\mathcal{A}_1 = (\mathcal{D}, \mathcal{I}_1) \ldots \mathcal{A}_n = (\mathcal{D}, \mathcal{I}_n)$

Some notation and definitions

- Signature Σ, functions $f, g, h, \ldots$ and predicates $P, Q, R, \ldots$
- Variables $x, y, z, \ldots$, quantifier-free formulas $\varphi, \psi, \ldots$
- **Structure** $\mathcal{A} = (\mathcal{D}, \mathcal{I})$, $\mathcal{I}(f) : \mathcal{D}^n \mapsto \mathcal{D} \quad \mathcal{I}(P) : \mathcal{D}^n \mapsto \{\top, \bot\}$
- **Assignment** $\xi : \xi(x) \mapsto \mathcal{D}$
- Evaluation $[\![\cdot]\!]_{\mathcal{A},\xi} \quad [\![x]\!]_{\mathcal{A},\xi} \stackrel{\text{def}}{=} \xi(x) \quad [\![f(x)]\!]_{\mathcal{A},\xi} \stackrel{\text{def}}{=} \mathcal{I}(f)([\![x]\!]_{\mathcal{A},\xi})$
- φ is **satisfiable** in $\mathcal{A}, \xi$ iff $[\![\varphi]\!]_{\mathcal{A},\xi} = \top$ ($\mathcal{A}, \xi$ is a **model** of φ)
- φ is **valid** in $\mathcal{A}$ ($\mathcal{A} \models \varphi$) iff satisfiable for all ξ ($\mathcal{A}$ is a **model**)
- A **theory** ***T*** is a set of Σ-structures $\mathcal{A}_1 = (\mathcal{D}, \mathcal{I}_1) \ldots \mathcal{A}_n = (\mathcal{D}, \mathcal{I}_n)$
- Example: $\Sigma = \{\hat{0}, \hat{1}, \ldots, \hat{+}, \hat{\cdot}, \hat{\geq}, \ldots\}$ $\mathcal{D} = \mathbb{Z}$ $\mathcal{I}(\hat{n}) = n \quad \mathcal{I}(\hat{+}) = +$

  $(\hat{3}x + y \mathrel{\hat{\geq}} \hat{5})$ is satisfiable $\quad \xi \stackrel{\text{def}}{=} \{x \mapsto 1, y \mapsto 2\}$ is a model

  $(\hat{2}x + \hat{3}y \mathrel{\hat{\geq}} 0) \wedge (-\hat{1} \mathrel{\hat{\geq}} x) \wedge (-\hat{1} \mathrel{\hat{\geq}} y)$ is unsatisfiable

Outline

Introduction

CDCL-based SAT solvers

The DPLL(T) architecture

Some relevant T-solvers

Combination of theories

CDCL-based SAT solvers

- Conflict-Driven Clause-Learning paradigm
  - Architecture of modern SAT solvers (e.g. Minisat, Lingeling, ...)
  - The "DPLL" part of DPLL(T)

- Combine efficient model search and conflict analysis
  - Model search
    - Stack-based representation of partial truth assignment (trail), extended by performing deductions and decisions
      - When all variables are assigned, retur SAT with trail as model
  - Conflict analysis
    - When a conflict is detected, apply boolean resolution to generate a new implied clause that contradicts the trail
      - learn the blocking clause and use it for non-chronological backtracking
      - when the empty clause is derived, return UNSAT

The CDCL algorithm for SAT

```
CDCL(F)
  A = [], decision_level = 0
  while (true)
    if (deduce(F, A))
       if (!all_assigned(F, A))
         lit = decide(F, A)
         decision_level++
         A = A + (lit, -)
       else return SAT
    else
      lvl, cls = analyze(F, A)
      if (lvl < 0) return UNSAT
      else
         backtrack(F, A, lvl)
         learn(cls)
         decision_level = lvl
```

The CDCL algorithm for SAT

```
CDCL(F)
  A = [], decision_level = 0
  while (true)
    if (deduce(F, A))
       if (!all_assigned(F, A))
         lit = decide(F, A)
         decision_level++
         A = A + (lit, -)
       else return SAT
    else
      lvl, cls = analyze(F, A)
      if (lvl < 0) return UNSAT
      else
         backtrack(F, A, lvl)
         learn(cls)
         decision_level = lvl
```

Trail of assignments *(lit, reason)*

Model Search

Conflict Analysis

CDCL: model search

- Explore search space by adding elements to the trail
  - Trail encodes a set of partial assignments

$$\neg A; B; C \mapsto \{\sigma \mid \sigma(A) = \bot \wedge \sigma(B) = \sigma(C) = \top\}$$

- deductions using unit propagation
  - If a clause has one unassigned literal and all the others set to false, propagate the value of the missing one

$$\left.\begin{array}{l}\text{Trail: } \neg A; B; C \\ \text{Clause: } A \vee \neg B \vee \neg D\end{array}\right\} \neg A; B; C; {\color{red}\neg D}$$

  - All literals assigned by unit propagation have an associated reason clause in the trail
    - The unit clause that forced the assignment to the literal

CDCL: model search

- if a clause has all literals assigned to false, deduce returns false and marks the clause as conflicting

- otherwise, if no more deduction is possible, decide picks an unassigned literal to add to the trail
  - No reason is attached to the literal in this case
  - Decisions partition the trail into **decision levels**

- When **all literals** are **assigned**, SAT is returned
  - The trail is a **model** for the input CNF

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$

Decide A_1

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$\boxed{c_2 : \neg A_1 \vee A_3 \vee A_9}$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$

Deduce

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$

Decide A_{12}

Example

Input clauses

$\boxed{c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$

Deduce

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$\boxed{c_3 : \neg A_2 \vee \neg A_3 \vee A_4}$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1

Deduce

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$\boxed{c_4 : \neg A_4 \vee A_5 \vee A_{10}}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3

Deduce

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$\boxed{c_5 : \neg A_4 \vee A_6 \vee A_{11}}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Deduce

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4
A_6	c_5

Example

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

Conflict!

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4
A_6	c_5

CDCL: conflict analysis

- Goal: backtrack from inconsistent assignment, and avoid repeating the same mistake in the future
- Naive approach: collect **all decisions** in the trail, and learn a blocking clause implying that at least one of them must be flipped
- Proof-based approach: exploit the information in the trail to generate an explanation for the conflict
  - Generate a new lemma, using boolean resolution, that blocks the current assignment and all those sharing the same reason for inconsistency
  - Resolution rule

$$\frac{A \vee B_1 \vee x \qquad A \vee B_2 \vee \neg x}{A \vee B_1 \vee B_2}$$

CDCL: conflict analysis

- Goal: backtrack from inconsistent assignment, and avoid repeating the same mistake in the future
- Naive approach: collect **all decisions** in the trail, and learn a blocking clause implying that at least one of them must be flipped
- Proof-based approach: exploit the information in the trail to generate an explanation for the conflict
  - Generate a new lemma, using boolean resolution, that blocks the current assignment and all those sharing the same reason for inconsistency
  - Resolution rule

Antecedents

$$\frac{\boxed{A \vee B_1 \vee \boxed{x}} \qquad \boxed{A \vee B_2 \vee \neg x}}{\boxed{A \vee B_1 \vee B_2}}$$

Pivot Variable

Resolvent

Example - “1st UIP” learning strategy

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$\boxed{c_6 : \neg A_5 \vee \neg A_6}$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

$\neg A_5 \vee \neg A_6$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4
A_6	c_5

Example - “1st UIP” learning strategy

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$\boxed{c_5 : \neg A_4 \vee A_6 \vee A_{11}}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

$$\frac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}}$$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4
$\boxed{A_6}$	$\boxed{c_5}$

Example - “1st UIP” learning strategy

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$
$c_2 : \neg A_1 \vee A_3 \vee A_9$
$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$
$\boxed{c_4 : \neg A_4 \vee A_5 \vee A_{10}}$
$c_5 : \neg A_4 \vee A_6 \vee A_{11}$
$c_6 : \neg A_5 \vee \neg A_6$
$c_7 : A_1 \vee A_7 \vee \neg A_{12}$
$c_8 : A_1 \vee A_8$
$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

$$\cfrac{\cfrac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}} \qquad \neg A_4 \vee A_5 \vee A_{10}}{\neg A_4 \vee A_{11} \vee A_{10}}$$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
$\boxed{A_5}$	$\boxed{c_4}$
A_6	c_5

Example - “1st UIP” learning strategy

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$c_2 : \neg A_1 \vee A_3 \vee A_9$

$\boxed{c_3 : \neg A_2 \vee \neg A_3 \vee A_4}$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

$$\dfrac{\dfrac{\dfrac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}} \qquad \neg A_4 \vee A_5 \vee A_{10}}{\neg A_4 \vee A_{11} \vee A_{10}} \qquad \neg A_2 \vee \neg A_3 \vee A_4}{A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}$$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
$\boxed{A_4}$	$\boxed{c_3}$
A_5	c_4
A_6	c_5

Example - “1st UIP” learning strategy

Input clauses

$$\boxed{c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}}$$
$$c_2 : \neg A_1 \vee A_3 \vee A_9$$
$$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$$
$$c_4 : \neg A_4 \vee A_5 \vee A_{10}$$
$$c_5 : \neg A_4 \vee A_6 \vee A_{11}$$
$$c_6 : \neg A_5 \vee \neg A_6$$
$$c_7 : A_1 \vee A_7 \vee \neg A_{12}$$
$$c_8 : A_1 \vee A_8$$
$$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
$\boxed{A_2}$	$\boxed{c_1}$
A_4	c_3
A_5	c_4
A_6	c_5

$$\dfrac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}}$$

$$\dfrac{\neg A_5 \vee \neg A_4 \vee A_{11} \qquad \neg A_4 \vee A_5 \vee A_{10}}{\neg A_4 \vee A_{11} \vee A_{10}}$$

$$\dfrac{\neg A_4 \vee A_{11} \vee A_{10} \qquad \neg A_2 \vee \neg A_3 \vee A_4}{A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}$$

$$\dfrac{\neg A_3 \vee A_2 \vee \neg A_{12} \qquad A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_3}$$

Example - “1st UIP” learning strategy

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$
$c_2 : \neg A_1 \vee A_3 \vee A_9$
$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$
$c_4 : \neg A_4 \vee A_5 \vee A_{10}$
$c_5 : \neg A_4 \vee A_6 \vee A_{11}$
$c_6 : \neg A_5 \vee \neg A_6$
$c_7 : A_1 \vee A_7 \vee \neg A_{12}$
$c_8 : A_1 \vee A_8$
$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

$$\dfrac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}}$$

$$\dfrac{\neg A_5 \vee \neg A_4 \vee A_{11} \qquad \neg A_4 \vee A_5 \vee A_{10}}{\neg A_4 \vee A_{11} \vee A_{10}}$$

$$\dfrac{\neg A_4 \vee A_{11} \vee A_{10} \qquad \neg A_2 \vee \neg A_3 \vee A_4}{A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}$$

$$\dfrac{\neg A_3 \vee A_2 \vee \neg A_{12} \qquad A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_3}$$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
A_3	c_2
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4
A_6	c_5

Reached level limit

Example – "decision" learning strategy

Input clauses

$c_1 : \neg A_3 \vee A_2 \vee \neg A_{12}$

$\boxed{c_2 : \neg A_1 \vee A_3 \vee A_9}$

$c_3 : \neg A_2 \vee \neg A_3 \vee A_4$

$c_4 : \neg A_4 \vee A_5 \vee A_{10}$

$c_5 : \neg A_4 \vee A_6 \vee A_{11}$

$c_6 : \neg A_5 \vee \neg A_6$

$c_7 : A_1 \vee A_7 \vee \neg A_{12}$

$c_8 : A_1 \vee A_8$

$c_9 : \neg A_7 \vee \neg A_8 \vee \neg A_{13}$

$$\frac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}}$$

$$\frac{\neg A_5 \vee \neg A_4 \vee A_{11} \qquad \neg A_4 \vee A_5 \vee A_{10}}{\neg A_4 \vee A_{11} \vee A_{10}}$$

$$\frac{\neg A_4 \vee A_{11} \vee A_{10} \qquad \neg A_2 \vee \neg A_3 \vee A_4}{A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}$$

$$\frac{\neg A_3 \vee A_2 \vee \neg A_{12} \qquad A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_3}$$

$$\frac{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_3 \qquad \neg A_1 \vee A_3 \vee A_9}{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_1 \vee A_9}$$

Trail

Lit	Reason
$\neg A_9$	$-$
$\neg A_{10}$	$-$
$\neg A_{11}$	$-$
A_{13}	$-$
A_1	$-$
$\boxed{A_3}$	$\boxed{c_2}$
A_{12}	$-$
A_2	c_1
A_4	c_3
A_5	c_4
A_6	c_5

Example – “decision” learning strategy

Input clauses

$$
\begin{aligned}
c_1 &: \neg A_3 \vee A_2 \vee \neg A_{12} \\
c_2 &: \neg A_1 \vee A_3 \vee A_9 \\
c_3 &: \neg A_2 \vee \neg A_3 \vee A_4 \\
c_4 &: \neg A_4 \vee A_5 \vee A_{10} \\
c_5 &: \neg A_4 \vee A_6 \vee A_{11} \\
c_6 &: \neg A_5 \vee \neg A_6 \\
c_7 &: A_1 \vee A_7 \vee \neg A_{12} \\
c_8 &: A_1 \vee A_8 \\
c_9 &: \neg A_7 \vee \neg A_8 \vee \neg A_{13}
\end{aligned}
$$

$$\frac{\neg A_5 \vee \neg A_6 \qquad \neg A_4 \vee \neg A_6 \vee A_{11}}{\neg A_5 \vee \neg A_4 \vee A_{11}}$$

$$\frac{\neg A_5 \vee \neg A_4 \vee A_{11} \qquad \neg A_4 \vee A_5 \vee A_{10}}{\neg A_4 \vee A_{11} \vee A_{10}}$$

$$\frac{\neg A_4 \vee A_{11} \vee A_{10} \qquad \neg A_2 \vee \neg A_3 \vee A_4}{A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}$$

$$\frac{\neg A_3 \vee A_2 \vee \neg A_{12} \qquad A_{11} \vee A_{10} \vee \neg A_2 \vee \neg A_3}{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_3}$$

$$\frac{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_3 \qquad \neg A_1 \vee A_3 \vee A_9}{\color{blue}{\neg A_{12} \vee A_{11} \vee A_{10} \vee \neg A_1 \vee A_9}}$$

Trail

Lit	Reason
$\color{blue}{\neg A_9}$	$-$
$\color{blue}{\neg A_{10}}$	$-$
$\color{blue}{\neg A_{11}}$	$-$
A_{13}	$-$
$\color{blue}{A_1}$	$-$
A_3	c_2
$\color{blue}{A_{12}}$	$-$
A_2	c_1
A_4	c_3
A_5	c_4
A_6	c_5

Outline

Introduction

CDCL-based SAT solvers

The DPLL(T) architecture

Some relevant T-solvers

Combination of theories

The lazy approach to SMT

- Deciding the satisfiability of φ modulo $\mathcal{T}$ can be reduced to deciding $\mathcal{T}$-satisfiability of **conjunctions (sets) of constraints**
  - Can exploit efficient decision procedures for sets of constraints, existing for many important theories

- **Naive approach:** convert φ to an equivalent φ' in disjunctive normal form (DNF), and check each conjunction separately

- Main idea of **lazy SMT**: use an efficient SAT solver to enumerate conjuncts without computing the DNF explicitly

A basic approach

- **Offline** lazy SMT

```
F = CNF_bool( φ )
while true:
    res, M = check_SAT(F)
    if res == true:
        M' = to_T(M)
        res = check_T(M')
        if res == true:
            return SAT
        else:
            F += !M
    else:
        return UNSAT
```

A basic approach

- **Offline** lazy SMT

```
F = CNF_bool( φ )
while true:
   res, M = check_SAT(F)
   if res == true:
      M' = to_T(M)
      res = check_T(M')
      if res == true:
         return SAT
      else:
         F += !M
   else:
      return UNSAT
```

Boolean reasoning

A basic approach

- **Offline** lazy SMT

```
F = CNF_bool( φ )
while true:
   res, M = check_SAT(F)
   if res == true:
      M' = to_T(M)
      res = check_T(M')
      if res == true:
         return SAT
      else:
         F += !M
   else:
      return UNSAT
```

Boolean reasoning

Theory reasoning

A basic approach

- **Offline** lazy SMT

```
F = CNF_bool( φ )
while true:
   res, M = check_SAT(F)
   if res == true:
      M' = to_T(M)
      res = check_T(M')
      if res == true:
         return SAT
      else:
         F += !M
   else:
      return UNSAT
```

Boolean reasoning

Theory reasoning

Block bad solutions

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2
\end{array}
$$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
& A_1 \vee \color{red}{P_1} \\
& \neg P_2 \vee \color{red}{A_2} \\
& \color{red}{\neg A_3} \vee \neg P_2 \\
& \color{red}{\neg A_4} \vee \neg P_1 \\
& \color{red}{P_1} \vee A_3 \\
& \color{red}{A_5} \vee \neg P_1 \\
& \color{red}{P_1} \vee \color{red}{A_6} \vee \neg P_2
\end{array}
$$

$$M = \{P_1, P_2, \neg A_1, A_2, \neg A_3, \neg A_4, A_5, A_6\}$$

$$
\begin{aligned}
M' = \{&\neg(2x_2 - x_3 > 2), (x_1 - x_5 \leq 1), \neg(3x_1 - 2x_2 \leq 3), \\
&\neg(3x_1 - x_3 \leq 6), (x_2 - x_4 \leq 6), (x_3 = 3x_5 + 4)\}
\end{aligned}
$$

Example

$$
\begin{array}{llll}
\varphi & \stackrel{\text{def}}{=} & \varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & & A_1 \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & & \neg P_2 \vee A_2 \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & & \neg A_3 \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & & \neg A_4 \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & & P_1 \vee A_3 \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & & A_5 \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & & P_1 \vee A_6 \vee \neg P_2
\end{array}
$$

$$M = \{P_1, P_2, \neg A_1, A_2, \neg A_3, \neg A_4, A_5, A_6\}$$

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

$$\boxed{\neg(3x_1 - 3x_5 - 4 \leq 6) \mapsto \neg(x_1 - x_5 \leq 10/3) \mapsto (x_1 - x_5 > 10/3)}$$

UNSAT $\rightarrow$ add $\neg M$ and continue

DPLL(T)

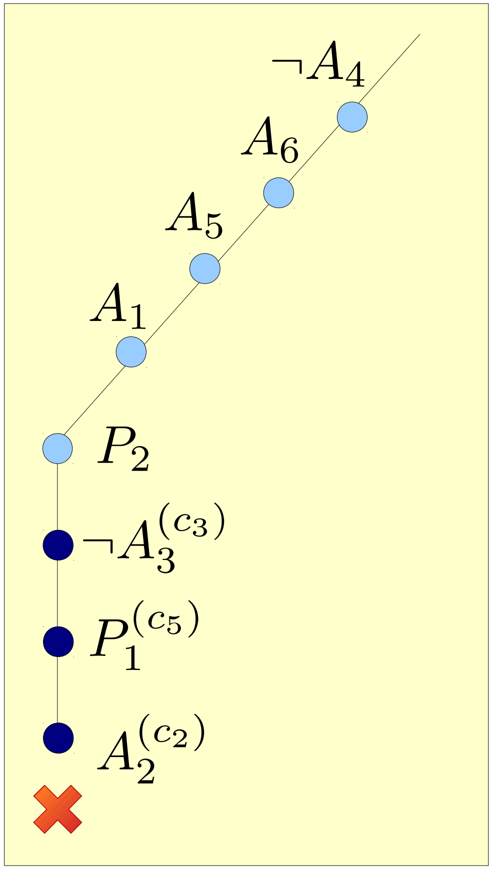
- **Online** approach to lazy SMT
- **Tight integration** between a CDCL-like SAT solver ("DPLL") and the decision procedure for *T* ("T-solver"), based on:
  - *T*-driven backjumping and learning
  - Early pruning
  - *T*-solver incrementality
  - *T*-propagation
  - Filtering of assignments to check
  - Creation of new *T*-atoms and *T*-lemmas "on-demand"
  - ...

T-backjumping and *T*-learning

- When unsat, *T*-solver can produce **reason for inconsistency**
  - ***T*-conflict set**: inconsistent subset of the input constraints
- *T*-conflict clause given as input to the CDCL **conflict analysis**
  - Drives non-chronological backtracking (backjumping)
  - Can be learned by the SAT solver
- The less redundant the *T*-conflict set, the more search is saved
  - Ideally, should be **minimal** (irredundant)
    - Removing any element makes the set consistent
  - But for some theories might be expensive to achieve
    - Trade-off between size and cost

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2
\end{array}
$$

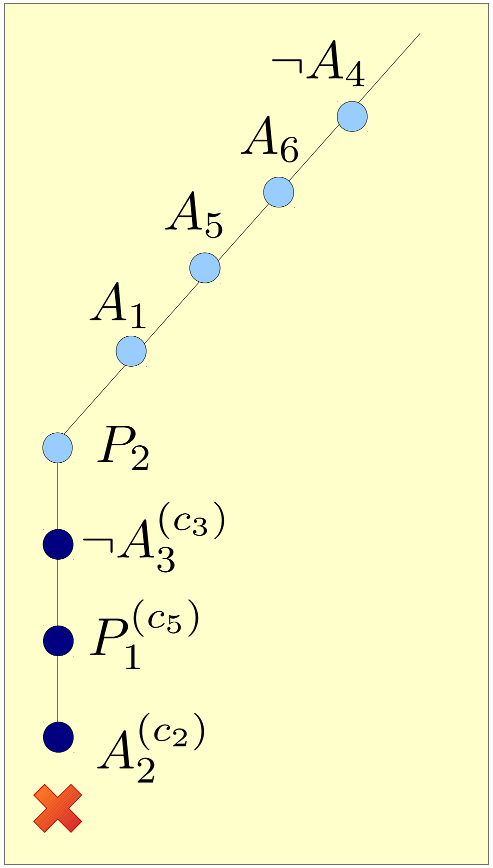


$$M = [\neg A_4, A_6, A_5, A_1, P_2, \neg A_3, P_1, A_2]$$

$$
\begin{array}{ll}
M' = \{ & \neg(3x_1 - x_3 \leq 6), (x_3 = 3x_5 + 4), (x_2 - x_4 \leq 6), \\
 & \neg(2x_2 - x_3 > 2), \neg(3x_1 - 2x_2 \leq 3), (x_1 - x_5 \leq 1)\}
\end{array}
$$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
A_1 \vee P_1 \\
\neg P_2 \vee A_2 \\
\neg A_3 \vee \neg P_2 \\
\neg A_4 \vee \neg P_1 \\
P_1 \vee A_3 \\
A_5 \vee \neg P_1 \\
P_1 \vee A_6 \vee \neg P_2
\end{array}
$$



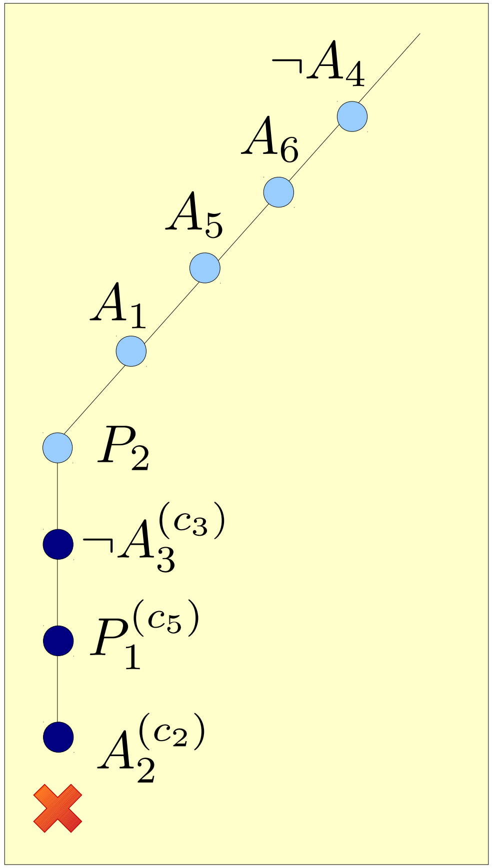
$$M = [\neg A_4, A_6, A_5, A_1, P_2, \neg A_3, P_1, A_2]$$

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

T-conflict set

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \lor P_1 & \\
c_2: & \neg P_2 \lor (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \lor \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \lor \neg P_1 & \\
c_5: & P_1 \lor (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \lor \neg P_1 & \\
c_7: & P_1 \lor (x_3 = 3x_5 + 4) \lor \neg P_2 &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
& A_1 \lor P_1 \\
& \neg P_2 \lor A_2 \\
& \neg A_3 \lor \neg P_2 \\
& \neg A_4 \lor \neg P_1 \\
& P_1 \lor A_3 \\
& A_5 \lor \neg P_1 \\
& P_1 \lor A_6 \lor \neg P_2
\end{array}
$$

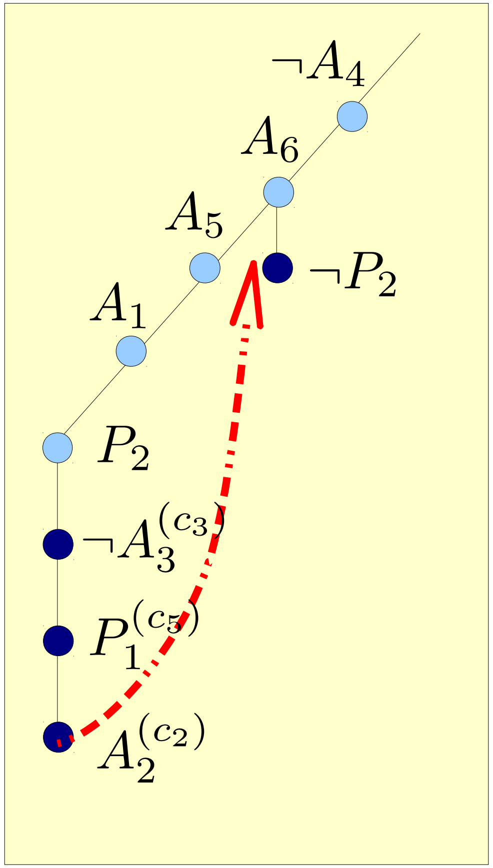


Conflict analysis:

$$
\frac{\overbrace{A_4 \lor \neg A_6 \lor \neg A_2}^{\mathcal{T}\text{-conflict clause}} \qquad \overbrace{\neg P_2 \lor A_2}^{c_2}}{A_4 \lor \neg A_6 \lor \neg P_2}
$$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \varphi^{\text{Bool}} \stackrel{\text{def}}{=} \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & A_1 \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \neg P_2 \vee A_2 \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \neg A_3 \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \neg A_4 \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & P_1 \vee A_3 \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & A_5 \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & P_1 \vee A_6 \vee \neg P_2 \\
c_8: & A_4 \vee \neg A_6 \vee \neg P_2 &
\end{array}
$$

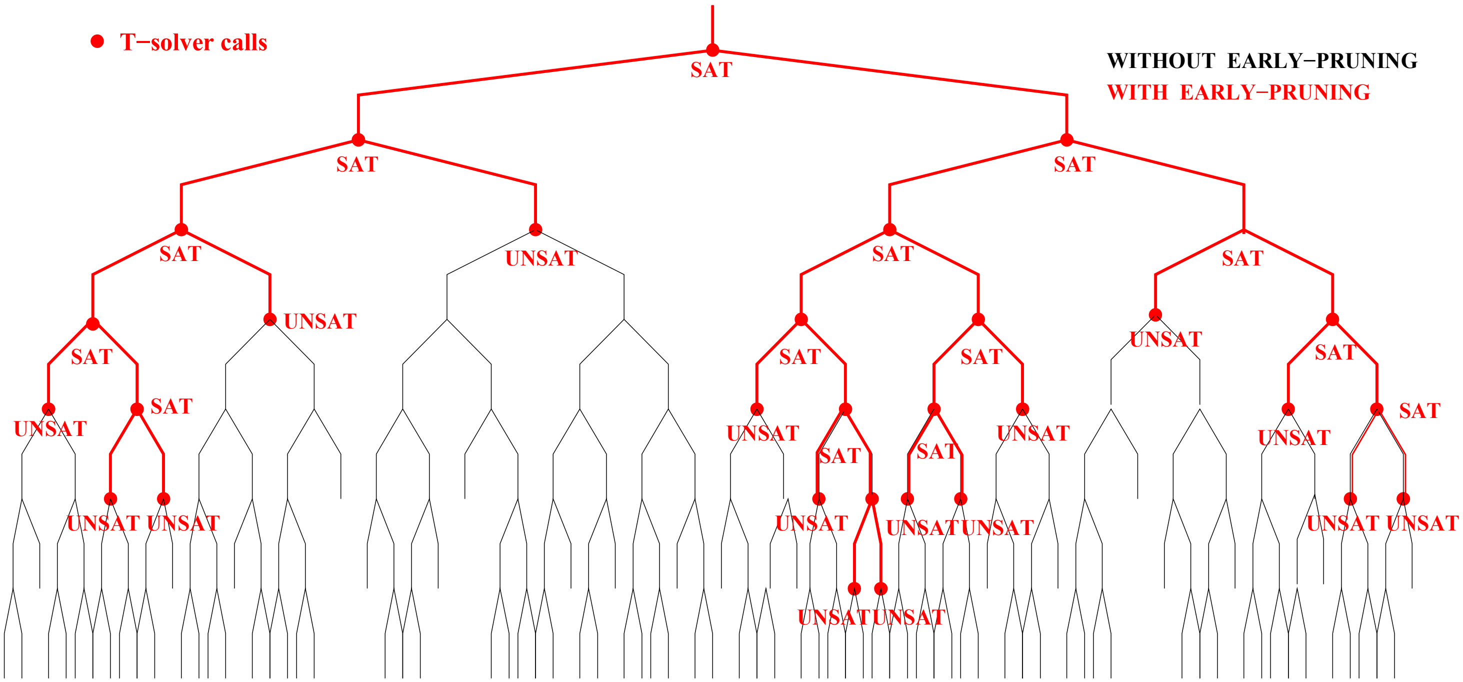


Conflict analysis:

$$
\frac{\overbrace{A_4 \vee \neg A_6 \vee \neg A_2}^{\mathcal{T}\text{-conflict clause}} \qquad \overbrace{\neg P_2 \vee A_2}^{c_2}}{A_4 \vee \neg A_6 \vee \neg P_2}
$$

Early pruning

- Invoke *T*-solver on **intermediate assignments**, during the CDCL search
  - If **unsat** is returned, can backtrack immediately
- **Advantage:** can drastically prune the search tree
- **Drawback:** possibly many useless (expensive) *T*-solver calls



Early pruning

- Different strategies to call *T*-solver
  - Eagerly, every time a new atom is assigned
  - After every round of BCP
  - Heuristically, based on some statistics (e.g. effectivenes, …)

- **No need of a conclusive answer** during early pruning calls
  - Can apply **approximate checks**
  - Trade effectiveness for efficiency

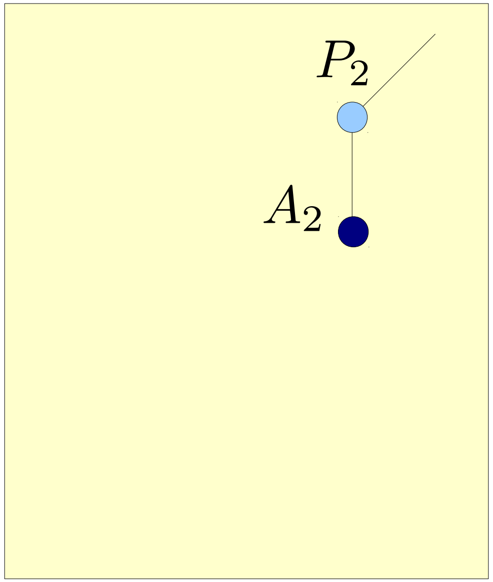
- Example: on linear integer arithmetic, solve only the real relaxation during early pruning calls

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2
\end{array}
$$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
& A_1 \vee P_1 \\
& \neg P_2 \vee A_2 \\
& \neg A_3 \vee \neg P_2 \\
& \neg A_4 \vee \neg P_1 \\
& P_1 \vee A_3 \\
& A_5 \vee \neg P_1 \\
& P_1 \vee A_6 \vee \neg P_2
\end{array}
$$

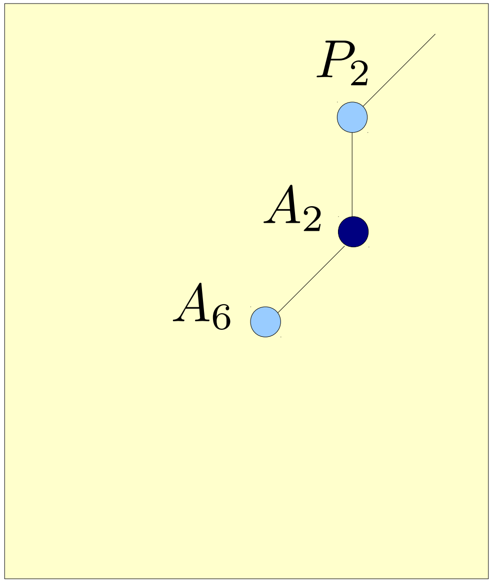


$$M = [P_2, A_2]$$

SAT

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \varphi^{\text{Bool}} \stackrel{\text{def}}{=} \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & A_1 \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \neg P_2 \vee A_2 \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \neg A_3 \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \neg A_4 \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & P_1 \vee A_3 \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & A_5 \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & P_1 \vee A_6 \vee \neg P_2
\end{array}
$$

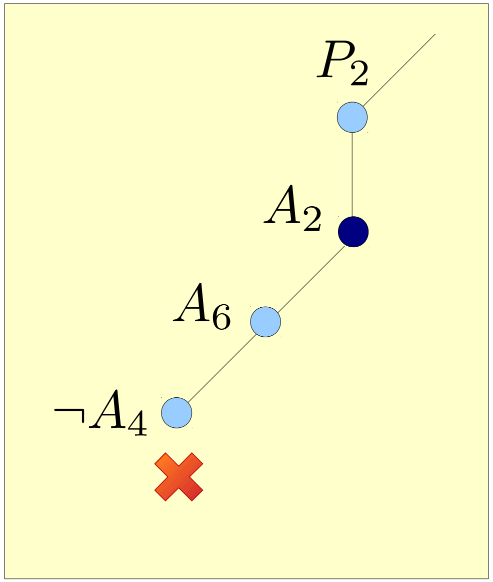


$$M = [P_2, A_2, A_6]$$

SAT

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \varphi^{\text{Bool}} \stackrel{\text{def}}{=} \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & A_1 \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \le 1) & \neg P_2 \vee A_2 \\
c_3: & \neg(3x_1 - 2x_2 \le 3) \vee \neg P_2 & \neg A_3 \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \le 6) \vee \neg P_1 & \neg A_4 \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \le 3) & P_1 \vee A_3 \\
c_6: & (x_2 - x_4 \le 6) \vee \neg P_1 & A_5 \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & P_1 \vee A_6 \vee \neg P_2
\end{array}
$$



$$M = [P_2, A_2, A_6, \neg A_4]$$

UNSAT

T-conflict = $\{\neg(3x_1 - x_3 \le 6), (x_3 = 3x_5 + 4), (x_1 - x_5 \le 1)\}$

T-solver incrementality

- With early pruning, *T*-solvers invoked **very frequently** on **similar** problems
  - **Stack** of constraints (the assignment stack of CDCL) **incrementally updated**
- **Incrementality:** when a new constraint is added, no need to redo all the computation “from scratch”
- **Backtrackability:** support cheap (stack-based) removal of constraints without “resetting” the internal state
- **Crucial for efficiency**
  - Distinguishing feature for effective integration in DPLL(T)

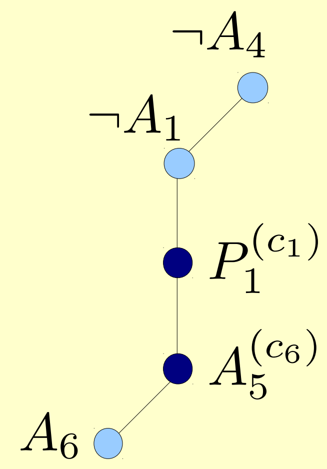
T-propagation

- T-solvers might support **deduction** of unassigned constraints
  - If early pruning check on M returns sat, *T*-solver might also return a set D of unassigned atoms such that $M \models_{\mathcal{T}} l$ for all $l \in D$
- **T-propagation:** add each such l to the CDCL stack
  - As if BCP was applied to the (*T*-valid) clause $\neg M \vee l$ (*T*-reason)
  - But **do not compute the *T*-reason clause** explicitly yet
- **Lazy explanation:** compute *T*-reason clause only if needed during conflict analysis
  - Like *T*-conflicts, the less redundant the better

Example

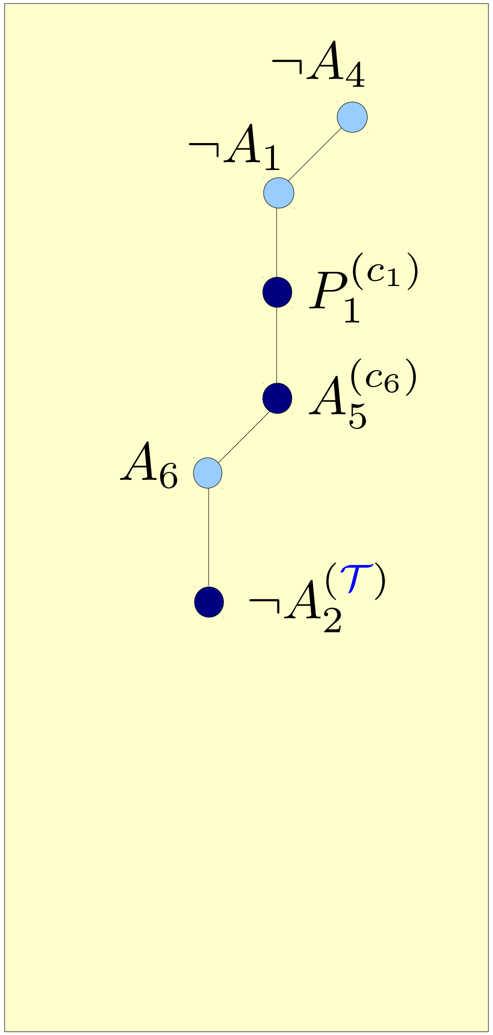
$$
\begin{array}{lll@{\qquad}l}
\varphi & \stackrel{\text{def}}{=} & & \varphi^{\text{Bool}} \stackrel{\text{def}}{=} \\
c_1 : & (2x_2 - x_3 > 2) \vee P_1 & & A_1 \vee P_1 \\
c_2 : & \neg P_2 \vee (x_1 - x_5 \leq 1) & & \neg P_2 \vee A_2 \\
c_3 : & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & & \neg A_3 \vee \neg P_2 \\
c_4 : & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & & \neg A_4 \vee \neg P_1 \\
c_5 : & P_1 \vee (3x_1 - 2x_2 \leq 3) & & P_1 \vee A_3 \\
c_6 : & (x_2 - x_4 \leq 6) \vee \neg P_1 & & A_5 \vee \neg P_1 \\
c_7 : & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & & P_1 \vee A_6 \vee \neg P_2 \\
c_8 : & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & & P_2 \vee A_7 \vee A_8 \\
 & (x_3 + x_5 - 4x_1 \geq 0) & &
\end{array}
$$

$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6]$$



Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \le 1) & \\
c_3: & \neg(3x_1 - 2x_2 \le 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \le 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \le 3) & \\
c_6: & (x_2 - x_4 \le 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & \\
c_8: & P_2 \vee (2x_2 - 3x_1 \ge 5) \vee & \\
 & (x_3 + x_5 - 4x_1 \ge 0) &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
A_1 \vee P_1 & \\
\neg P_2 \vee A_2 & \\
\neg A_3 \vee \neg P_2 & \\
\neg A_4 \vee \neg P_1 & \\
P_1 \vee A_3 & \\
A_5 \vee \neg P_1 & \\
P_1 \vee A_6 \vee \neg P_2 & \\
P_2 \vee A_7 \vee A_8 &
\end{array}
$$



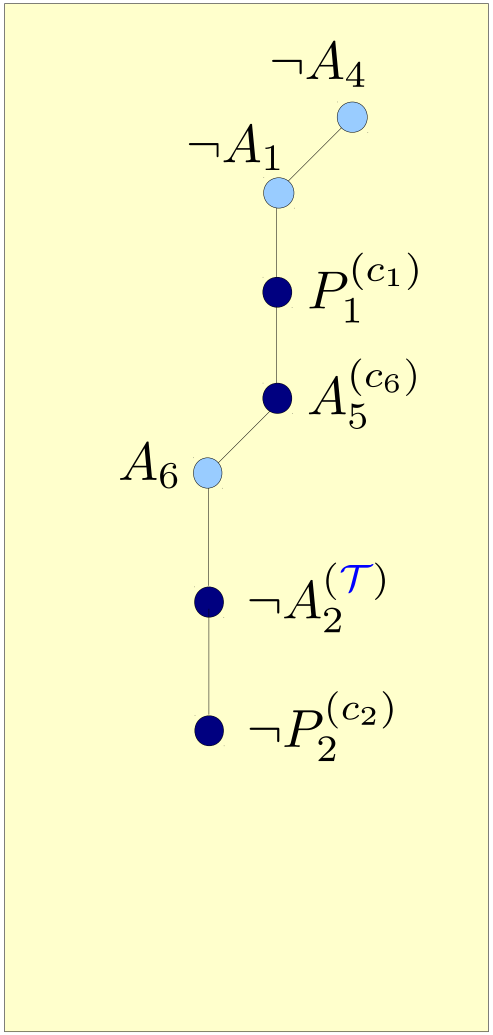
$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6]$$

$$
\cfrac{\cfrac{\neg(3x_1 - x_3 \le 6) \qquad (x_3 = 3x_5 + 4)}{\neg(3x_1 - 3x_5 \le 10)}}{\neg(x_1 - x_5 \le 1) \equiv \neg A_2}
$$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & \\
c_8: & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & \\
 & (x_3 + x_5 - 4x_1 \geq 0) &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2 \\
 & P_2 \vee A_7 \vee A_8 \\
 &
\end{array}
$$

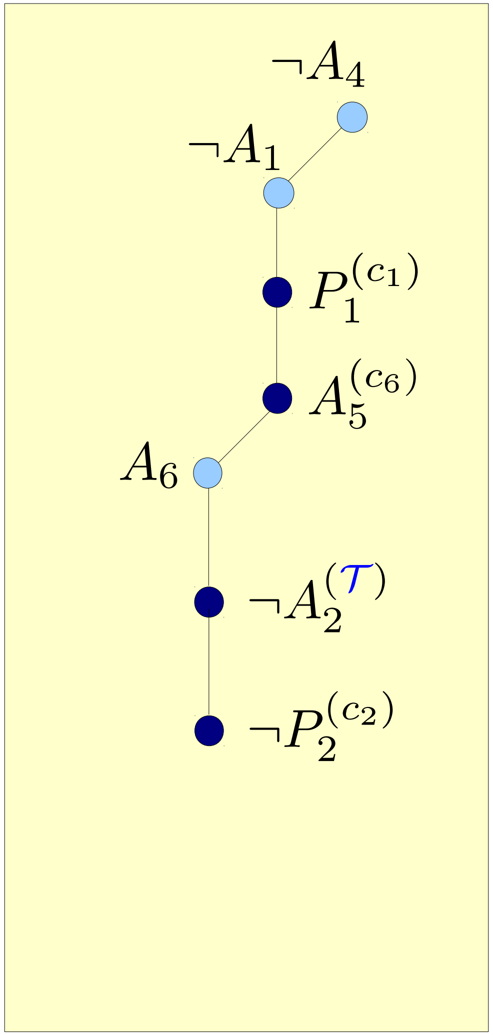
$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2]$$



Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1 : & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2 : & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3 : & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4 : & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5 : & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6 : & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7 : & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & \\
c_8 : & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & \\
 & (x_3 + x_5 - 4x_1 \geq 0) &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2 \\
 & P_2 \vee A_7 \vee A_8
\end{array}
$$

$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2]$$

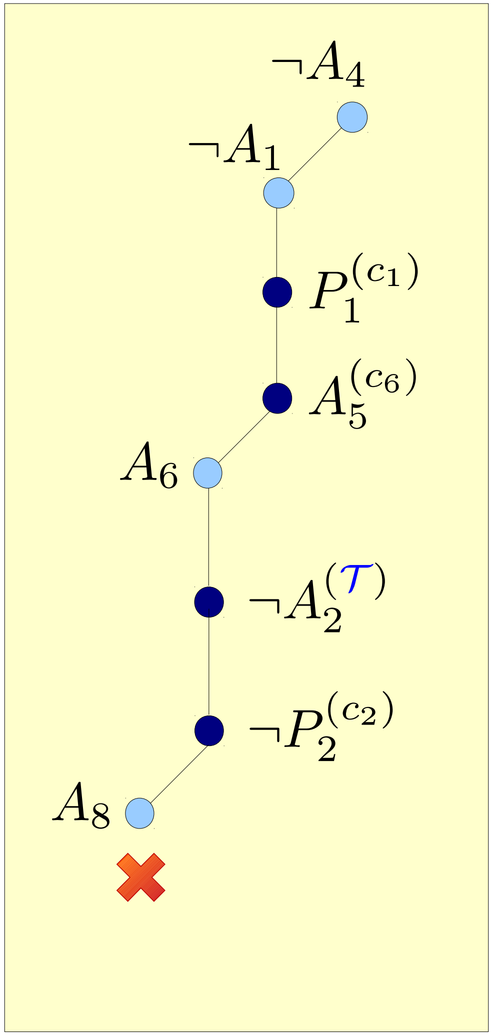


Example

$$
\begin{array}{llll}
\varphi & \stackrel{\text{def}}{=} & \varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & & A_1 \vee {\color{red}P_1} \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & & {\color{red}\neg P_2} \vee A_2 \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & & \neg A_3 \vee {\color{red}\neg P_2} \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & & {\color{red}\neg A_4} \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & & {\color{red}P_1} \vee A_3 \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & & {\color{red}A_5} \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & & {\color{red}P_1} \vee {\color{red}A_6} \vee {\color{red}\neg P_2} \\
c_8: & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & & P_2 \vee A_7 \vee A_8 \\
 & (x_3 + x_5 - 4x_1 \geq 0) & &
\end{array}
$$

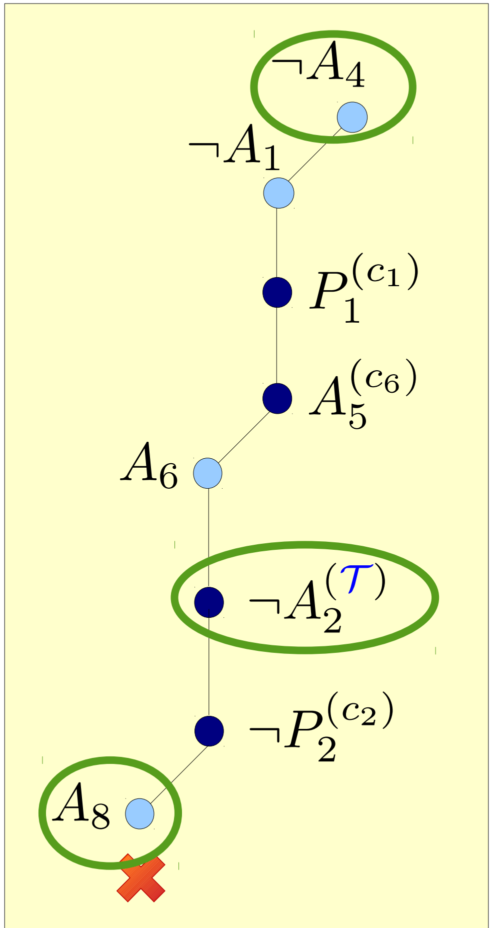
$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2, A_8]$$

$$
\dfrac{\dfrac{\neg(3x_1 - x_3 \leq 6) \qquad \neg(x_1 - x_5 \leq 1)}{\neg(-x_3 + 3x_5 \leq 3)} \qquad (x_3 + x_5 - 4x_1 \geq 0)}{\perp}
$$



Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & \\
c_8: & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & \\
 & (x_3 + x_5 - 4x_1 \geq 0) &
\end{array}
\qquad
\begin{array}{l}
\varphi^{\text{Bool}} \stackrel{\text{def}}{=} \\
A_1 \vee {\color{red}P_1} \\
{\color{red}\neg P_2} \vee A_2 \\
\neg A_3 \vee {\color{red}\neg P_2} \\
{\color{red}\neg A_4} \vee \neg P_1 \\
{\color{red}P_1} \vee A_3 \\
{\color{red}A_5} \vee \neg P_1 \\
{\color{red}P_1} \vee {\color{red}A_6} \vee {\color{red}\neg P_2} \\
P_2 \vee A_7 \vee A_8
\end{array}
$$



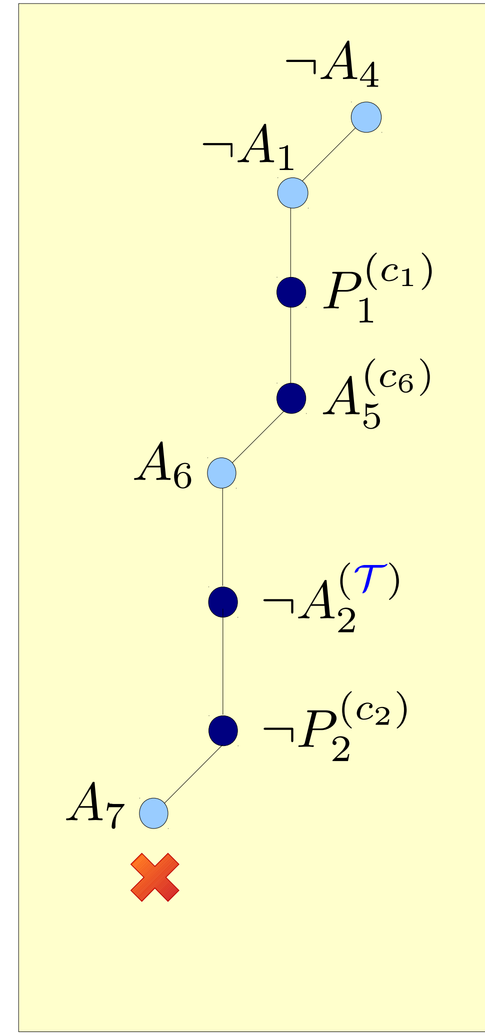
$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2, A_8]$$

$$
\cfrac{\cfrac{\neg(3x_1 - x_3 \leq 6) \qquad \neg(x_1 - x_5 \leq 1)}{\neg(-x_3 + 3x_5 \leq 3)} \qquad (x_3 + x_5 - 4x_1 \geq 0)}{\perp}
$$

Conflict analysis → compute T-reason for $\neg A_2$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \lor P_1 \\
c_2: & \neg P_2 \lor (x_1 - x_5 \le 1) \\
c_3: & \neg(3x_1 - 2x_2 \le 3) \lor \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \le 6) \lor \neg P_1 \\
c_5: & P_1 \lor (3x_1 - 2x_2 \le 3) \\
c_6: & (x_2 - x_4 \le 6) \lor \neg P_1 \\
c_7: & P_1 \lor (x_3 = 3x_5 + 4) \lor \neg P_2 \\
c_8: & P_2 \lor (2x_2 - 3x_1 \ge 5) \lor \\
 & (x_3 + x_5 - 4x_1 \ge 0)
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \lor P_1 \\
 & \neg P_2 \lor A_2 \\
 & \neg A_3 \lor \neg P_2 \\
 & \neg A_4 \lor \neg P_1 \\
 & P_1 \lor A_3 \\
 & A_5 \lor \neg P_1 \\
 & P_1 \lor A_6 \lor \neg P_2 \\
 & P_2 \lor A_7 \lor A_8 \\
 &
\end{array}
$$



$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2, A_7]$$

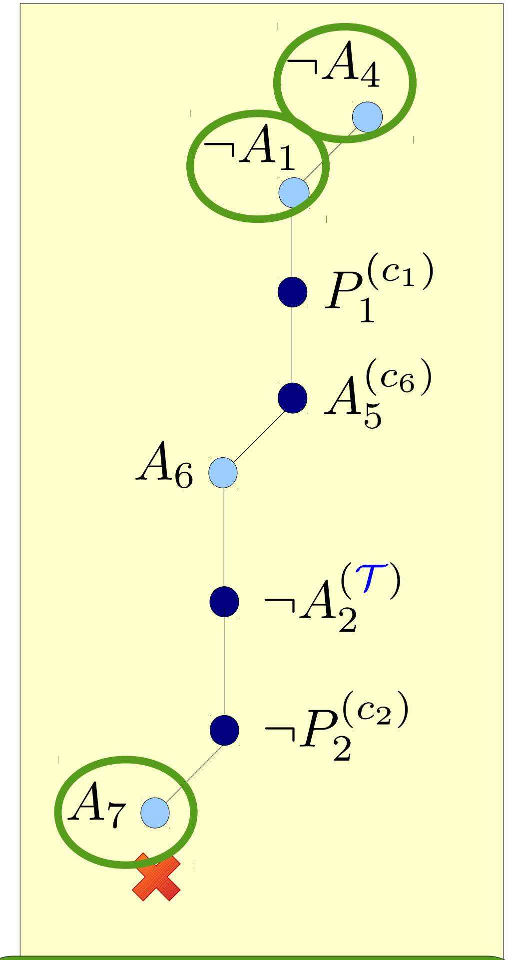
$$
\dfrac{\dfrac{\neg(3x_1 - x_3 \le 6) \qquad \neg(2x_2 - x_3 > 2)}{\neg(3x_1 - 2x_2 \le 4)} \qquad (2x_2 - 3x_1 \ge 5)}{\bot}
$$

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & \\
c_3: & \neg(3x_1 - 2x_2 \leq 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \leq 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & \\
c_8: & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & \\
 & (x_3 + x_5 - 4x_1 \geq 0) &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2 \\
 & P_2 \vee A_7 \vee A_8
\end{array}
$$

$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2, A_7]$$

$$
\frac{\dfrac{\neg(3x_1 - x_3 \leq 6) \qquad \neg(2x_2 - x_3 > 2)}{\neg(3x_1 - 2x_2 \leq 4)} \qquad (2x_2 - 3x_1 \geq 5)}{\bot}
$$



$\neg A_2$ not involved in conflict analysis $\rightarrow$ *no need to compute T-reason*

Filtering of assignments

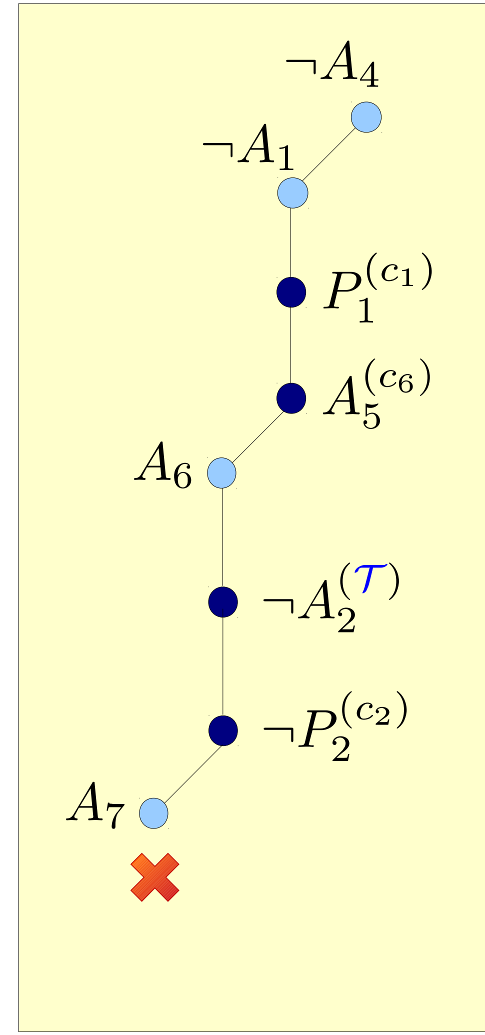
- Remove **unnecessary literals** from current assignment M
  - **Irrelevant** literals: l s.t. $M \setminus \{l\} \models \varphi$ (φ arbitary, not CNF)
  - **Ghost** literals: l occurs only in clauses satisfied by $M \setminus \{l\}$
  - **Pure** literals: $\neg l \in M$ and l occurs only positively in φ
    - Note: this is **not** the pure-literal rule of SAT!
- **Pros:**
  - reduce effort for *T*-solver
  - increases the chances of finding a solution
- **Cons:**
  - may weaken the effect of early pruning (esp. with *T*-propagation)
  - may introduce overhead in SAT search
- Typically used for expensive theories

Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \le 1) \\
c_3: & \neg(3x_1 - 2x_2 \le 3) \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 \le 6) \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \le 3) \\
c_6: & (x_2 - x_4 \le 6) \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 \\
c_8: & P_2 \vee (2x_2 - 3x_1 \ge 5) \vee \\
 & (x_3 + x_5 - 4x_1 \ge 0)
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2 \\
 & P_2 \vee A_7 \vee A_8 \\
 &
\end{array}
$$

$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2, A_7]$$

$$
\frac{\dfrac{\neg(3x_1 - x_3 \le 6) \qquad \neg(2x_2 - x_3 > 2)}{\neg(3x_1 - 2x_2 \le 4)} \qquad (2x_2 - 3x_1 \ge 5)}{\bot}
$$



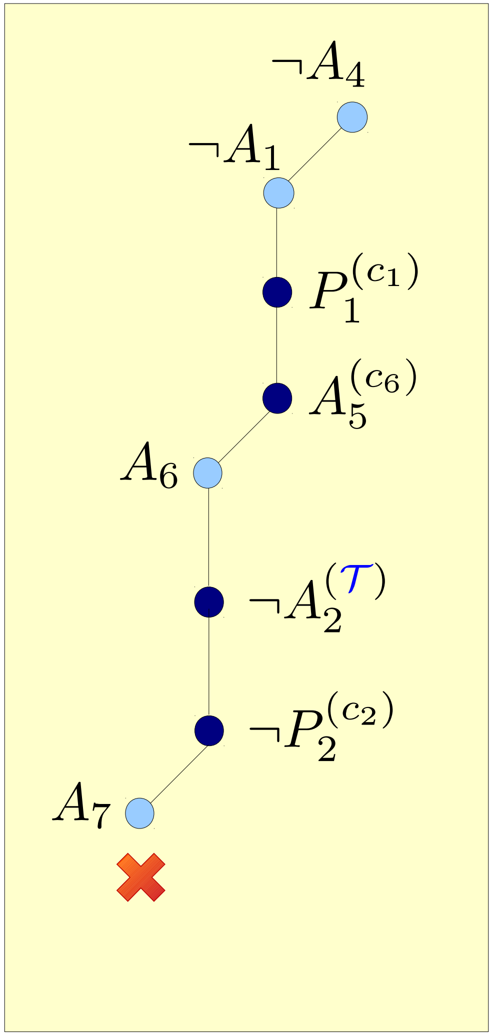
Example

$$
\begin{array}{llll}
\varphi & \stackrel{\text{def}}{=} & & \varphi^{\text{Bool}} \stackrel{\text{def}}{=} \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & & A_1 \vee P_1 \\
c_2: & \neg P_2 \vee (x_1 - x_5 \leq 1) & & \neg P_2 \vee A_2 \\
c_3: & \neg(3x_1 - 2x_2 & & \neg A_3 \vee \neg P_2 \\
c_4: & \neg(3x_1 - x_3 & & \neg A_4 \vee \neg P_1 \\
c_5: & P_1 \vee (3x_1 - 2x_2 \leq 3) & & P_1 \vee A_3 \\
c_6: & (x_2 - x_4 \leq 6) \vee \neg P_1 & & A_5 \vee \neg P_1 \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & & P_1 \vee A_6 \vee \neg P_2 \\
c_8: & P_2 \vee (2x_2 - 3x_1 \geq 5) \vee & & P_2 \vee A_7 \vee A_8 \\
 & (x_3 + x_5 - 4x_1 \geq 0) & &
\end{array}
$$

Ghost!

$$M = [\neg A_4, \neg A_1, P_1, A_5, A_6, \neg A_2, \neg P_2, A_7]$$

$$
\cfrac{\cfrac{\neg(3x_1 - x_3 \leq 6) \qquad \neg(2x_2 - x_3 > 2)}{\neg(3x_1 - 2x_2 \leq 4)} \qquad (2x_2 - 3x_1 \geq 5)}{\perp}
$$

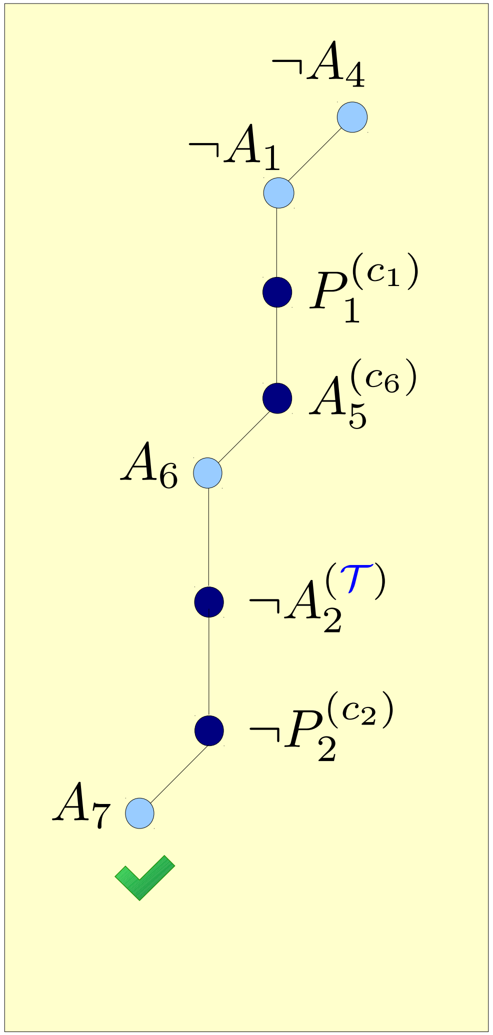


Example

$$
\begin{array}{lll}
\varphi & \stackrel{\text{def}}{=} & \\
c_1: & (2x_2 - x_3 > 2) \vee P_1 & \\
c_2: & \neg P_2 \vee (x_1 - x_5 \le 1) & \\
c_3: & \neg(3x_1 - 2x_2 \le 3) \vee \neg P_2 & \\
c_4: & \neg(3x_1 - x_3 \le 6) \vee \neg P_1 & \\
c_5: & P_1 \vee (3x_1 - 2x_2 \le 3) & \\
c_6: & (x_2 - x_4 \le 6) \vee \neg P_1 & \\
c_7: & P_1 \vee (x_3 = 3x_5 + 4) \vee \neg P_2 & \\
c_8: & P_2 \vee (2x_2 - 3x_1 \ge 5) \vee & \\
 & (x_3 + x_5 - 4x_1 \ge 0) &
\end{array}
\qquad
\begin{array}{ll}
\varphi^{\text{Bool}} & \stackrel{\text{def}}{=} \\
 & A_1 \vee P_1 \\
 & \neg P_2 \vee A_2 \\
 & \neg A_3 \vee \neg P_2 \\
 & \neg A_4 \vee \neg P_1 \\
 & P_1 \vee A_3 \\
 & A_5 \vee \neg P_1 \\
 & P_1 \vee A_6 \vee \neg P_2 \\
 & P_2 \vee A_7 \vee A_8 \\
 &
\end{array}
$$

$$M = [\neg A_4, \qquad , P_1, A_5, A_6, \neg A_2, \neg P_2, A_7]$$

SAT



T-atoms and *T*-lemmas on demand

- Some *T*-solvers might need to perform **internal case splits** to decide satisfiability
  - Example: linear integer arithmetic
    $$(x - 3y \leq 0), (y - 2x \leq 0), (x + 3y \leq 3) \mapsto \begin{cases} \text{case } (y \leq 0) \mapsto \bot \\ \text{case } (y \geq 1) \mapsto \bot \end{cases}$$
- **Splitting on-demand**: use the SAT solver for case splits
  - Encode splits as *T*-valid clauses (*T*-lemmas) with fresh *T*-atoms
  - Generated on-the-fly during search, when needed
  - **Benefits:** reuse the efficient SAT search
    - simplify the implementation
    - exploit advanced search-space exploration techniques (backjumping, learning, restarts, ...)
  - **Potential drawback:** "pollute" the SAT search

T-atoms and *T*-lemmas on demand

- *T*-solver can now return **unknown** also for complete checks
  - In this case, it must also produce one or more *T*-lemmas
  - SAT solver learns the lemmas and continues searching
  - eventually, *T*-solver **can decide** sat/unsat
- **Termination** issues
  - If SAT solver drops lemmas, might get into an infinite loop
    - similar to the Boolean case (and the "basic" SMT case), similar solution (e.g. monotonically increase # of kept lemmas)
  - *T*-solver can generate an infinite number of new *T*-atoms!
    - For several theories (e.g. linear integer arithmetic, arrays) enough to draw new *T*-atoms **from a finite set** (dependent on the input problem)

T-solver interface example

```
class TheorySolver {
    bool tell_atom(Var boolatom, Expr tatom);

    void new_decision_level();
    void backtrack(int level);

    void assume(Lit l);
    lbool check(bool approx);

    void get_conflict(LitList &out);

    Lit get_next_implied();
    bool get_explanation(Lit implied, LitList &out);

    bool get_lemma(LitList &out);

    Expr get_value(Expr term);
};
```

DPLL(T) example

```
def DPLL-T():
   while True:
      conflict = False
      if unit_propagation():
         res = T.check(!all_assigned())
         if res == False: conflict = True
         elif res == True: conflict = theory_propagation()
         elif learn_T_lemmas(): continue
         elif !all_assigned(): decide()
         else:
            build_model()
            return SAT
      else: conflict = True
      if conflict:
         lvl, cls = conflict_analysis()
         if lvl < 0: return UNSAT
         else:
            backtrack(lvl)
            learn(cls)
```

DPLL(T) example

```
def DPLL-T():
    while True:
        conflict = False
        if unit_propagation()
            res = T.check(!all_assigned())
            if res == False: conflict = True
            elif res == True: conflict = theory_propagation()
            elif learn_T_lemmas(): continue
            elif !all_assigned(): decide()
            else:
                build_model()
                return SAT
        else: conflict = True
        if conflict:
            lvl, cls = conflict_analysis()
            if lvl < 0: return UNSAT
            else:
                backtrack(lvl)
                learn(cls)
```

call `T.assume(lit)`

call `T.get_next_implied()`

call `T.get_lemma()`

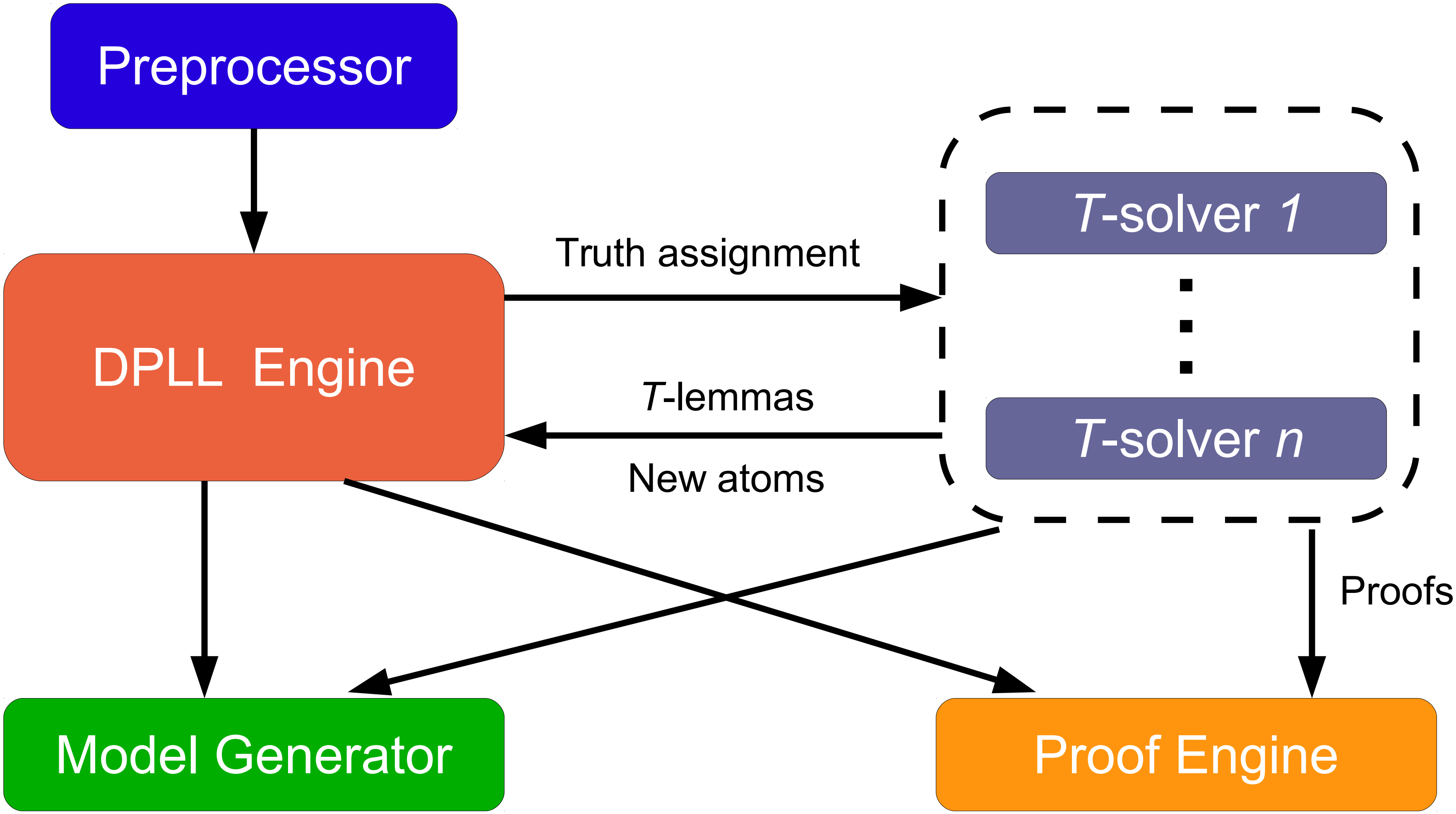
call `T.new_decision_level()`
`T.assume(lit)`

call `T.get_value(e)`

call `T.get_conflict(c)`
`T.get_explanation(l, e)`

call `T.backtrack(lvl)`

An example lazy SMT architecture (MathSAT)



Outline

Introduction

CDCL-based SAT solvers

The DPLL(T) architecture

Some relevant T-solvers

Combination of theories

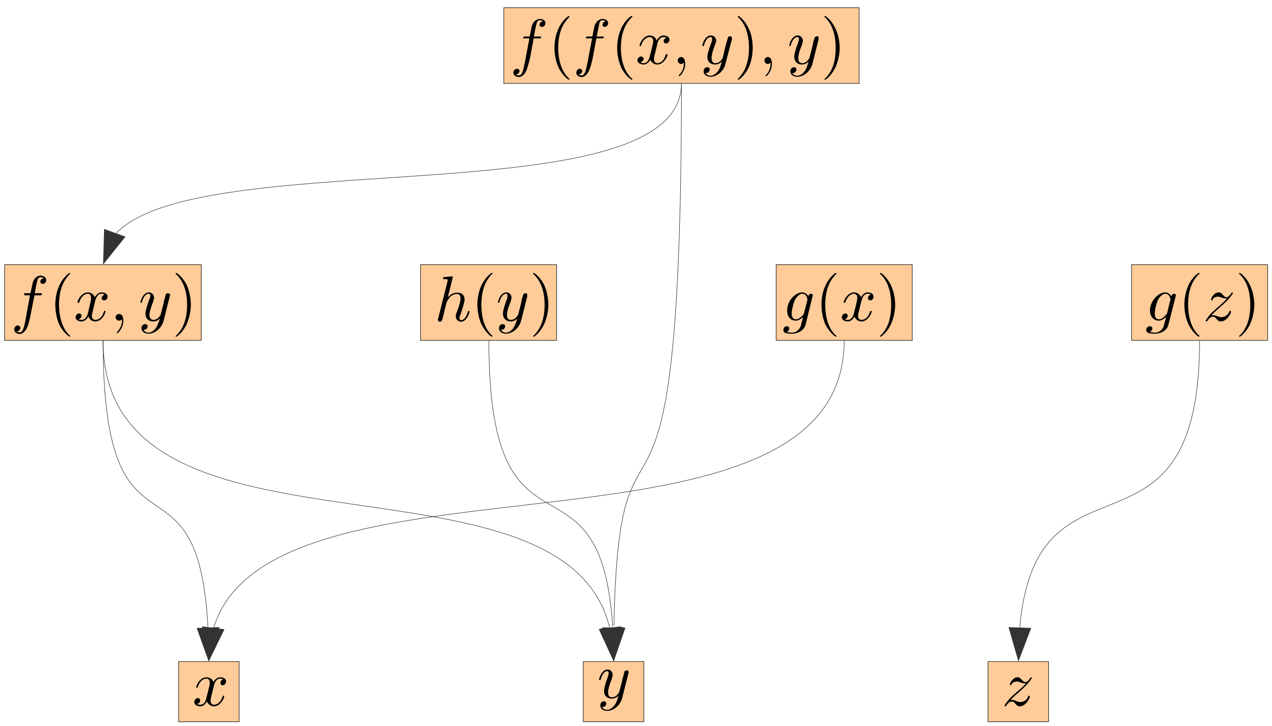
Equality (EUF)

- Polynomial time $O(n \log n)$ **congruence closure** procedure
- Fully incremental and backtrackable (stack-based)
- Supports **efficient *T*-propagation**
  - Exhaustive for positive equalities
  - Incomplete for disequalities
- Lazy explanations and conflict generation

- Typically used as a **"core"** *T*-solver
- Supports efficient extensions, e.g.
  - Integer offsets
  - Bit-vector slicing and concatenation

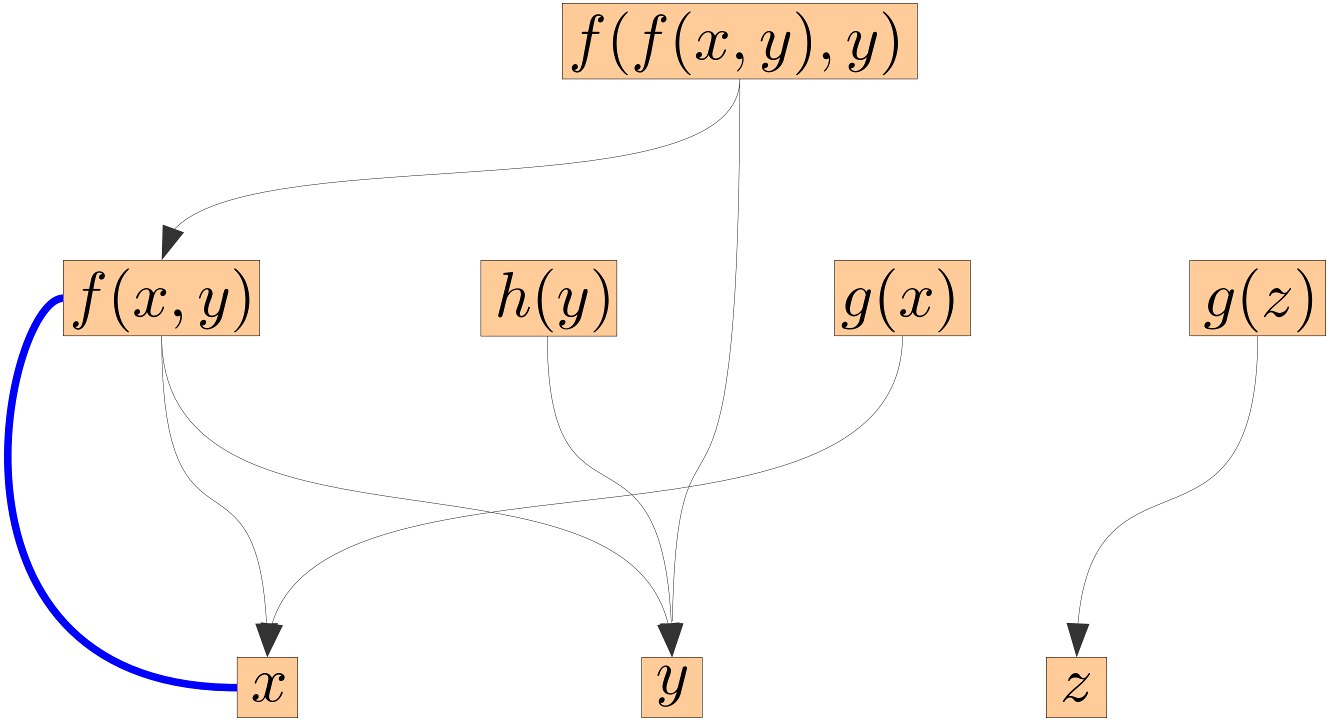
Example

$$[(f(x,y)=x),(h(y)=g(x)),(f(f(x,y),y)=z),\neg(g(x)=g(z))]$$



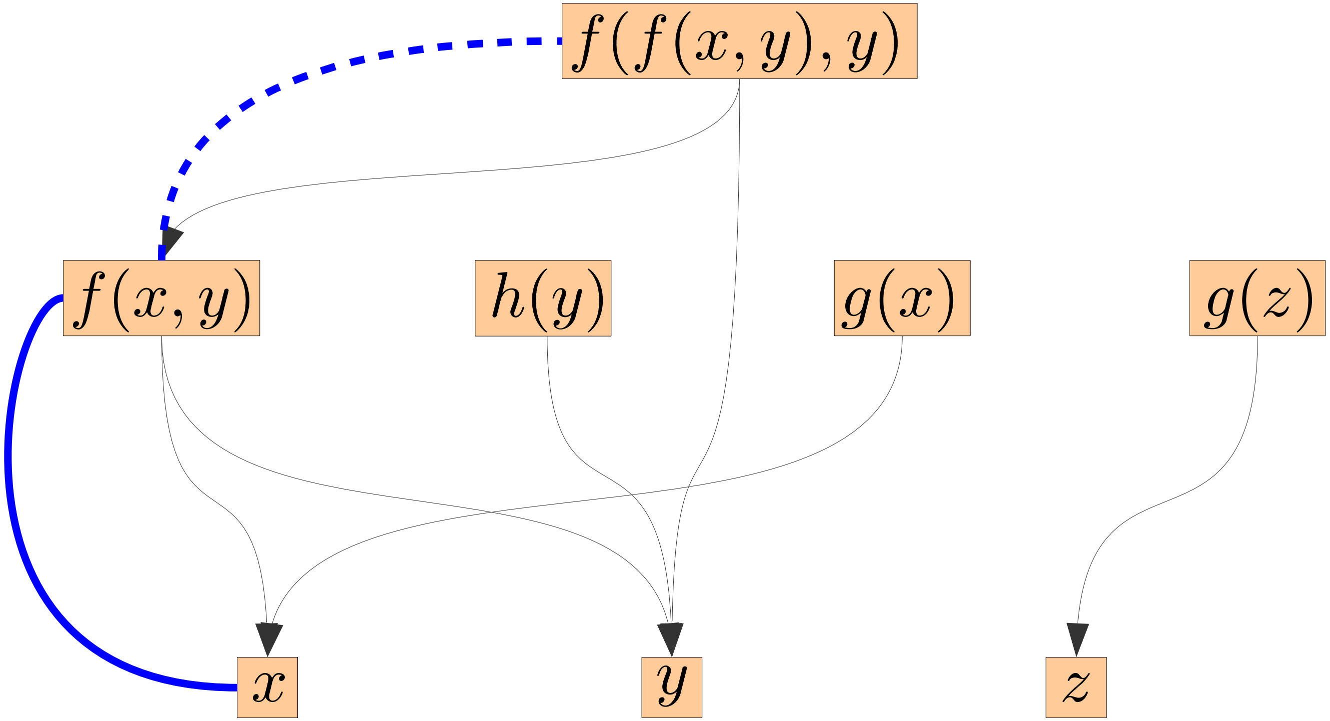
Example

$$[\boxed{(f(x,y)=x)}, (h(y)=g(x)), (f(f(x,y),y)=z), \neg(g(x)=g(z))]$$



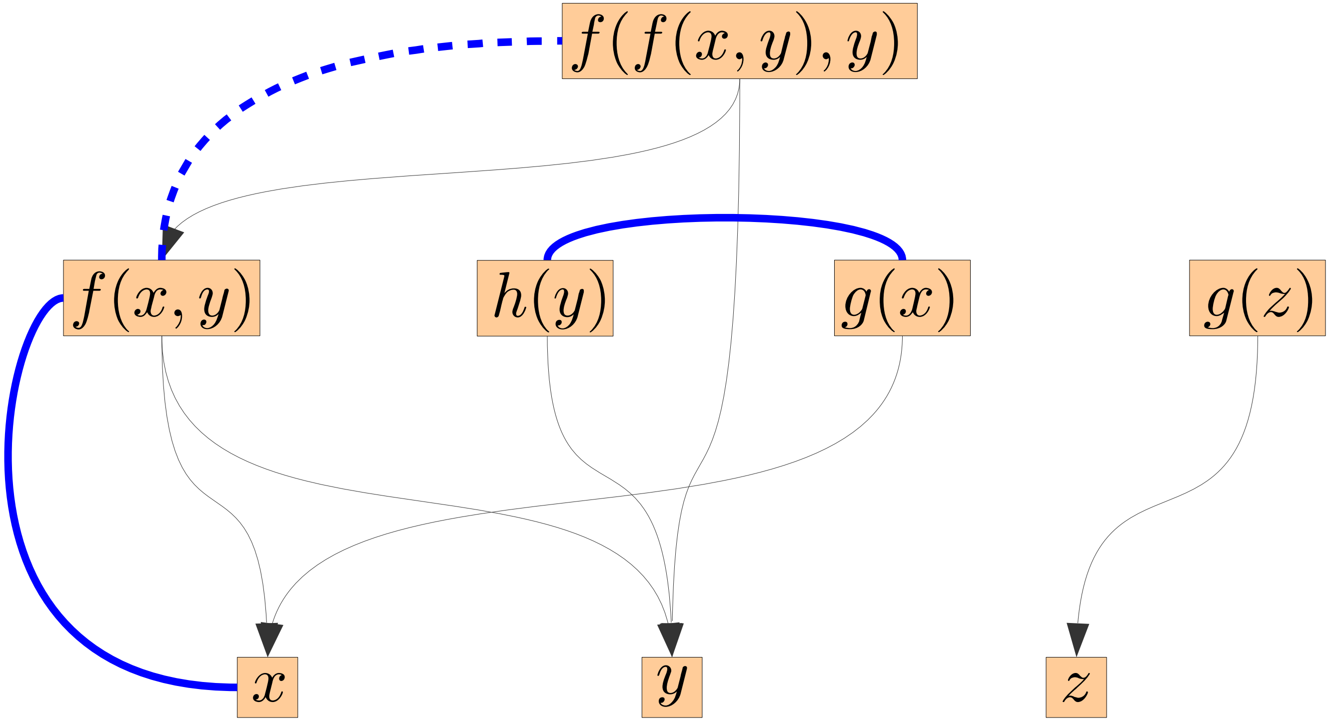
Example

$$[\boxed{(f(x,y)=x)}, (h(y)=g(x)), (f(f(x,y),y)=z), \neg(g(x)=g(z))]$$



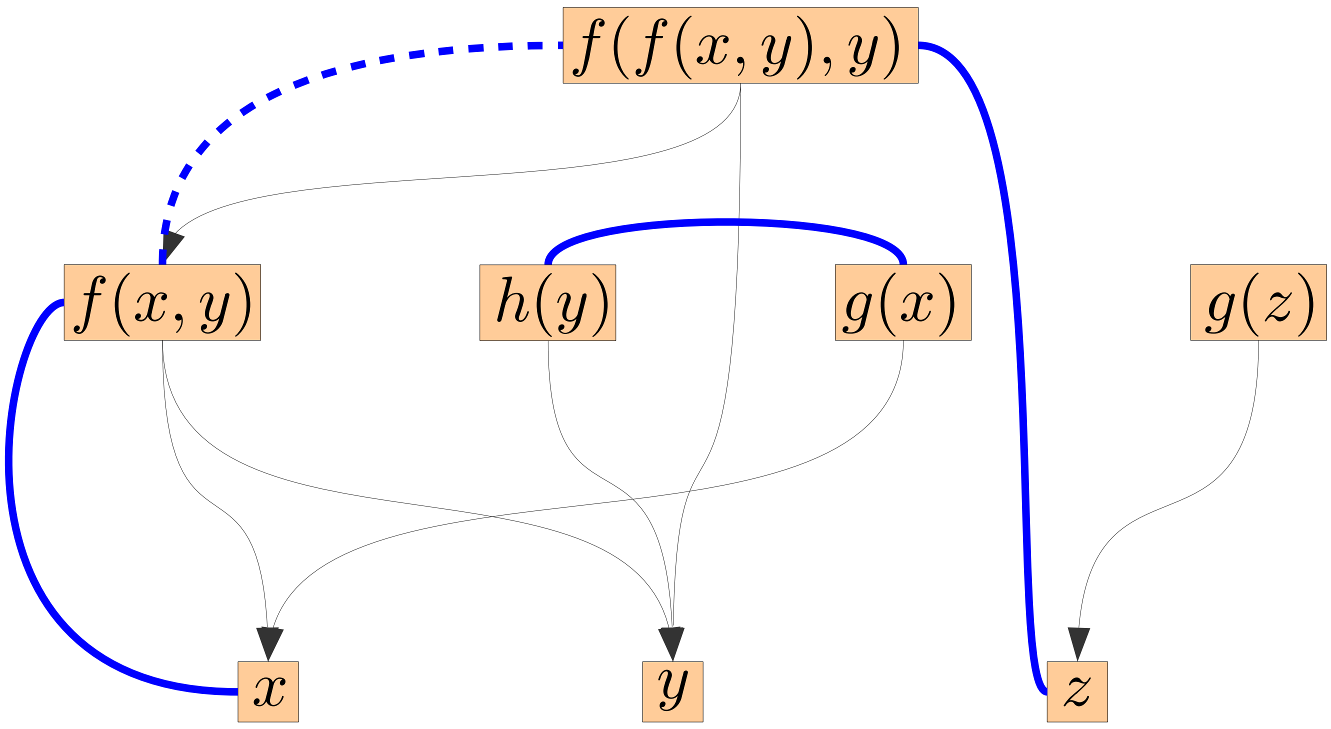
Example

$$[(f(x,y)=x), \boxed{(h(y)=g(x))}, (f(f(x,y),y)=z), \neg(g(x)=g(z))]$$



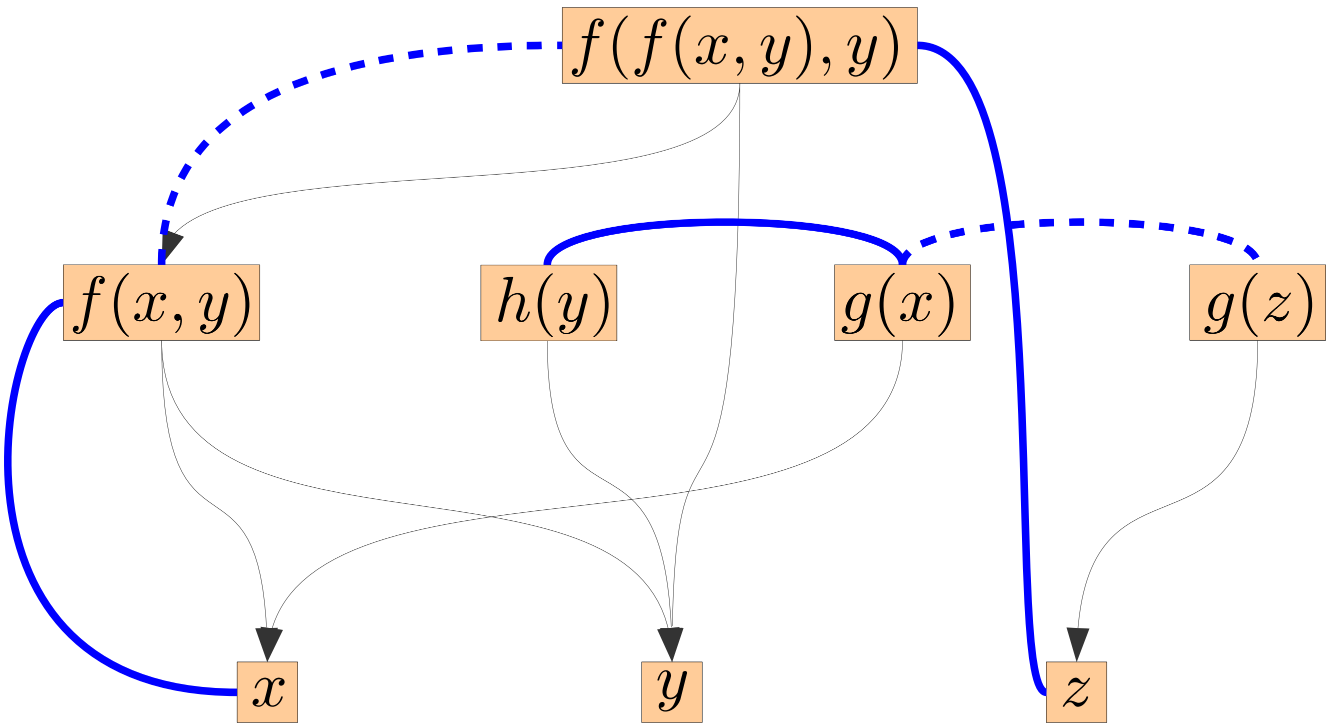
Example

$$[(f(x,y)=x),(h(y)=g(x)),\boxed{(f(f(x,y),y)=z)},\neg(g(x)=g(z))]$$



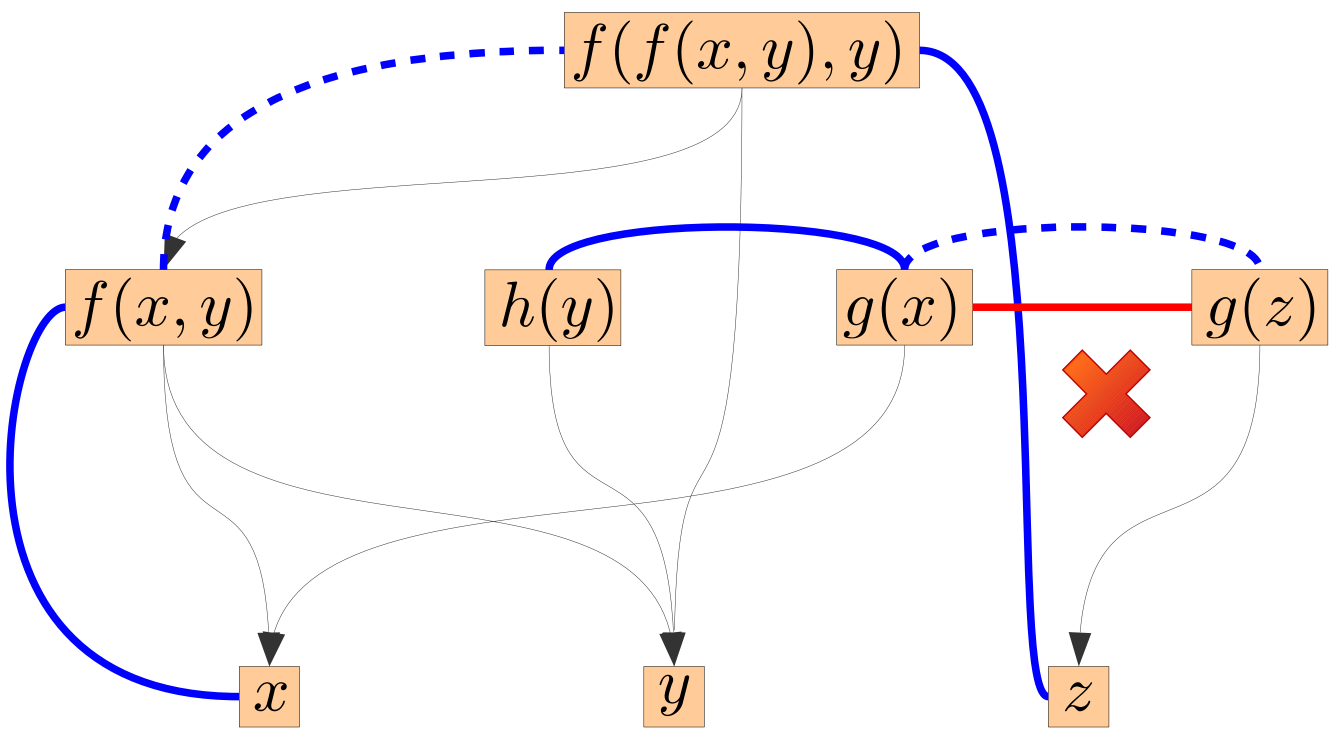
Example

$$[(f(x,y) = x), (h(y) = g(x)), \boxed{(f(f(x,y),y) = z)}, \neg(g(x) = g(z))]$$



Example

$$[(f(x,y)=x),(h(y)=g(x)),(f(f(x,y),y)=z),\boxed{\neg(g(x)=g(z))}]$$

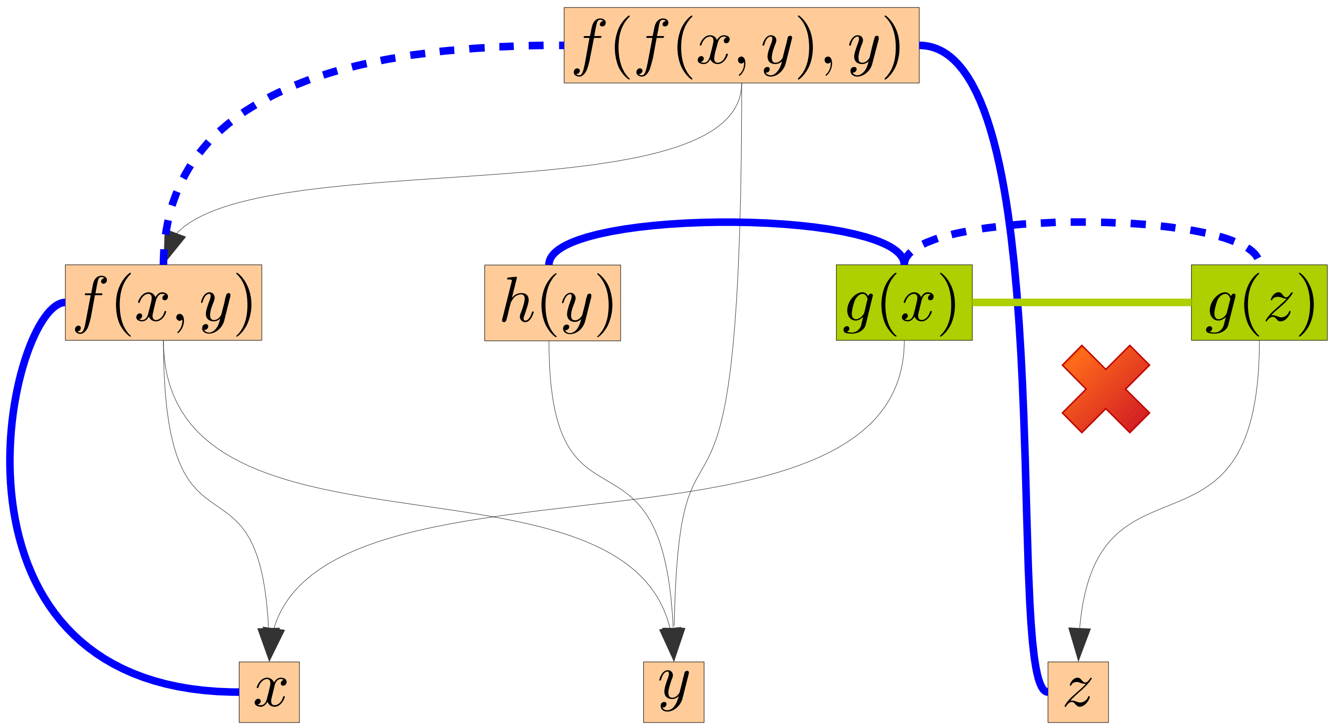


Example

$$[(f(x,y) = x), (h(y) = g(x)), (f(f(x,y),y) = z), \boxed{\neg(g(x) = g(z))}]$$

`get_conflict():`

$$\neg(g(x) = g(z))$$

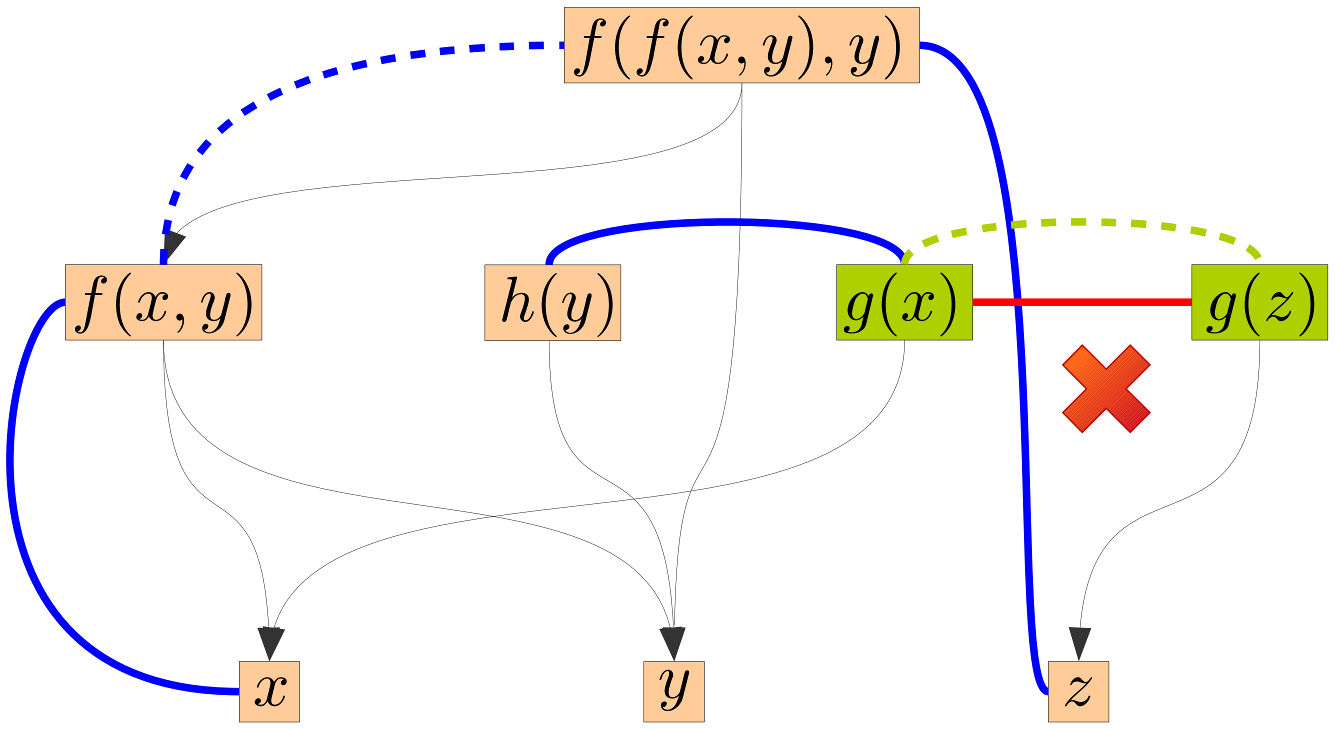


Example

$$[(f(x,y) = x), (h(y) = g(x)), (f(f(x,y),y) = z), \boxed{\neg(g(x) = g(z))}]$$

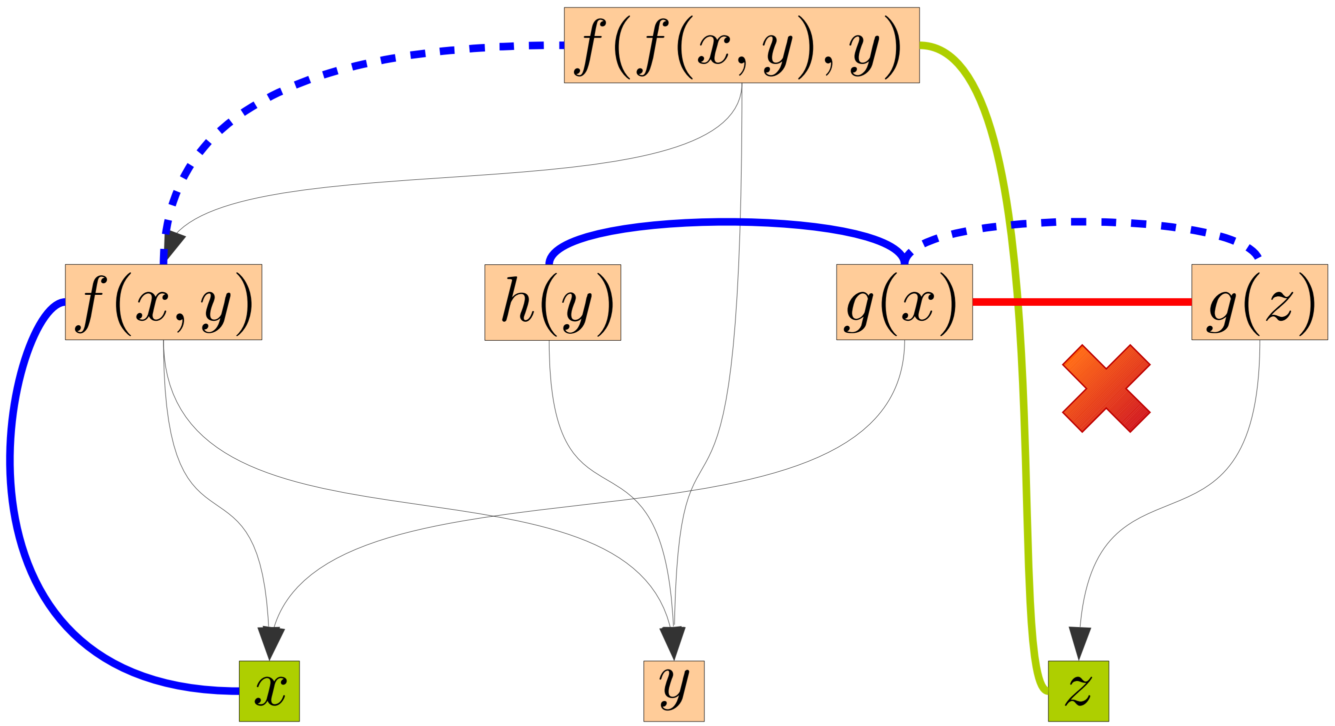
`get_conflict():`

$$\neg(g(x) = g(z))$$



Example

$$[(f(x,y) = x), (h(y) = g(x)), (f(f(x,y),y) = z), \boxed{\neg(g(x) = g(z))}]$$



`get_conflict():`

$$\neg(g(x) = g(z))$$

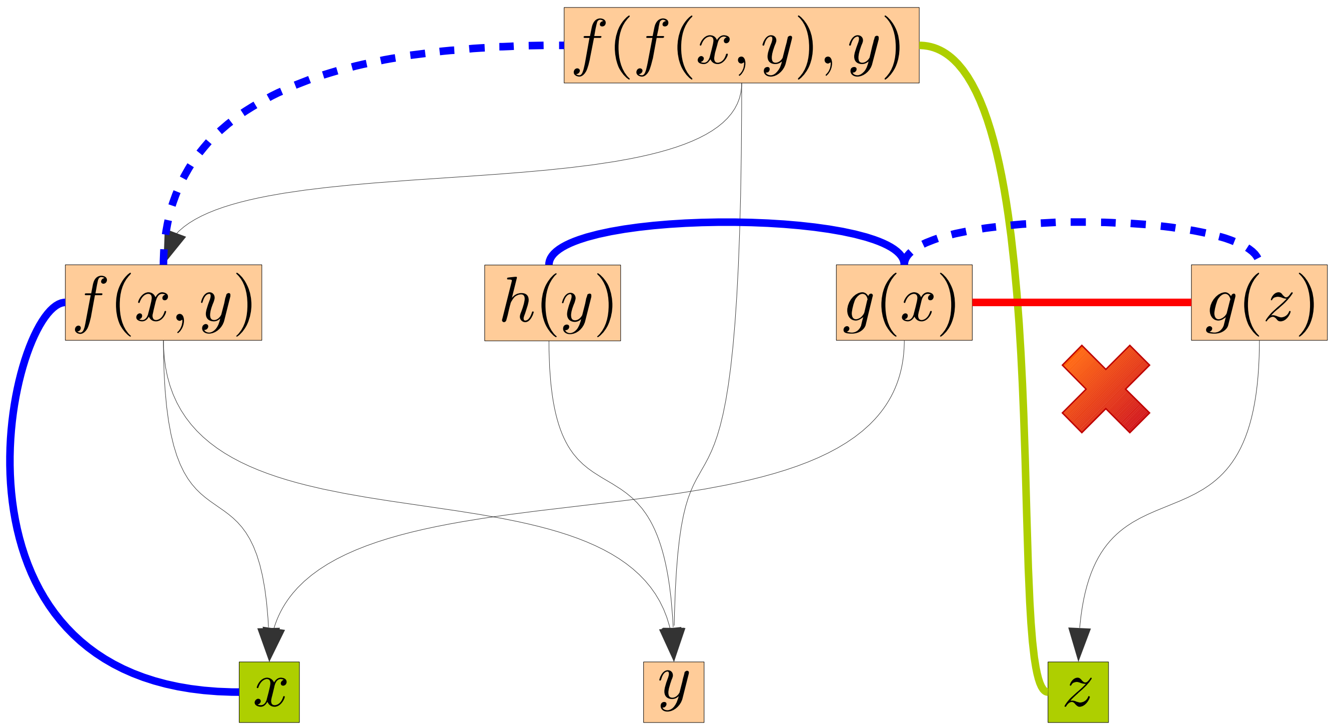
Example

$$[(f(x,y)=x),(h(y)=g(x)),(f(f(x,y),y)=z),\boxed{\neg(g(x)=g(z))}]$$

`get_conflict():`

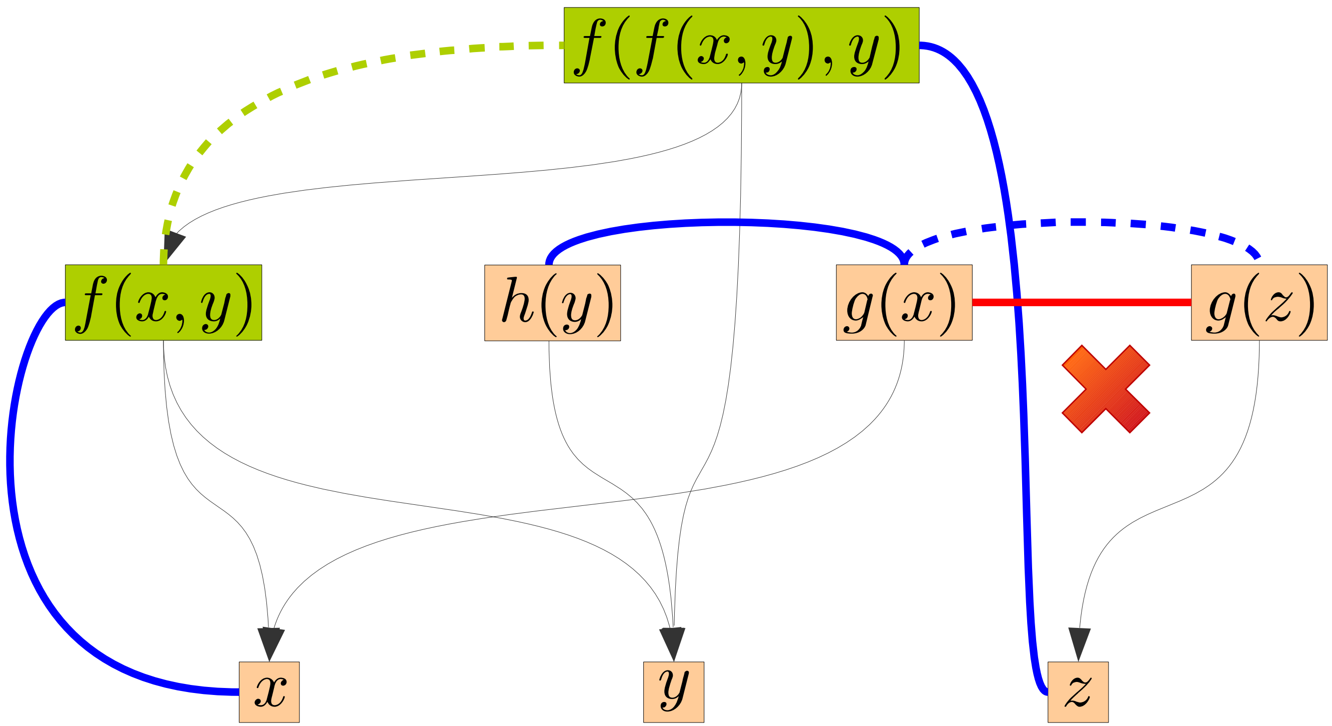
$$\neg(g(x)=g(z))$$

$$(f(f(x,y),y)=z)$$



Example

$$[(f(x,y)=x),(h(y)=g(x)),(f(f(x,y),y)=z),\boxed{\neg(g(x)=g(z))}]$$



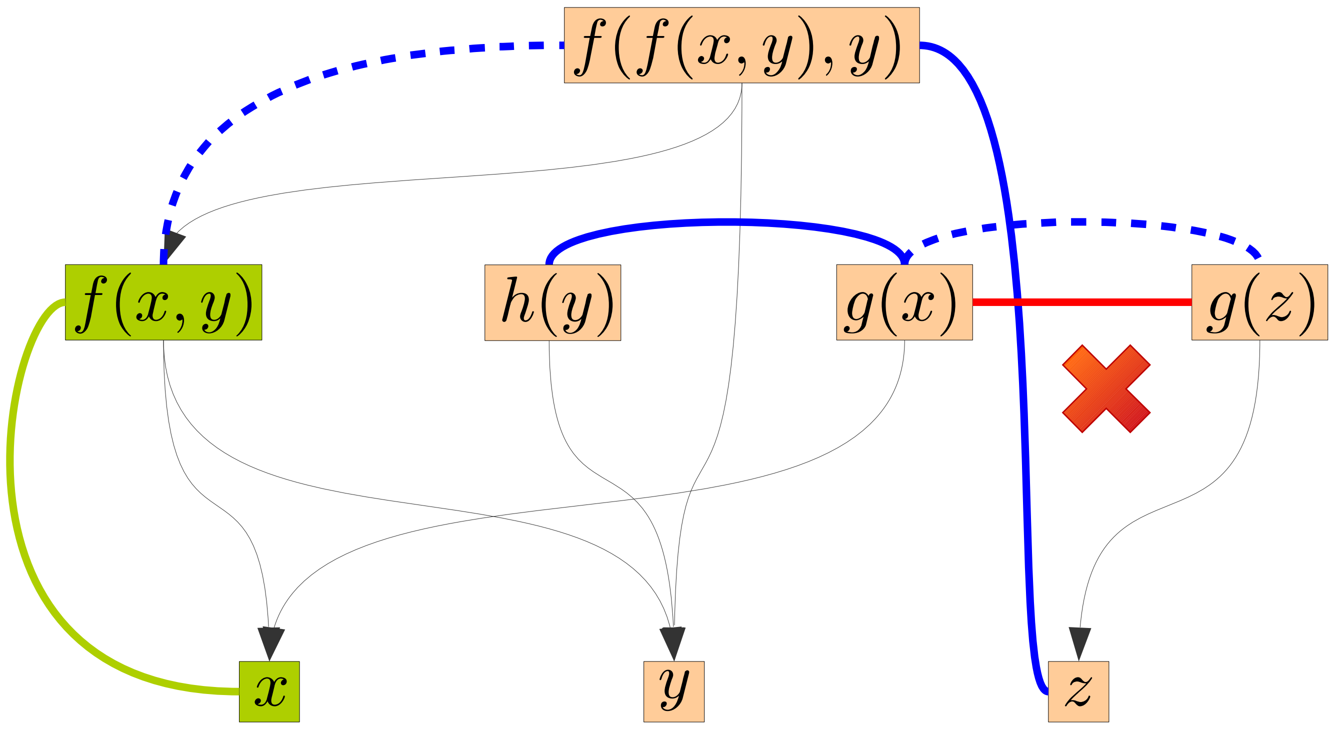
`get_conflict():`

$$\neg(g(x)=g(z))$$

$$(f(f(x,y),y)=z)$$

Example

$$[(f(x,y)=x),(h(y)=g(x)),(f(f(x,y),y)=z),\boxed{\neg(g(x)=g(z))}]$$



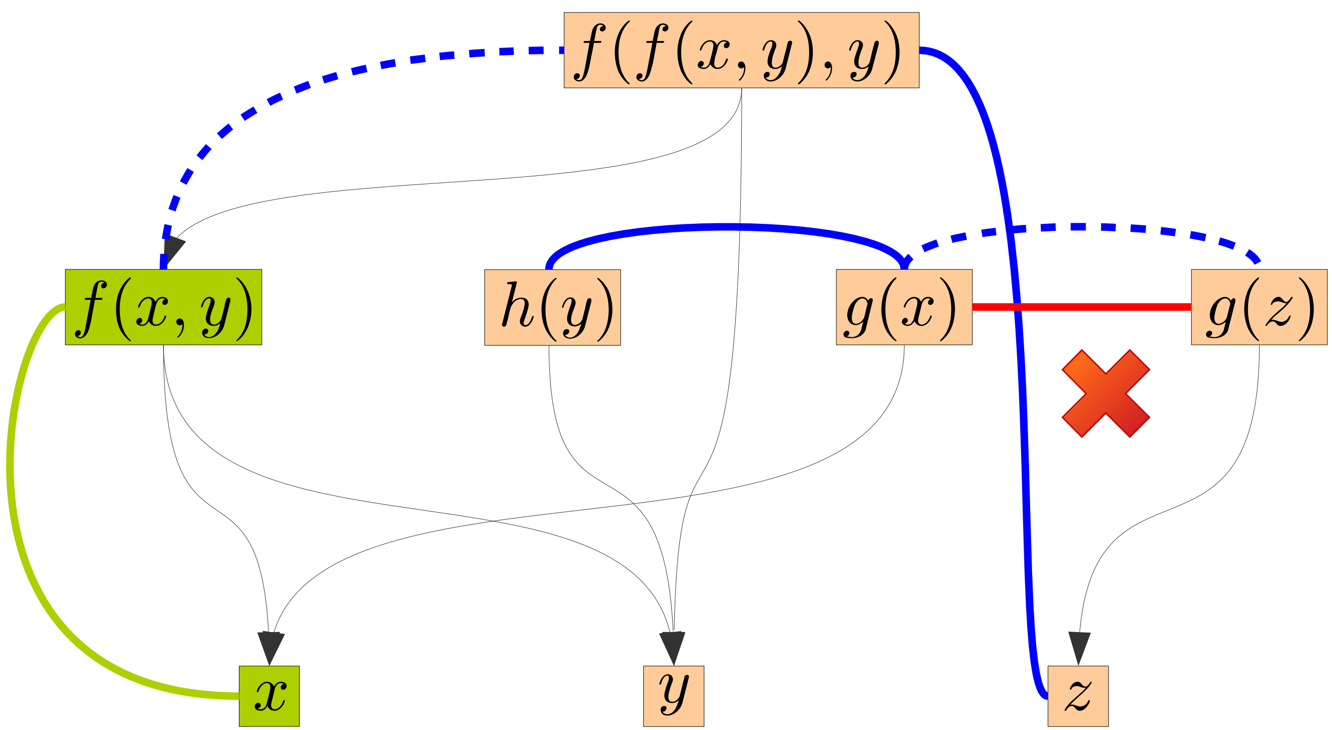

```
get_conflict():
```

$$\neg(g(x)=g(z))$$

$$(f(f(x,y),y)=z)$$

Example

$$[(f(x,y)=x),(h(y)=g(x)),(f(f(x,y),y)=z),\boxed{\neg(g(x)=g(z))}]$$

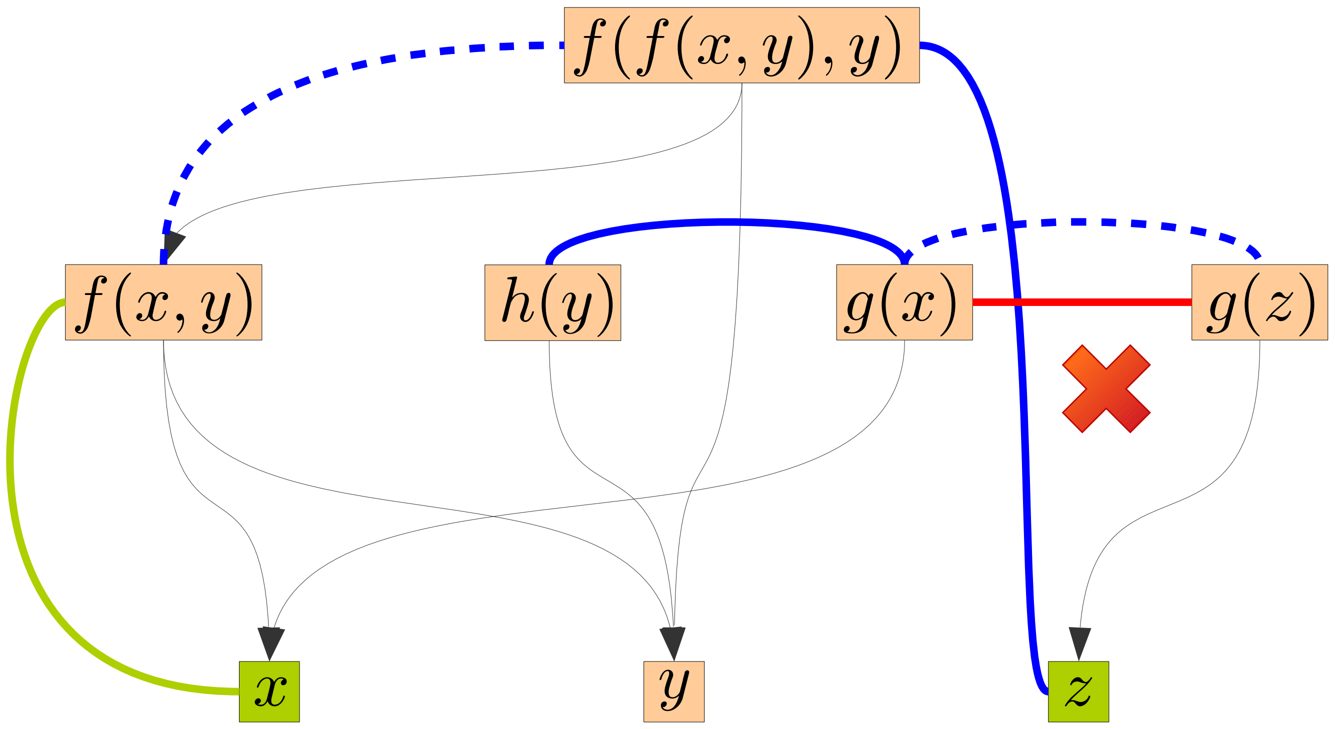



```
get_conflict():
```

$$\neg(g(x)=g(z))$$
$$(f(f(x,y),y)=z)$$
$$(f(x,y)=x)$$

Example

$$[(f(x,y) = x), (h(y) = g(x)), (f(f(x,y),y) = z), \boxed{\neg(g(x) = g(z))}]$$

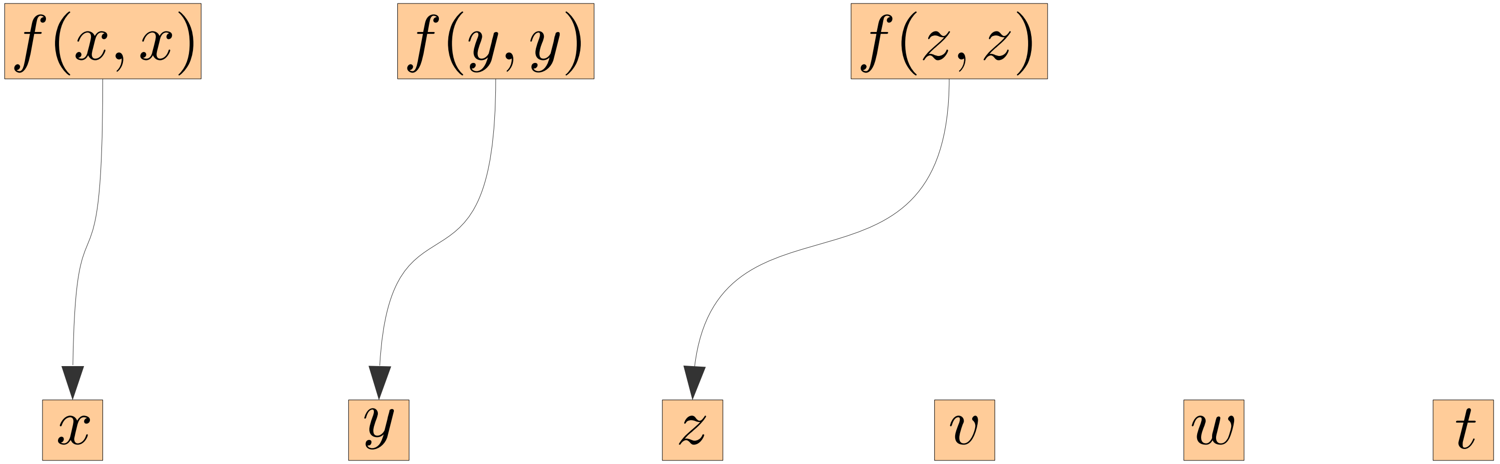


`get_conflict():`

$$\neg(g(x) = g(z))$$
$$(f(f(x,y),y) = z)$$
$$(f(x,y) = x)$$

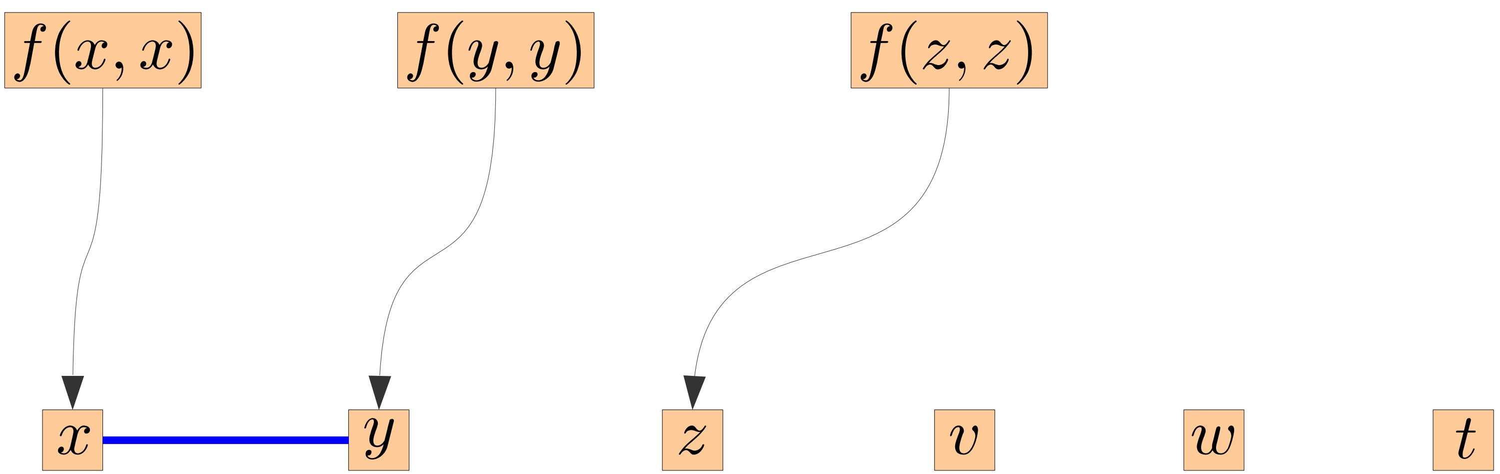
Example: redundant explanations

$$[(x = y), (x = z), (f(x,x) = v), (f(y,y) = w), (f(z,z) = t), \neg(v = t)]$$



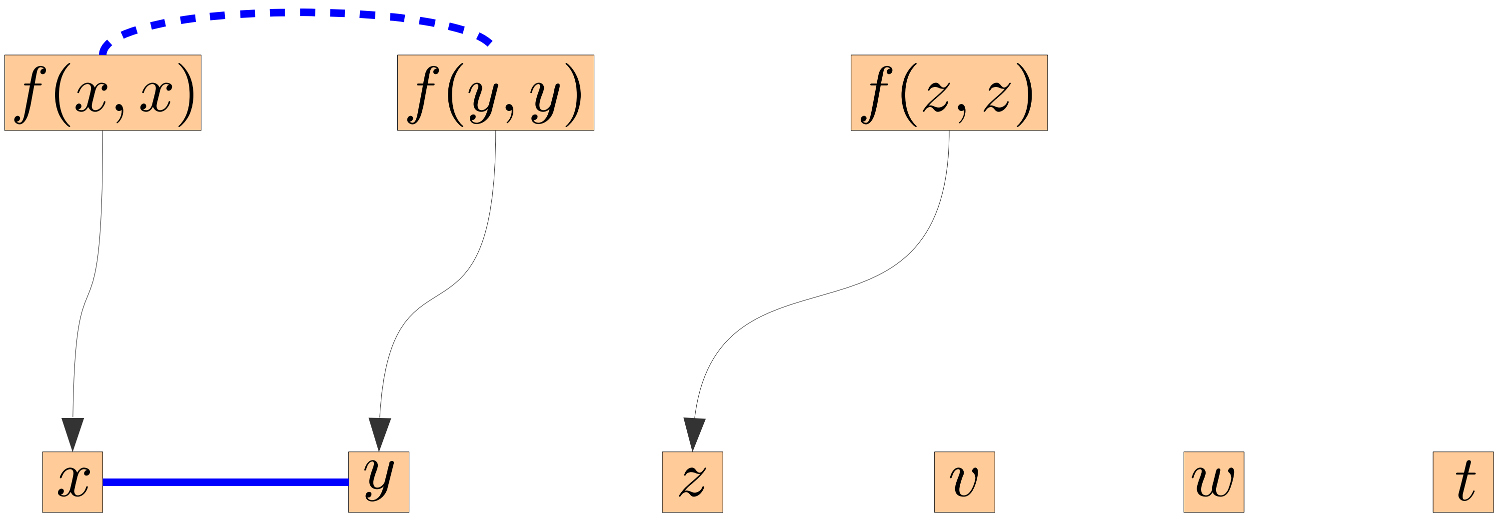
Example: redundant explanations

$$[\boxed{(x = y),}\,(x = z), (f(x, x) = v), (f(y, y) = w), (f(z, z) = t), \neg(v = t)]$$



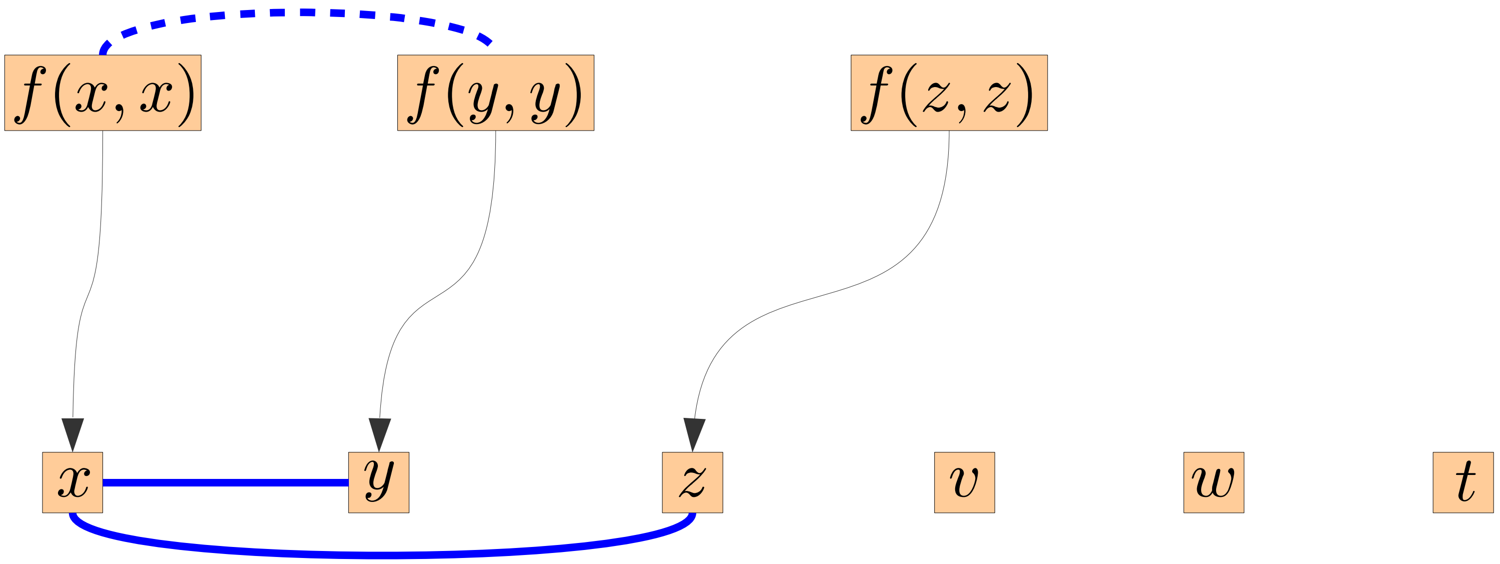
Example: redundant explanations

$$[\boxed{(x = y),}(x = z), (f(x, x) = v), (f(y, y) = w), (f(z, z) = t), \neg(v = t)]$$



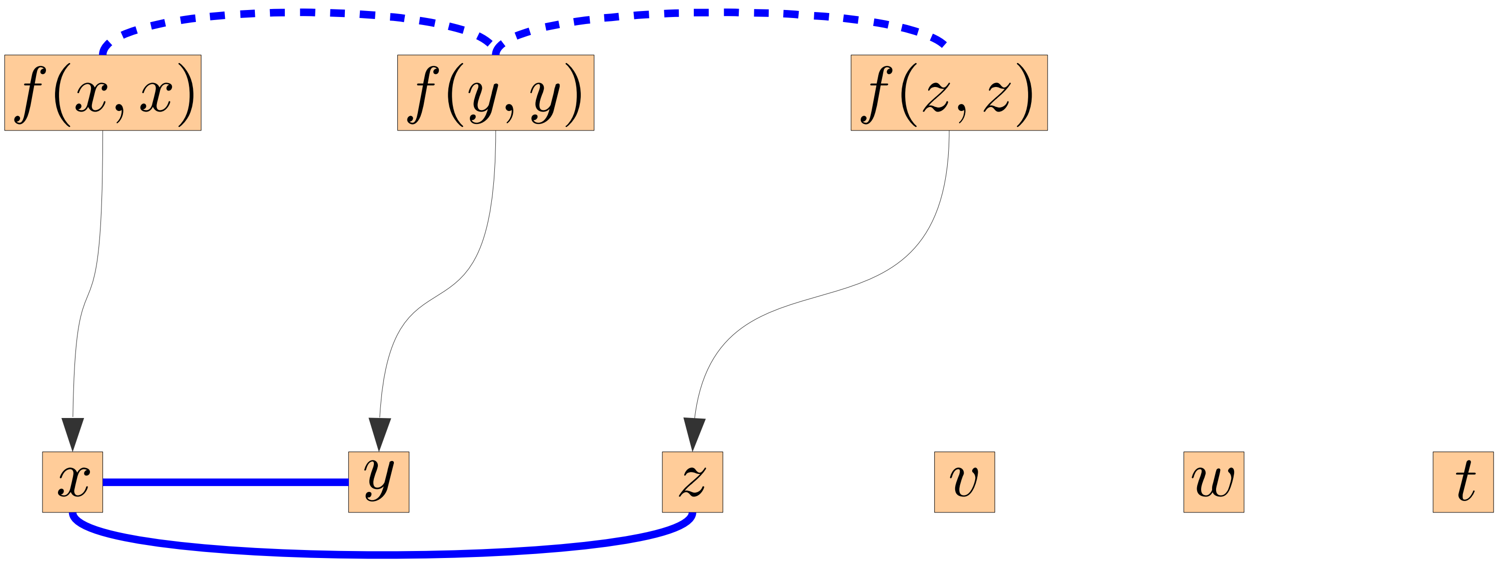
Example: redundant explanations

$$[(x = y), \boxed{(x = z)}, (f(x,x) = v), (f(y,y) = w), (f(z,z) = t), \neg(v = t)]$$



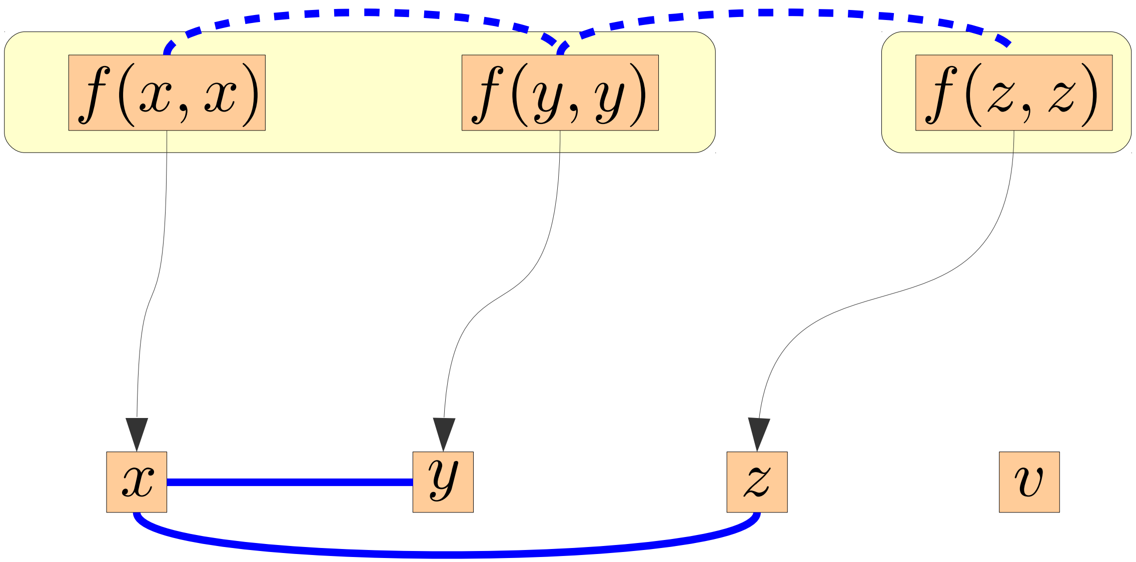
Example: redundant explanations

$$[(x = y), \boxed{(x = z)}, (f(x,x) = v), (f(y,y) = w), (f(z,z) = t), \neg(v = t)]$$



Example: redundant explanations

$$[(x = y), \boxed{(x = z)}, (f(x,x) = v), (f(y,y) = w), (f(z,z) = t), \neg(v = t)]$$



Here we connect the ***equivalence classes*** of

$f(x,x)$ and $f(z,z)$

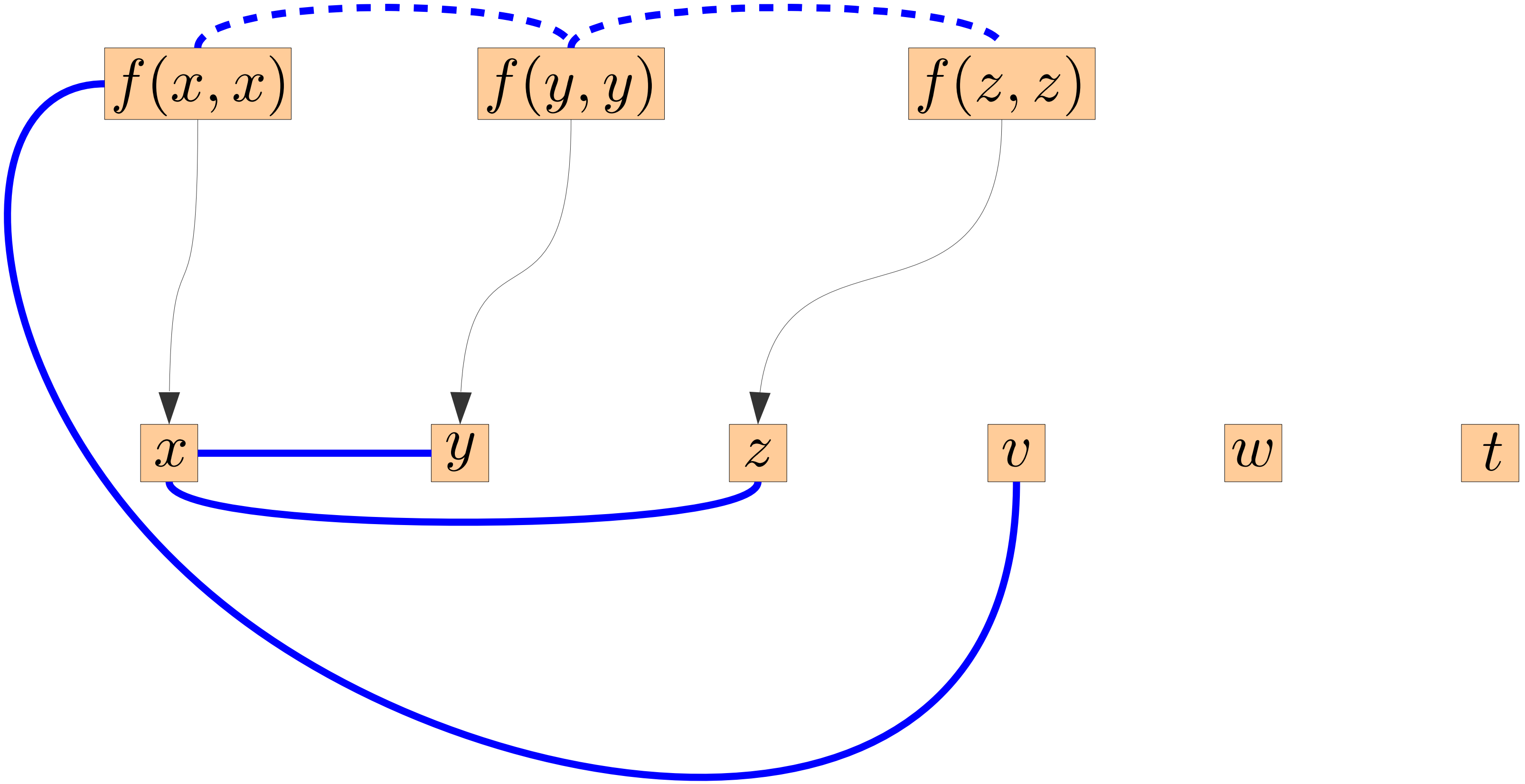
But the representative for

$$\{f(x,x), f(y,y)\}$$

might be $f(y,y)$

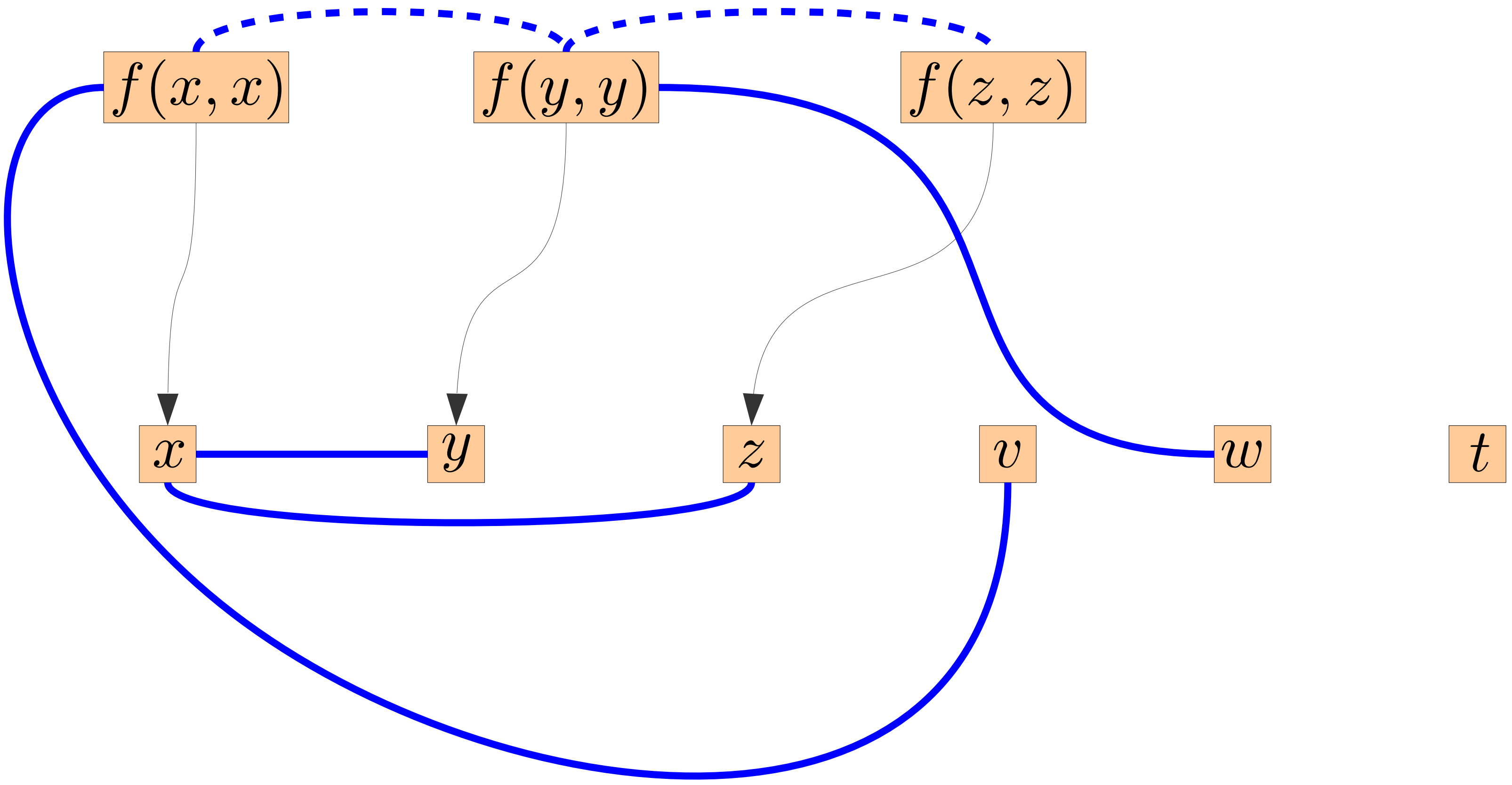
Example: redundant explanations

$$[(x = y), (x = z), \boxed{(f(x,x) = v)}, (f(y,y) = w), (f(z,z) = t), \neg(v = t)]$$



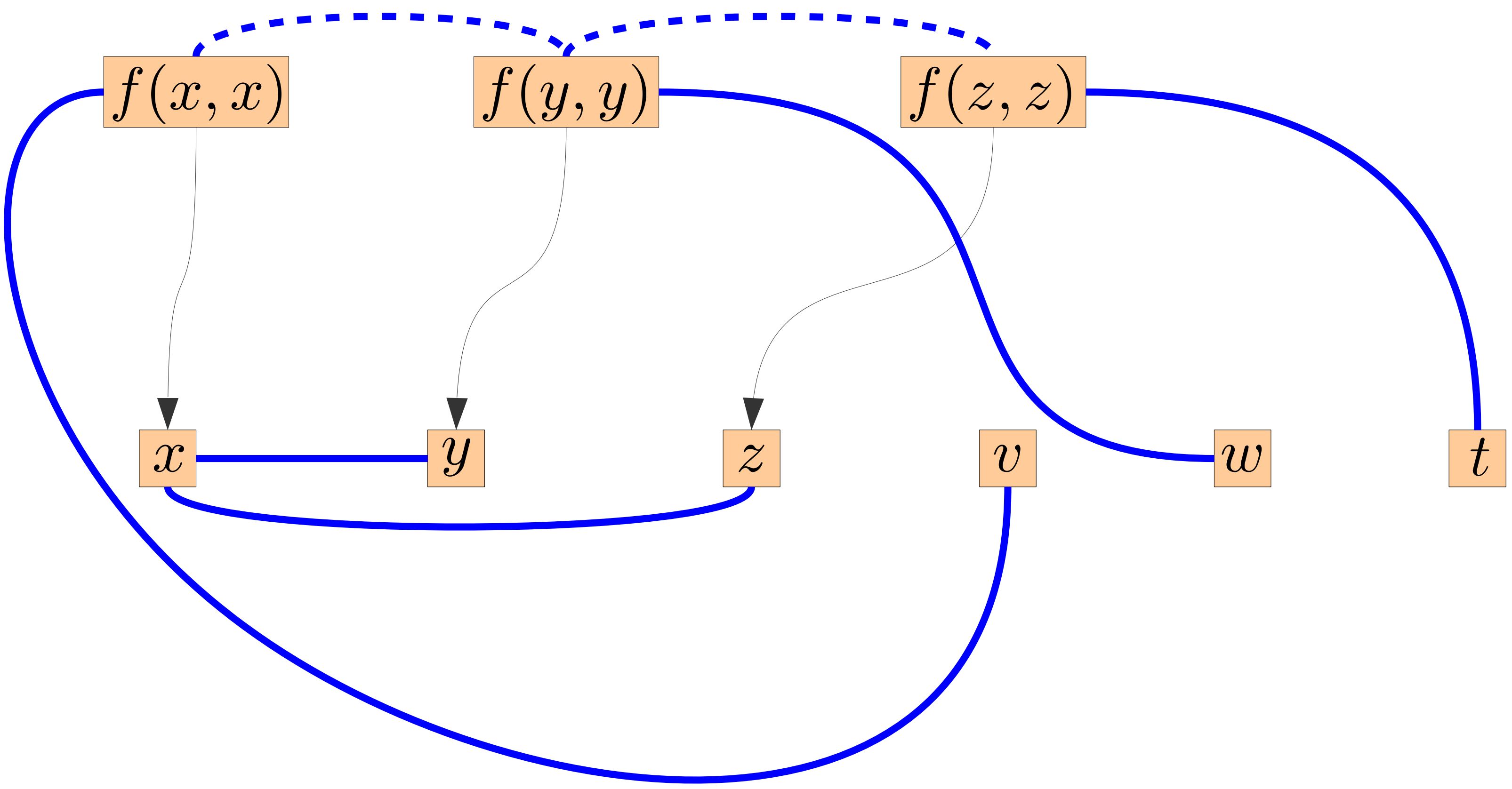
Example: redundant explanations

$$[(x = y), (x = z), (f(x,x) = v), \boxed{(f(y,y) = w)}, (f(z,z) = t), \neg(v = t)]$$



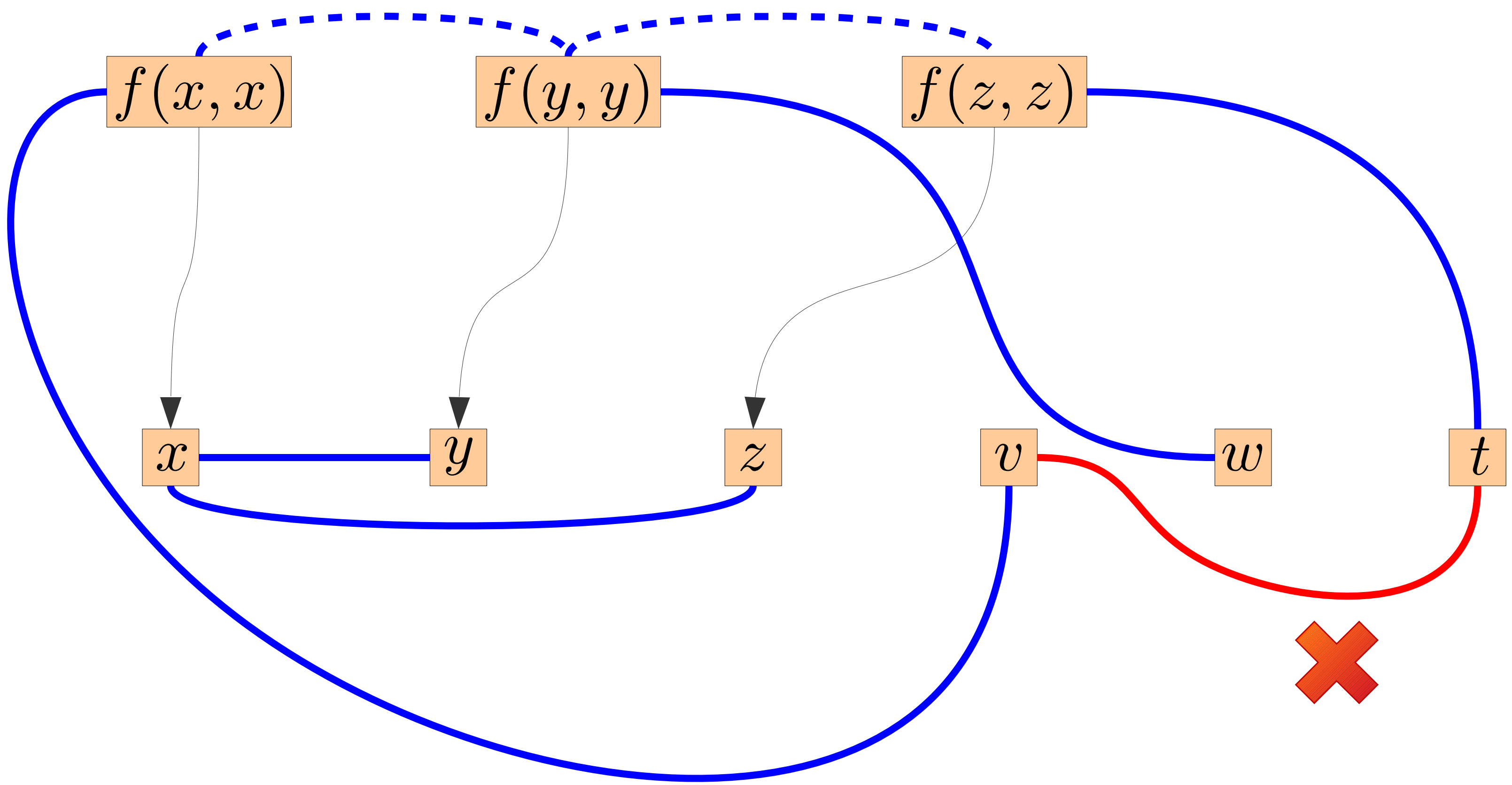
Example: redundant explanations

$$[(x = y), (x = z), (f(x, x) = v), (f(y, y) = w), \boxed{(f(z, z) = t)}, \neg(v = t)]$$



Example: redundant explanations

$$[(x = y), (x = z), (f(x,x) = v), (f(y,y) = w), (f(z,z) = t), \boxed{\neg(v = t)}]$$



Linear Rational Arithmetic (LRA)

- Constraints of the form $\sum_i a_i x_i \leq c$

- Variant of **simplex** specifically designed for DPLL(T)
  - **Very efficient** backtracking
  - Incremental checks
  - **Cheap deduction** of unassigned literals
  - **Minimal explanations** generation
  - Can handle efficiently also strict inequalities
    - Rewrite $(t < 0)$ to $(t + \varepsilon \leq 0)$, treat ε symbolically
  - Worst-case exponential (although LRA is polynomial), but fast in practice

Simplex for DPLL(T)

Preprocessing: $\sum a_h x_h \leq u \mapsto x_{\text{slack}} = \sum a_h x_h \wedge x_{\text{slack}} \leq u$

Tableau of equations (fixed) + bounds (added/removed)

Candidate solution β always consistent with the tableau

$$
\begin{array}{ll}
x_{\text{slack } 1} = & \\
x_{\text{slack } 2} = & \\
\vdots & \\
x_{\text{slack } i} = & a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{im}x_m \\
\vdots & \\
x_{\text{slack } n} = &
\end{array}
\qquad
\begin{array}{|c|}\hline -\infty \\ l_2 \\ \vdots \\ l_i \\ \vdots \\ l_n \\ \hline\end{array}
\leq
\begin{array}{|c|}\hline x_{\text{slack } 1} \\ x_{\text{slack } 2} \\ \vdots \\ x_{\text{slack } i} \\ \vdots \\ x_{\text{slack } n} \\ \hline\end{array}
\leq
\begin{array}{|c|}\hline u_1 \\ +\infty \\ \vdots \\ u_i \\ \vdots \\ u_n \\ \hline\end{array}
$$

Simplex for DPLL(T)

Preprocessing: $\sum a_h x_h \leq u \mapsto x_{\text{slack}} = \sum a_h x_h \wedge x_{\text{slack}} \leq u$

Tableau of equations (fixed) + bounds (added/removed)

Candidate solution β always consistent with the tableau

Pivoting steps to make β satisfy the bounds

$$
\begin{array}{l}
x_j = \\
x_{\text{slack } 2} = \\
\vdots \\
x_{\text{slack } i} = \sum_{a'_{ij}>0} a'_{ij} x_{\text{slack } j} + \sum_{a'_{ik}<0} a'_{ik} x_{\text{slack } k} \\
\vdots \\
x_h =
\end{array}
\qquad
\begin{bmatrix} -\infty \\ l_2 \\ \vdots \\ l_i \\ \vdots \\ l_n \end{bmatrix}
\leq
\begin{bmatrix} x_{\text{slack } 1} \\ x_{\text{slack } 2} \\ \vdots \\ x_{\text{slack } i} \\ \vdots \\ x_{\text{slack } n} \end{bmatrix}
\leq
\begin{bmatrix} u_1 \\ +\infty \\ \vdots \\ u_i \\ \vdots \\ u_n \end{bmatrix}
$$

Simplex for DPLL(T)

Preprocessing: $\sum a_h x_h \leq u \mapsto x_{\text{slack}} = \sum a_h x_h \wedge x_{\text{slack}} \leq u$

Tableau of equations (fixed) + bounds (added/removed)

Candidate solution β always consistent with the tableau

$\beta(x_{\text{slack}\ i}) < l_i$

and for the others β can not change $\Rightarrow$ conflict!

$x_j =$

$x_{\text{slack } 2} =$

$\vdots$

$x_{\text{slack}\ i} = \sum_{a'_{ij} > 0} a'_{ij} x_{\text{slack}\ j} + \sum_{a'_{ik} < 0} a'_{ik} x_{\text{slack}\ k}$

$\vdots$

$x_h =$

$-\infty$		$x_{\text{slack } 1}$		u_1
l_2		$x_{\text{slack } 2}$		$+\infty$
$\vdots$		$\vdots$		$\vdots$
l_i	$>$	$x_{\text{slack } i}$	$\leq$	u_i
$\vdots$		$\vdots$		$\vdots$
l_n		$x_{\text{slack } n}$		u_n

Simplex for DPLL(T)

Preprocessing: $\sum a_h x_h \leq u \mapsto x_{\text{slack}} = \sum a_h x_h \wedge x_{\text{slack}} \leq u$

Tableau of equations (fixed) + bounds (added/removed)

Candidate solution β always consistent with the tableau

$\beta(x_{\text{slack } i}) < l_i$
and for the others β can not change $\Rightarrow$ conflict!

$x_j =$

$x_{\text{slack } 2} =$

$\vdots$

$x_{\text{slack } i} = \sum_{a'_{ij} > 0} a'_{ij} x_{\text{slack } j} + \sum_{a'_{ik} < 0} a'_{ik} x_{\text{slack } k}$

$\vdots$

$x_h =$

```
get_conflict():
```

$\{(\sum a_h x_h \leq u)\}_j$ for $x_{\text{slack } j}$ $\cup$

$\{(\sum a_h x_h \geq l)\}_k$ for $x_{\text{slack } k}$ $\cup$

$\{(\sum a_h x_h \geq l)\}_i$ for $x_{\text{slack } i}$

Example

$$[(3x_2 - x_1 \leq 1), (x_1 + x_2 \geq 0), (x_3 - 2x_1 \geq 3), (2x_3 \leq 1)]$$

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{aligned} s_1 &= 3x_2 - x_1 \\ s_2 &= x_1 + x_2 \\ s_3 &= x_3 - 2x_1 \\ s_4 &= 2x_3 \end{aligned}$$

bounds

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

candidate solution β

$$\begin{array}{rcl} x_1 & \mapsto & 0 \\ x_2 & \mapsto & 0 \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & 0 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 0 \\ s_4 & \mapsto & 0 \end{array}$$

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{aligned} s_1 &= 3x_2 - x_1 \\ s_2 &= x_1 + x_2 \\ s_3 &= x_3 - 2x_1 \\ s_4 &= 2x_3 \end{aligned}$$

bounds

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ \boxed{3 \leq s_3 \leq \infty} \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

candidate solution β

$$\begin{array}{lcl} x_1 & \mapsto & 0 \\ x_2 & \mapsto & 0 \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & 0 \\ s_2 & \mapsto & 0 \\ \boxed{s_3 \mapsto 0} \\ s_4 & \mapsto & 0 \end{array}$$

Find a bound violation

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{array}{lcl} s_1 & = & 3x_2 - x_1 \\ s_2 & = & x_1 + x_2 \\ \boxed{s_3 \;=\; x_3 - 2x_1} \\ s_4 & = & 2x_3 \end{array}$$

bounds

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

candidate solution β

$$\begin{array}{lcl} \boxed{x_1 \;\mapsto\; 0} \\ x_2 & \mapsto & 0 \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & 0 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 0 \\ s_4 & \mapsto & 0 \end{array}$$

Pick a variable for pivoting

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

bounds

candidate solution β

$$\begin{array}{rcl} s_1 & = & 3x_2 - \frac{1}{2}x_3 + \frac{1}{2}s_3 \\ s_2 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 + x_2 \\ x_1 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 \\ s_4 & = & 2x_3 \end{array}$$

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{3}{2} \\ x_2 & \mapsto & 0 \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & \frac{3}{2} \\ s_2 & \mapsto & -\frac{3}{2} \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 0 \end{array}$$

Pivot and update β

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{array}{rcl} s_1 & = & 3x_2 - \frac{1}{2}x_3 + \frac{1}{2}s_3 \\ s_2 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 + x_2 \\ x_1 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 \\ s_4 & = & 2x_3 \end{array}$$

bounds

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ \boxed{0} & \boxed{\leq} & \boxed{s_2} & \boxed{\leq} & \boxed{\infty} \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

candidate solution β

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{3}{2} \\ x_2 & \mapsto & 0 \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & \frac{3}{2} \\ \boxed{s_2} & \boxed{\mapsto} & \boxed{-\frac{3}{2}} \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 0 \end{array}$$

Find a bound violation

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{array}{rcl} s_1 & = & 3x_2 - \frac{1}{2}x_3 + \frac{1}{2}s_3 \\ \boxed{s_2} & \boxed{=} & \boxed{\frac{1}{2}x_3 - \frac{1}{2}s_3 + x_2} \\ x_1 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 \\ s_4 & = & 2x_3 \end{array}$$

bounds

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

candidate solution β

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{3}{2} \\ \boxed{x_2} & \boxed{\mapsto} & \boxed{0} \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & \frac{3}{2} \\ s_2 & \mapsto & -\frac{3}{2} \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 0 \end{array}$$

Pick a variable for pivoting

Example

$$[\underbrace{(3x_2 - x_1 \le 1)}_{s_1}, \underbrace{(x_1 + x_2 \ge 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \ge 3)}_{s_3}, \underbrace{(2x_3 \le 1)}_{s_4}]$$

tableau

$$\begin{array}{rcl} s_1 & = & 3s_2 - 2x_3 + 2s_3 \\ x_2 & = & s_2 - \frac{1}{2}x_3 + \frac{1}{2}s_3 \\ x_1 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 \\ s_4 & = & 2x_3 \end{array}$$

bounds

$$\begin{array}{rcccl} -\infty & \le & s_1 & \le & 1 \\ 0 & \le & s_2 & \le & \infty \\ 3 & \le & s_3 & \le & \infty \\ -\infty & \le & s_4 & \le & 1 \end{array}$$

candidate solution β

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{3}{2} \\ x_2 & \mapsto & \frac{3}{2} \\ x_3 & \mapsto & 0 \\ s_1 & \mapsto & 6 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 0 \end{array}$$

Pivot and update β

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{array}{rcl} s_1 & = & 3s_2 - 2x_3 + 2s_3 \\ x_2 & = & s_2 - \frac{1}{2}x_3 + \frac{1}{2}s_3 \\ x_1 & = & \frac{1}{2}x_3 - \frac{1}{2}s_3 \\ s_4 & = & 2x_3 \end{array}$$

bounds

$$\begin{array}{rcccl} \boxed{-\infty & \leq & s_1 & \leq & 1} \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

candidate solution β

$$\begin{array}{lcr} x_1 & \mapsto & -\frac{3}{2} \\ x_2 & \mapsto & \frac{3}{2} \\ x_3 & \mapsto & 0 \\ \boxed{s_1 & \mapsto & 6} \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 0 \end{array}$$

Find a bound violation

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

$$\begin{array}{rcl} \boxed{s_1 = 3s_2 - 2x_3 + 2s_3} \\ x_2 &=& s_2 - \frac{1}{2}x_3 + \frac{1}{2}s_3 \\ x_1 &=& \frac{1}{2}x_3 - \frac{1}{2}s_3 \\ s_4 &=& 2x_3 \end{array}$$

bounds

$$\begin{array}{rcccl} -\infty &\leq& s_1 &\leq& 1 \\ 0 &\leq& s_2 &\leq& \infty \\ 3 &\leq& s_3 &\leq& \infty \\ -\infty &\leq& s_4 &\leq& 1 \end{array}$$

candidate solution β

$$\begin{array}{rcr} x_1 &\mapsto& -\frac{3}{2} \\ x_2 &\mapsto& \frac{3}{2} \\ \boxed{x_3 \mapsto 0} \\ s_1 &\mapsto& 6 \\ s_2 &\mapsto& 0 \\ s_3 &\mapsto& 3 \\ s_4 &\mapsto& 0 \end{array}$$

Pick a variable for pivoting

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau | bounds | candidate solution β

$$\begin{array}{rcl} x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\ x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\ x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\ s_4 & = & -s_1 + 3s_2 + 2s_3 \end{array}$$

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{1}{4} \\ x_2 & \mapsto & \frac{1}{4} \\ x_3 & \mapsto & \frac{5}{2} \\ s_1 & \mapsto & 1 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 5 \end{array}$$

Pivot and update β

Example

$$[\underbrace{(3x_2 - x_1 \le 1)}_{s_1}, \underbrace{(x_1 + x_2 \ge 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \ge 3)}_{s_3}, \underbrace{(2x_3 \le 1)}_{s_4}]$$

tableau

bounds

candidate solution β

$$\begin{array}{rcl} x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\ x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\ x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\ s_4 & = & -s_1 + 3s_2 + 2s_3 \end{array}$$

$$\begin{array}{rcccl} -\infty & \le & s_1 & \le & 1 \\ 0 & \le & s_2 & \le & \infty \\ 3 & \le & s_3 & \le & \infty \\ \boxed{-\infty \le s_4 \le 1} \end{array}$$

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{1}{4} \\ x_2 & \mapsto & \frac{1}{4} \\ x_3 & \mapsto & \frac{5}{2} \\ s_1 & \mapsto & 1 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ \boxed{s_4 \mapsto 5} \end{array}$$

Find a bound violation

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau | bounds | candidate solution β

$$\begin{array}{rcl} x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\ x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\ x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\ \boxed{s_4 = -s_1 + 3s_2 + 2s_3} \end{array}
\qquad
\begin{array}{ccccc} -\infty & \leq & \boxed{s_1 \leq 1} \\ \boxed{0 \leq s_2} & \leq & \infty \\ \boxed{3 \leq s_3} & \leq & \infty \\ -\infty & \leq & \boxed{s_4 \leq 1} \end{array}
\qquad
\begin{array}{rcl} x_1 & \mapsto & -\frac{1}{4} \\ x_2 & \mapsto & \frac{1}{4} \\ x_3 & \mapsto & \frac{5}{2} \\ s_1 & \mapsto & 1 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 5 \end{array}$$

No suitable variable for pivoting!
Conflict

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

bounds

candidate solution β

$$\begin{array}{lcl} x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\ x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\ x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\ s_4 & = & -s_1 + 3s_2 + 2s_3 \end{array}$$

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

$$\begin{array}{lcr} x_1 & \mapsto & -\frac{1}{4} \\ x_2 & \mapsto & \frac{1}{4} \\ x_3 & \mapsto & \frac{5}{2} \\ s_1 & \mapsto & 1 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 5 \end{array}$$

Explanation:

$(3x_2 - x_1 \leq 1)$

Example

$$[\underbrace{(3x_2 - x_1 \le 1)}_{s_1}, \underbrace{(x_1 + x_2 \ge 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \ge 3)}_{s_3}, \underbrace{(2x_3 \le 1)}_{s_4}]$$

tableau

$$\begin{array}{rcl} x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\ x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\ x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\ s_4 & = & -s_1 + 3s_2 + 2s_3 \end{array}$$

bounds

$$\begin{array}{rcccl} -\infty & \le & s_1 & \le & 1 \\ 0 & \le & s_2 & \le & \infty \\ 3 & \le & s_3 & \le & \infty \\ -\infty & \le & s_4 & \le & 1 \end{array}$$

candidate solution β

$$\begin{array}{rcr} x_1 & \mapsto & -\frac{1}{4} \\ x_2 & \mapsto & \frac{1}{4} \\ x_3 & \mapsto & \frac{5}{2} \\ s_1 & \mapsto & 1 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 5 \end{array}$$

Explanation:

$(3x_2 - x_1 \le 1), (x_1 + x_2 \ge 0)$

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau | bounds | candidate solution β

$$\begin{array}{rcl} x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\ x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\ x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\ s_4 & = & -s_1 + 3s_2 + 2s_3 \end{array}$$

$$\begin{array}{rcccl} -\infty & \leq & s_1 & \leq & 1 \\ 0 & \leq & s_2 & \leq & \infty \\ 3 & \leq & s_3 & \leq & \infty \\ -\infty & \leq & s_4 & \leq & 1 \end{array}$$

$$\begin{array}{lcr} x_1 & \mapsto & -\frac{1}{4} \\ x_2 & \mapsto & \frac{1}{4} \\ x_3 & \mapsto & \frac{5}{2} \\ s_1 & \mapsto & 1 \\ s_2 & \mapsto & 0 \\ s_3 & \mapsto & 3 \\ s_4 & \mapsto & 5 \end{array}$$

Explanation:

$(3x_2 - x_1 \leq 1), (x_1 + x_2 \geq 0), (x_3 - 2x_1 \geq 3)$

Example

$$[\underbrace{(3x_2 - x_1 \leq 1)}_{s_1}, \underbrace{(x_1 + x_2 \geq 0)}_{s_2}, \underbrace{(x_3 - 2x_1 \geq 3)}_{s_3}, \underbrace{(2x_3 \leq 1)}_{s_4}]$$

tableau

bounds

candidate solution β

$$\begin{array}{rcl}
x_3 & = & -\frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 \\
x_2 & = & \frac{1}{4}s_1 + \frac{1}{4}s_2 \\
x_1 & = & -\frac{1}{4}s_1 + \frac{3}{4}s_2 \\
\boxed{s_4 = -s_1 + 3s_2 + 2s_3}
\end{array}
\qquad
\begin{array}{rcccl}
-\infty & \leq & s_1 & \leq & 1 \\
0 & \leq & s_2 & \leq & \infty \\
3 & \leq & s_3 & \leq & \infty \\
-\infty & \leq & \boxed{s_4 \leq 1}
\end{array}
\qquad
\begin{array}{rcl}
x_1 & \mapsto & -\frac{1}{4} \\
x_2 & \mapsto & \frac{1}{4} \\
x_3 & \mapsto & \frac{5}{2} \\
s_1 & \mapsto & 1 \\
s_2 & \mapsto & 0 \\
s_3 & \mapsto & 3 \\
s_4 & \mapsto & 5
\end{array}$$

Explanation:

$$(3x_2 - x_1 \leq 1), (x_1 + x_2 \geq 0), (x_3 - 2x_1 \geq 3), (2x_3 \leq 1)$$

Linear Integer Arithmetic (LIA)

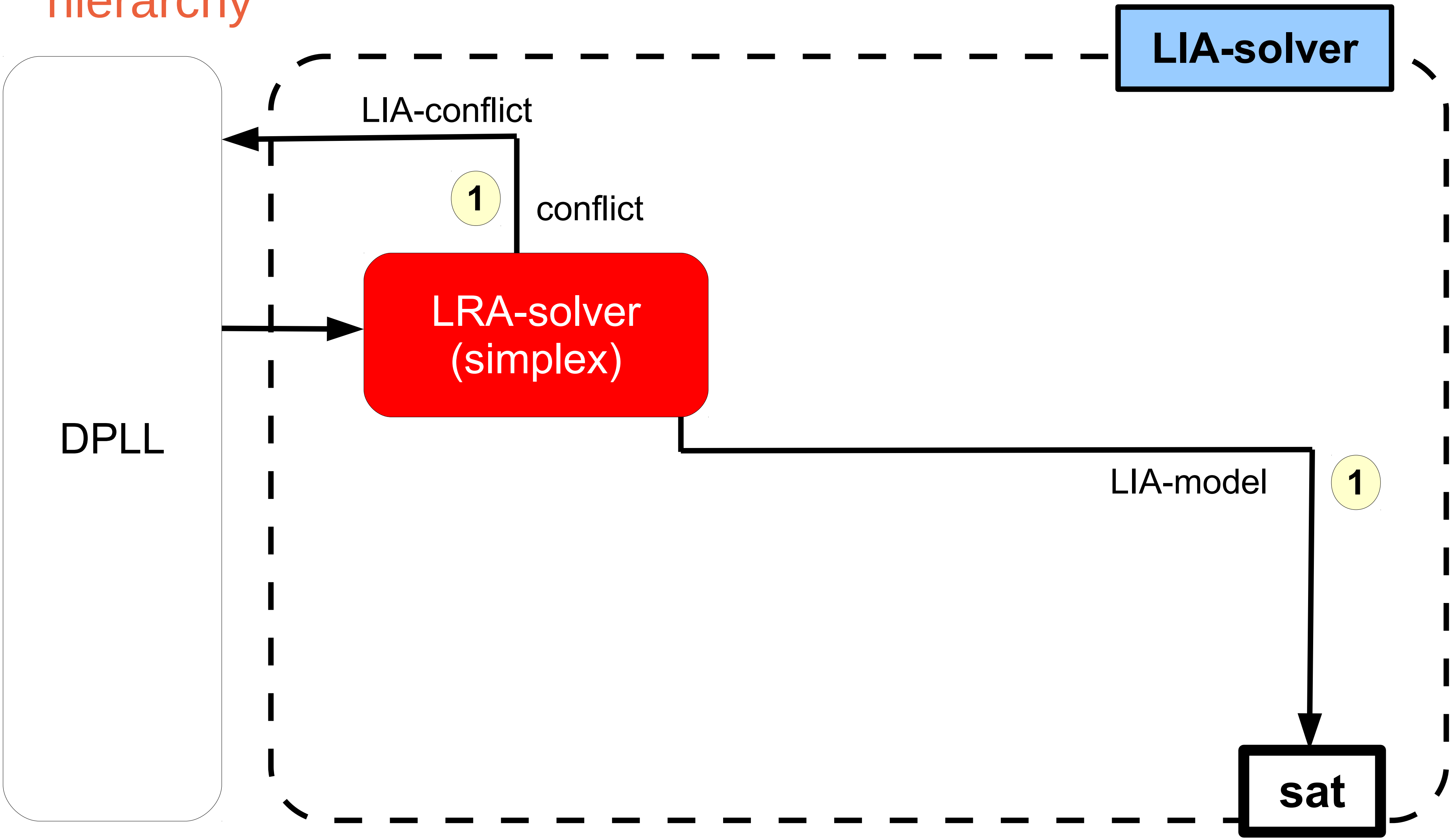
- **NP-complete** problem
- Popular approach: simplex + **branch and bound**
  - Approximate checks solve only over the rationals
  - In complete checks, force integrality of variables by adding either:
    - **Branch and bound** lemmas $(x \leq \lfloor c \rfloor) \vee (x \geq \lceil c \rceil)$
    - **Cutting plane** lemmas
      - Inequalities entailed by the current constraints, excluding only non-integer solutions
      - Gomory cuts commonly used
    - Using splitting on-demand
  - Might also include other specialized sub-solvers for tractable fragments
    - E.g. specialized equational reasoning

Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy

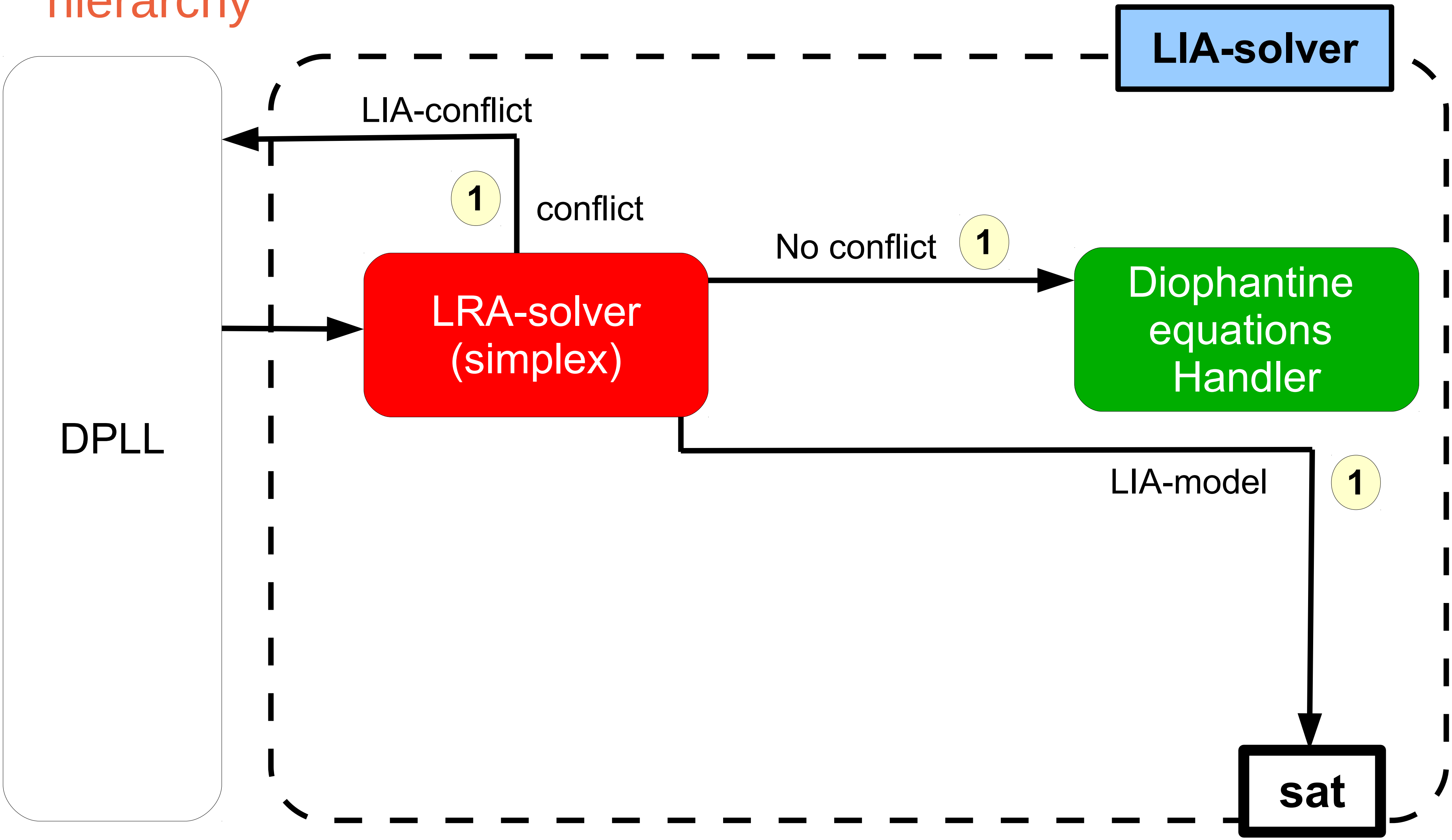
Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy



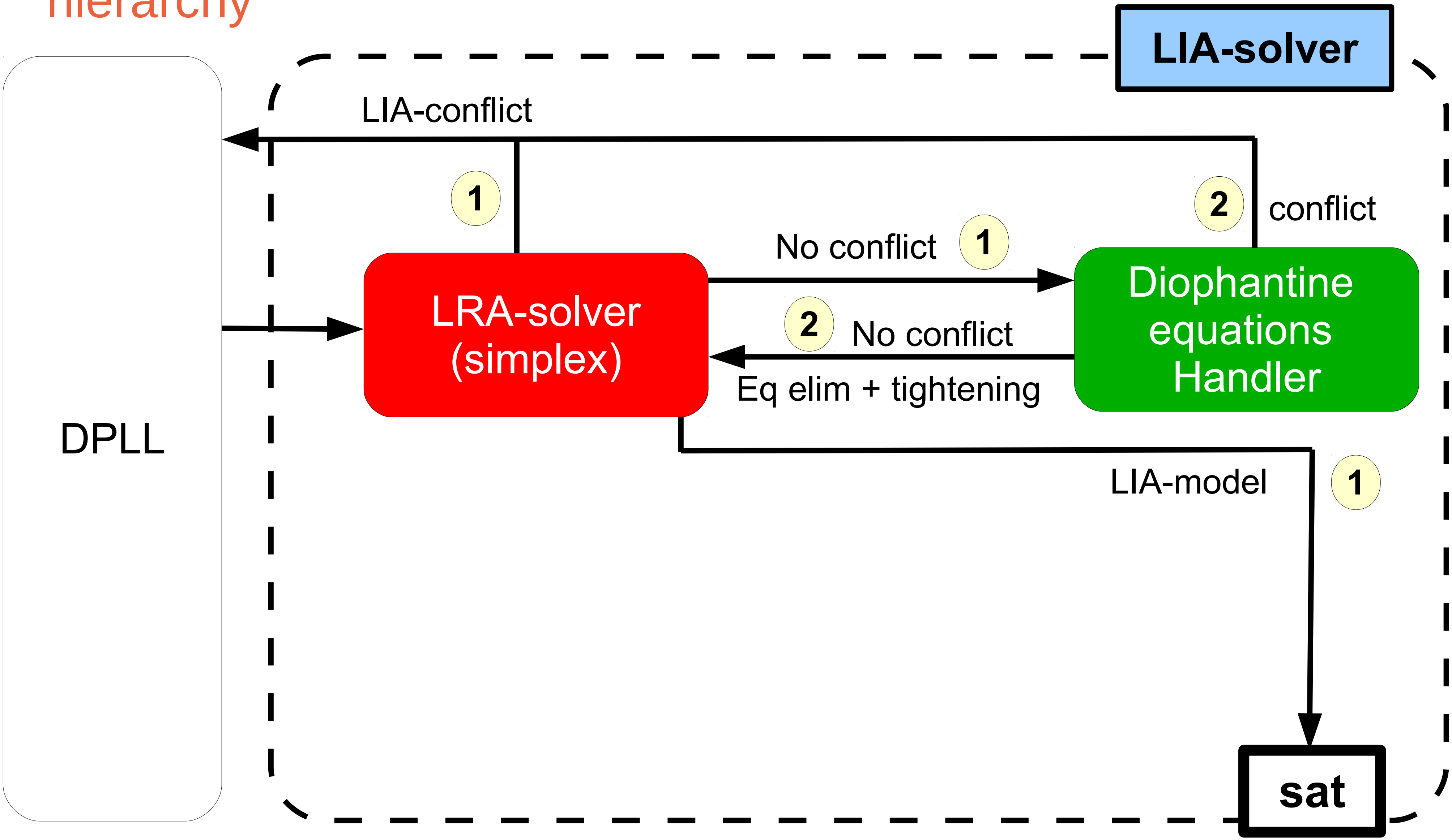
Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy



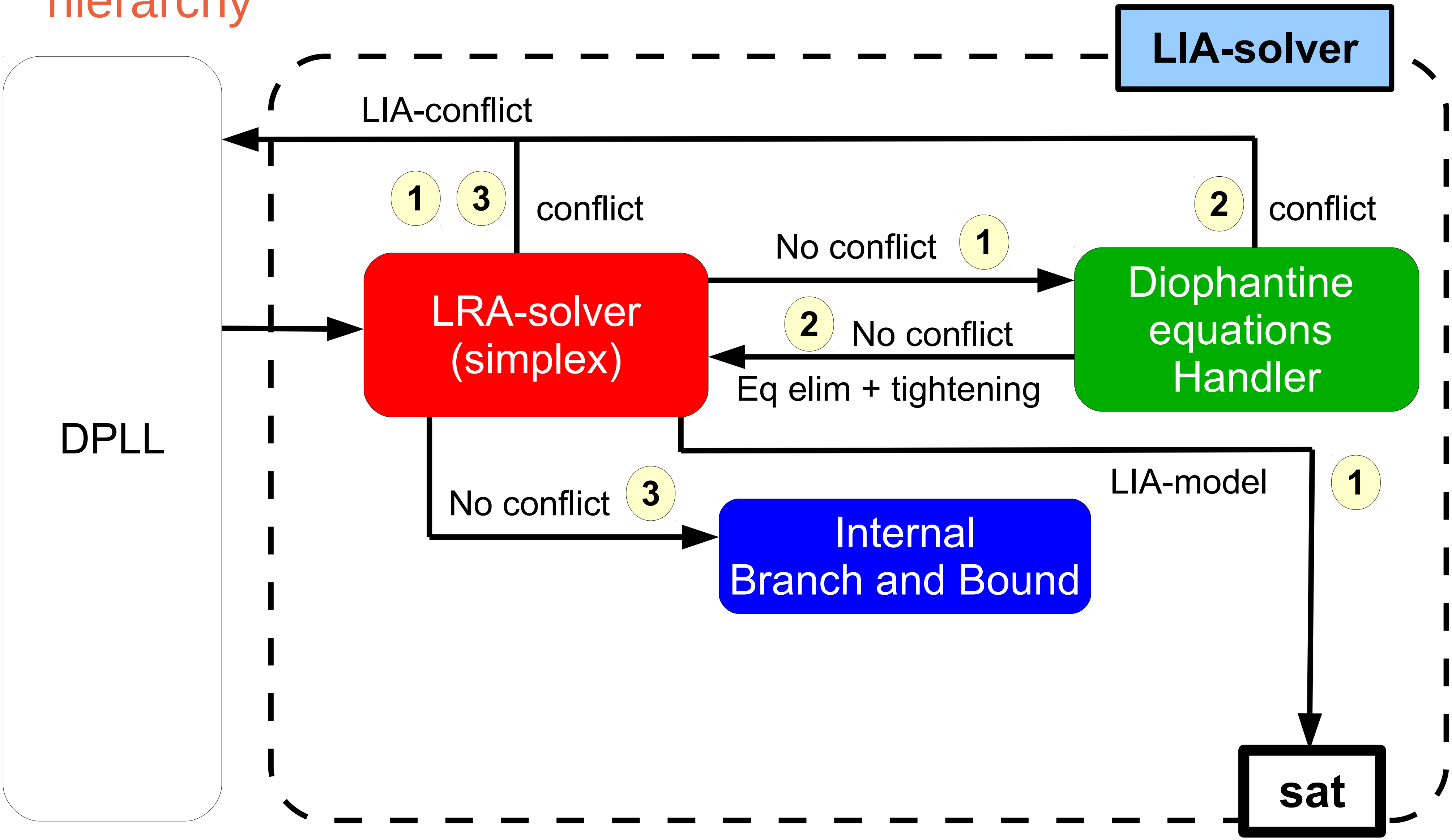
Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy



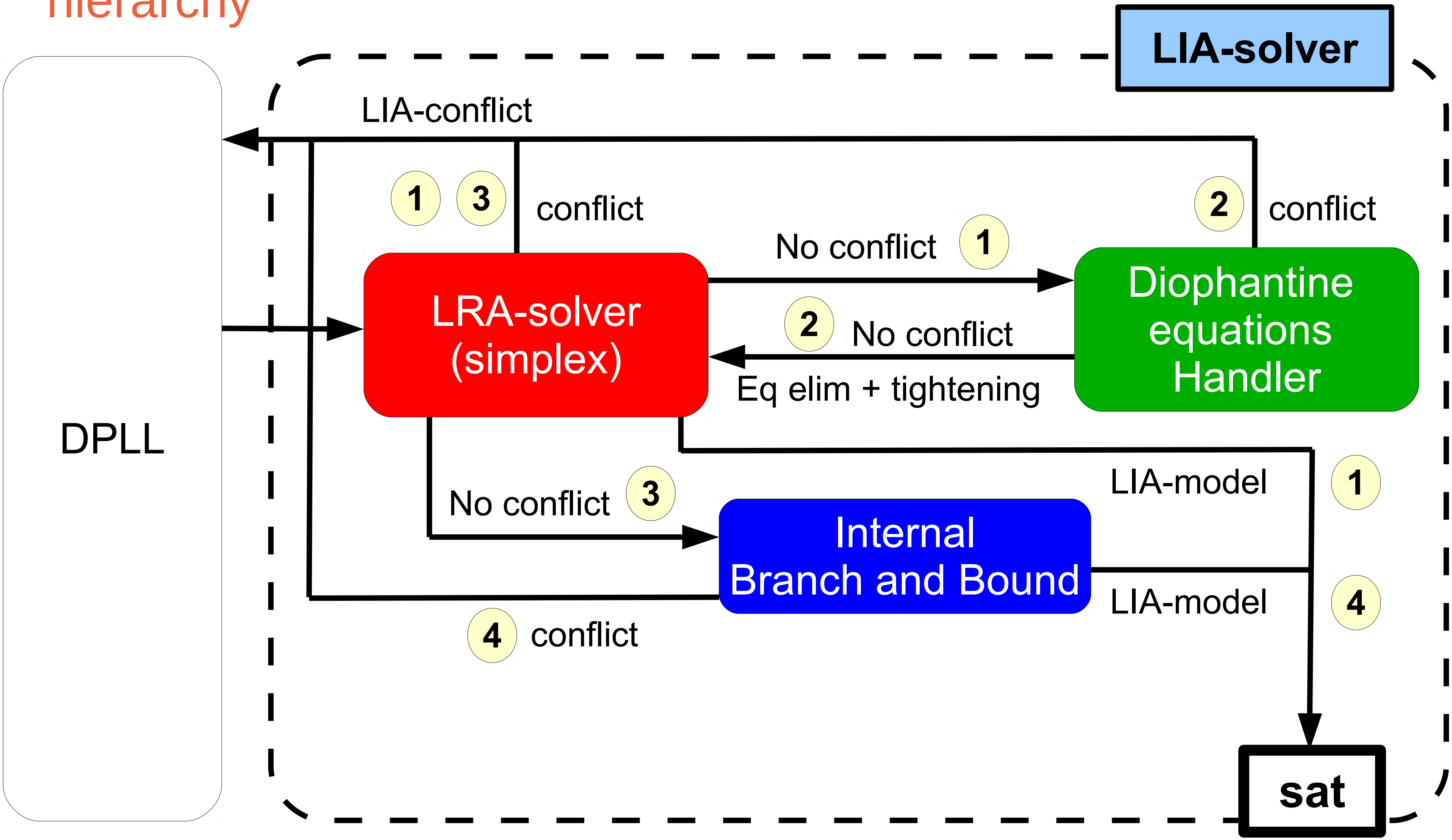
Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy



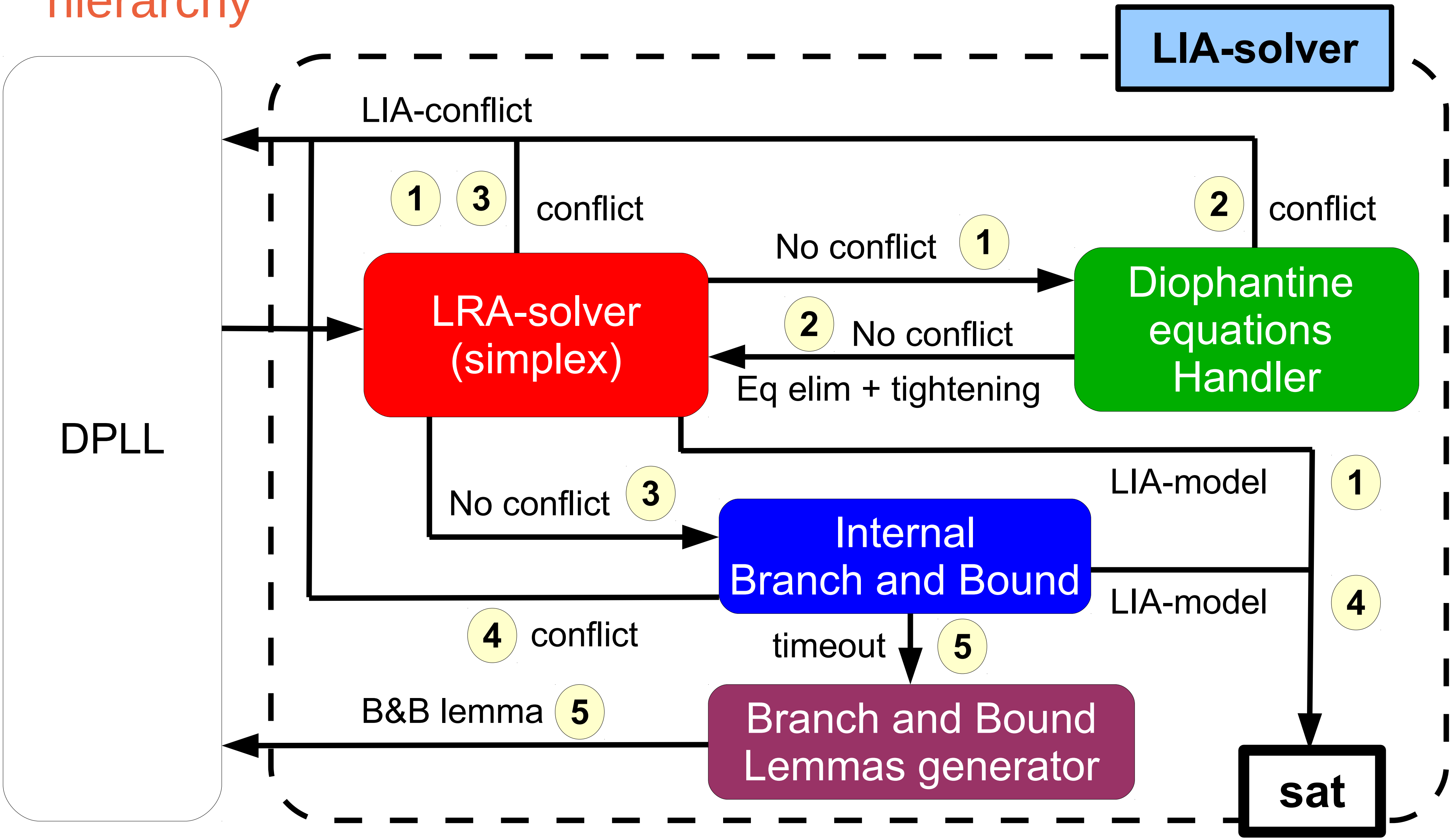
Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy



Layered architecture for LIA (MathSAT)

- Cooperation of several sub-modules organized as a layered hierarchy



The Diophantine equation handler

- Polynomial-time procedure for solving systems of equations in LA(Z) (Diophantine)
- Similar to the first part of the Omega test [Pug91]
- Extension of Gaussian elimination to integer constraints
  - Given $\sum_j a_{ij}x_j + a_{ik}x_k + c_i$ where $|a_{ik}|$ is the smallest:
    - Rewrite into $a_{ik} \cdot (x_k + \sum_{j \neq k} a^q_{ij}x_j + c^q_i) + (\sum_{j \neq k} a^r_{ij}x_j + c^r_i)$ where $\boxed{a_{ij} = a_{ik} \cdot a^q_{ij} + a^r_{ij}}$
    - Introduce a fresh var x_t and add the equation $x_t = x_k + \sum_{j \neq k} a^q_{ij}x_j + c^q_i$ to the system
    - Substitute x_k with $a_{ik}x_t + (\sum_{j \neq k} a^r_{ij}x_j + c^r_i)$ in the other equations
    - Return unsat when there is an equation $\sum_j a_{hj}x_j + c_h$ such that $\mathrm{GCD}(a_{h1}, \ldots a_{hn})$ does not divide c_h
    - Return sat when the system is in triangular form

The Diophantine equation handler

- Features:
  - Can generate *T*-lemmas when unsat is detected
    - Can actually generate detailed proofs of unsatisfiability as linear combinations of the input constraints
  - When sat is detected, the solution can be used to eliminate equalities
    - Allows for performing tightening of inequalities, with which the simplex solver can discover more conflicts
  - Can be performed incrementally with efficient backtracking

Tightening - Example

$$E := \begin{cases} 2x_1 - 5x_3 = 0 \\ x_2 - 3x_4 = 0 \end{cases} \qquad I := \begin{cases} -2x_1 - x_2 - x_3 \le -7 \\ 2x_1 + x_2 + x_3 \le 8 \end{cases}$$

- Give *E* to the Diophantine equations handler
- Obtain (parametric) solution $S := \begin{cases} x_1 = 2x_3 + t \\ x_2 = 3x_4 \\ x_3 = 2t \end{cases}$, *t* fresh
- Substitute x_1, x_2, x_3 in *I*, obtaining $I' := \begin{cases} -3x_4 - 12t \le -7 \\ 3x_4 + 12t \le 8 \end{cases}$
- Tighten, obtaining $I'' := \begin{cases} -\frac{3}{3}x_4 - \frac{12}{3}t \le \lfloor -\frac{7}{3} \rfloor \\ \frac{3}{3}x_4 + \frac{12}{3}t \le \lfloor \frac{8}{3} \rfloor \end{cases}$

Tightening - Example

$$E := \begin{cases} 2x_1 - 5x_3 = 0 \\ x_2 - 3x_4 = 0 \end{cases} \qquad I := \begin{cases} -2x_1 - x_2 - x_3 \leq -7 \\ 2x_1 + x_2 + x_3 \leq 8 \end{cases}$$

- Give *E* to the Diophantine equations handler
- Obtain (parametric) solution $S := \begin{cases} x_1 = 2x_3 + t \\ x_2 = 3x_4 \\ x_3 = 2t \end{cases}$, *t* fresh
- Substitute x_1, x_2, x_3 in *I*, obtaining $I' := \begin{cases} -3x_4 - 12t \leq -7 \\ 3x_4 + 12t \leq 8 \end{cases}$
- Tighten, obtaining $I'' := \begin{cases} -x_4 - 4t \leq -3 \\ x_4 + 4t \leq 2 \end{cases}$
- Give *I''* to the LRA-solver $\Longrightarrow$ **conflict**

Branch and Bound

- Given an LRA-model μ for the current set of constraints *S*
  - If there is an integer variable z such that $\mu(z) \notin \mathbb{Z}$
  - S is LIA-consistent iff either $S \cup \{(z \leq \lfloor \mu(z) \rfloor)\}$ or $S \cup \{(z \geq \lceil \mu(z) \rceil)\}$ is
- Branch and Bound idea: recursively solve subproblems

  $$S^i \cup \{(z_j \leq \lfloor \mu^i(z_j) \rfloor)\} \qquad S^i \cup \{(z_j \geq \lceil \mu^i(z_j) \rceil)\}$$

  until either a LIA-model is found or all of them are LA(Q)-inconsistent
- Implementation: a popular approach is to use "splitting on-demand"
  - Create new clause (lemma) $(z \leq \lfloor \mu(z) \rfloor) \vee (z \geq \lceil \mu(z) \rceil)$ and send it to DPLL, and continue searching

Branch and Bound with splitting on-demand

- Advantages:
  - Ease of implementation
    - No need to support case splits within the theory solver, can reuse DPLL
  - Exploit “for free” all the search space pruning techniques of modern DPLL solvers
    - Backjumping
    - Learning
    - …
- However, in our setting splitting on-demand has also some drawbacks

Splitting on-demand: drawbacks - 1

- Can not fully exploit equality elimination and tightening
  - New variables introduced by the Diophantine equations handler and tightened inequalities are invisible to DPLL
    - New variables can't be used in branch-and-bound lemmas
    - Tighened inequalities are thrown away when returning to DPLL

Splitting on-demand: drawbacks - 1

- Can not fully exploit equality elimination and tightening
  - New variables introduced by the Diophantine equations handler and tightened inequalities are invisible to DPLL
    - New variables can't be used in branch-and-bound lemmas
    - Tighened inequalities are thrown away when returning to DPLL

Example

$$S := \begin{cases} z = 2y - 3w \\ z + 3w \leq -1 \\ 2x + z \geq 1 \\ x + w \leq 0 \end{cases}$$

$$\mu_{LRA} := \begin{cases} x = \frac{2}{5} \\ y = -\frac{1}{2} \\ z = \frac{1}{5} \\ w = -\frac{2}{5} \end{cases}$$

Splitting on-demand: drawbacks - 1

- Can not fully exploit equality elimination and tightening
  - New variables introduced by the Diophantine equations handler and tightened inequalities are invisible to DPLL
    - New variables can't be used in branch-and-bound lemmas
    - Tighened inequalities are thrown away when returning to DPLL

Example
$$S := \begin{cases} z = 2y - 3w \\ z + 3w \leq -1 \\ 2x + z \geq 1 \\ x + w \leq 0 \end{cases}$$

After substitution of *z* and tightening:
$$S' := \begin{cases} y \leq -1 \\ 2x + 2y - 3w \geq 1 \\ x + w \leq 0 \end{cases}$$

$$\mu_{LRA} := \begin{cases} x = \frac{2}{5} \\ y = -\frac{1}{2} \\ z = \frac{1}{5} \\ w = -\frac{2}{5} \end{cases}$$

$$\mu'_{LRA} := \begin{cases} x = \frac{3}{5} \\ y = -1 \\ w = -\frac{3}{5} \end{cases}$$

Splitting on-demand: drawbacks - 1

- Can not fully exploit equality elimination and tightening
  - New variables introduced by the Diophantine equations handler and tightened inequalities are invisible to DPLL
    - New variables can't be used in branch-and-bound lemmas
    - Tighened inequalities are thrown away when returning to DPLL

Example

$$S := \begin{cases} z = 2y - 3w \\ z + 3w \leq -1 \\ 2x + z \geq 1 \\ x + w \leq 0 \end{cases}$$

After substitution of *z* and tightening:

$$S' := \begin{cases} y \leq -1 \\ 2x + 2y - 3w \geq 1 \\ x + w \leq 0 \end{cases}$$

If we branch on $(x \leq \lfloor \frac{3}{5} \rfloor)$ (i.e. $(x \leq 0)$)
Then the simplex finds a LIA-model for $S' \cup \{(x \leq 0)\}$

However, the model found for $S \cup \{(x \leq 0)\}$ is not good in LA(Z)

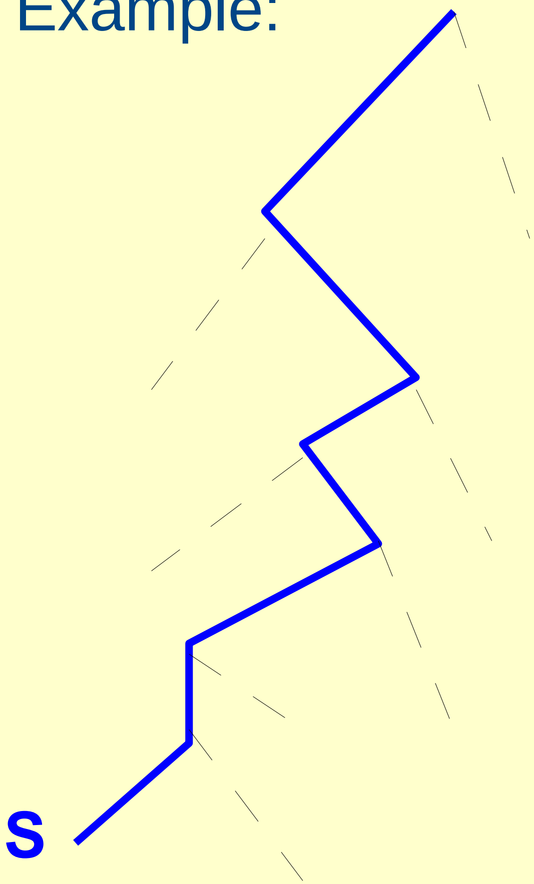
Splitting on-demand: drawbacks - 2

- Branch and bound lemmas might cease to be useful upon backtracking
  - Branch and bound aimed at finding a LIA-model for the constraints in the current branch
  - Splitting on-demand adds “global” lemmas
    - They might “pollute” the search space
    - Overhead in DPLL

Splitting on-demand: drawbacks - 3

- Non-chronological backtracking (backjumping) might hurt sometimes

  - Example:



- Set of constraints S in the current DPLL branch

  LA(Q)-model for S s.t. $\mu(x_k) = q_k \notin \mathbb{Z}$

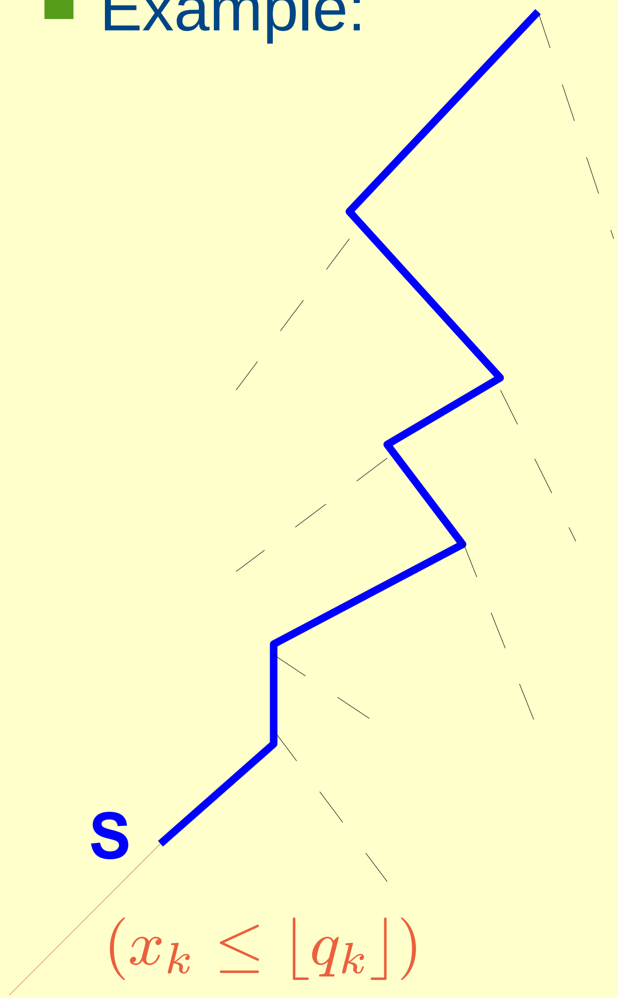
  Branch and bound lemma:

  $$(x_k \leq \lfloor q_k \rfloor) \vee (x_k \geq \lceil q_k \rceil)$$

  Suppose the LA(Q)-model for $S \cup \{(x_k \geq \lceil q_k \rceil)\}$ is also a LA(Z)-model, but we branch on $(x_k \leq \lfloor q_k \rfloor)$ first

Splitting on-demand: drawbacks - 3

- Non-chronological backtracking (backjumping) might hurt sometimes
  - Example:



- Set of constraints S in the current DPLL branch

  LA(Q)-model for S s.t. $\mu(x_k) = q_k \notin \mathbb{Z}$

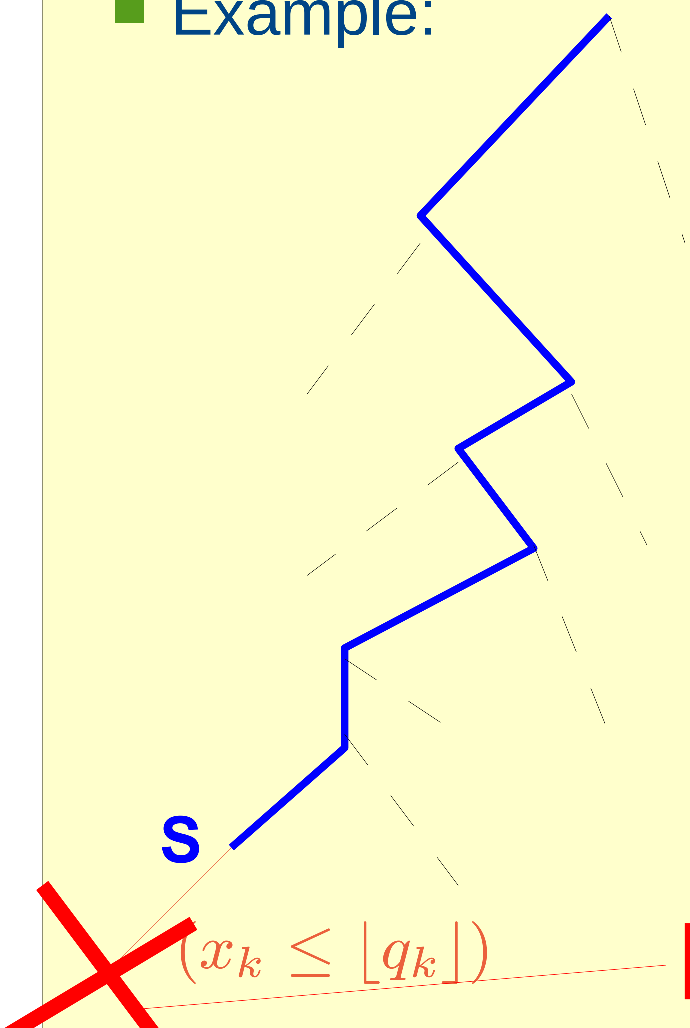
  Branch and bound lemma:

  $(x_k \leq \lfloor q_k \rfloor) \vee (x_k \geq \lceil q_k \rceil)$

  Suppose the LA(Q)-model for $S \cup \{(x_k \geq \lceil q_k \rceil)\}$ is also a LA(Z)-model, but we branch on $(x_k \leq \lfloor q_k \rfloor)$ first

Splitting on-demand: drawbacks - 3

- Non-chronological backtracking (backjumping) might hurt sometimes
  - Example:



Conflict!

- Set of constraints S in the current DPLL branch

LA(Q)-model for S s.t. $\mu(x_k) = q_k \notin \mathbb{Z}$

Branch and bound lemma:

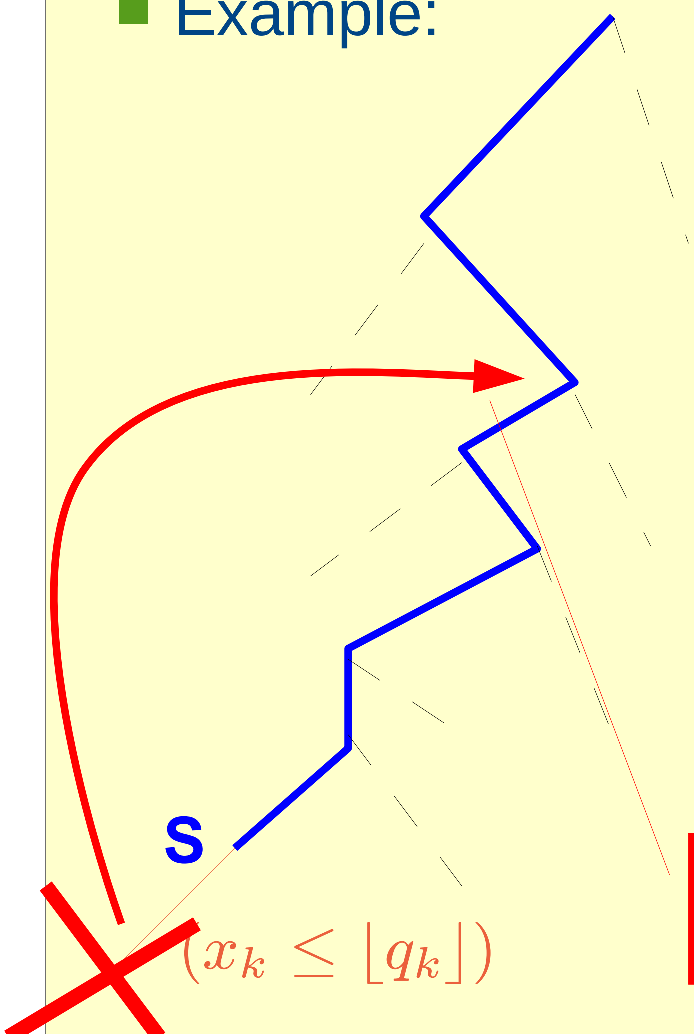
$$(x_k \leq \lfloor q_k \rfloor) \vee (x_k \geq \lceil q_k \rceil)$$

Suppose the LA(Q)-model for $S \cup \{(x_k \geq \lceil q_k \rceil)\}$ is also a LA(Z)-model, but we branch on $(x_k \leq \lfloor q_k \rfloor)$ first

Splitting on-demand: drawbacks - 3

- Non-chronological backtracking (backjumping) might hurt sometimes

  - Example:



- Set of constraints S in the current DPLL branch

LA(Q)-model for S s.t. $\mu(x_k) = q_k \notin \mathbb{Z}$

Branch and bound lemma:

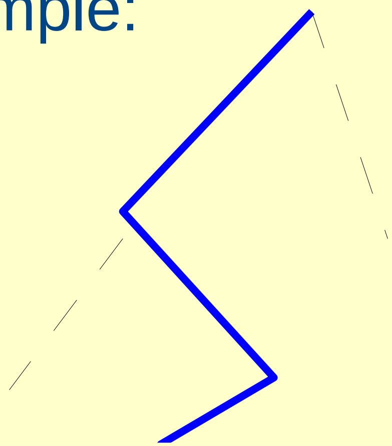
$$(x_k \leq \lfloor q_k \rfloor) \vee (x_k \geq \lceil q_k \rceil)$$

Suppose the LA(Q)-model for $S \cup \{(x_k \geq \lceil q_k \rceil)\}$ is also a LA(Z)-model, but we branch on $(x_k \leq \lfloor q_k \rfloor)$ first

Backjumping after conflict analysis

Splitting on-demand: drawbacks - 3

- Non-chronological backtracking (backjumping) might hurt sometimes
  - Example:



- Set of constraints S in the current DPLL branch

LA(Q)-model for S s.t. $\mu(x_k) = q_k \notin \mathbb{Z}$

Branch and bound lemma:

$$(x_k \leq \lfloor q_k \rfloor) \vee (x_k \geq \lceil q_k \rceil)$$

Suppose the LA(Q)-model for $S \cup \{(x_k \geq \lceil q_k \rceil)\}$ is also a LA(Z)-model, but we branch on $(x_k \leq \lfloor q_k \rfloor)$ first

After backjumping, might need to redo a lot of expensive computations (equality elimination, tightening) before finding S again

Splitting on-demand: drawbacks - 4

- Difficult to use dedicated heuristics for exploring the branch-and-bound search tree
  - Several sophisticated heuristics developed in the ILP community, crucial for performance
  - Might not be straightforward to integrate with those commonly used in DPLL

Internal branch and bound

- Possible solution (MathSAT): perform branch and bound search within the LIA-solver
  - Start from the result of equality elimination + tightening
  - Use dedicated heuristics for selecting the variables on which to branch
  - Do not backjump past the starting point
  - Remove redundant constraints before starting branch and bound
    - E.g. by exploiting polarity of variables
- Only perform a bounded (small) number of case splits, and then revert to splitting on-demand
  - Keep the benefits of splitting on-demand for hard problems

Bit-vectors (BV)

- Most solvers use an **eager approach** for BV, not DPLL(T)
  - **Heavy preprocessing** based on rewriting rules + SAT encoding ("bit-blasting")
    - Example: $$\begin{aligned}(x_{[1]} \neq 0_{[1]}) \wedge (y_{[31]} :: x_{[1]} \% 2_{[32]} = 0_{[32]}) &\mapsto \\ (x_{[1]} = 1_{[1]}) \wedge (y_{[31]} :: x_{[1]} \% 2_{[32]} = 0_{[32]}) &\mapsto \\ (y_{[31]} :: 1_{[1]} \% 2_{[32]} = 0_{[32]}) &\mapsto \bot\end{aligned}$$
- Alternative: **lazy bit-blasting**, compatible with DPLL(T)
  - Use a second SAT solver as T-solver for BV
    - bit-blast only BV-atoms, not the whole formula
  - Boolean skeleton of the formula handled by the main SAT solver
  - Easier integration with other T-solvers and DPLL(T)
  - Can integrate additional specialized sub-solvers
- Eager still better performance-wise

Lazy bit-blasting: implementation

- **For each BV-atom** α occurring in the input formula, create a fresh Boolean "label" variable l_α , and bit-blast to SAT-BV the formula $(l_\alpha \leftrightarrow \alpha)$

- **Exploit SAT solving under assumptions**
  - When the main solver generates the BV-assignment $\alpha_1 \ldots \alpha_n$
  - Invoke SAT-BV with the assumptions $l_{\alpha_1} \ldots l_{\alpha_n}$
  - If unsat, generate an unsat core of the assumptions $l_{\alpha_i} \ldots l_{\alpha_j}$
    - From its negation, generate a BV-lemma $\neg\alpha_i \vee \ldots \vee \neg\alpha_j$ and send it to the main solver as a blocking clause, like in standard DPLL(T)

SAT solving under assumptions

Modern CDCL-based SAT solvers allow for solving a CNF formula φ under assumptions on the values of some literals $\{l_1, l_2, \ldots, l_n\}$

- Logically equivalent to checking $\varphi \wedge \bigwedge_i l_i$
- But $l_1, \ldots, l_n$ are assumed only temporarily
  - A limited but very useful form of incremental solving

- If φ is unsat under the assumptions $\{l_1, l_2, \ldots, l_n\}$ we can ask the SAT solver to compute an unsatisfiable core of the assumptions
  - A subset $\{l_j, \ldots, l_k\} \subseteq \{l_1, \ldots, l_n\}$ that is sufficient for proving unsatisfiability

SAT under assumptions - implementation

- Modify the branching heuristics of the CDCL solver to always pick the next unassigned literal from $\{l_1, l_2, \ldots, l_n\}$ before other literals
  - The first n decision levels of the trail always correspond to the assumptions
- If an assumption literal is assigned to false, return unsat
  - Can only happen by unit propagation at a level < n
- Unsat core: start conflict analysis from the falsified assumption literal $\neg l_j$, and use the “decision” strategy to collect all the involved assumptions

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$
$c_2 : A_1 \vee A_5 \vee A_4$
$c_3 : A_4 \vee \neg A_5$
$c_4 : A_3 \vee A_5 \vee \neg A_4$
$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$
$\neg A_2$
$\neg A_3$

Trail

Lit	Reason

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$
A_4	$-$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$
A_4	$-$
A_5	c_4

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$
$c_2 : A_1 \vee A_5 \vee A_4$
$c_3 : A_4 \vee \neg A_5$
$c_4 : A_3 \vee A_5 \vee \neg A_4$
$c_5 : \neg A_4 \vee \neg A_5$

Assumptions

$\neg A_1$
$\neg A_2$
$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$
A_4	$-$
A_5	c_4

$$\frac{c_5 : \neg A_4 \vee \neg A_5 \qquad c_4 : A_3 \vee A_5 \vee \neg A_4}{c_6^l : A_3 \vee \neg A_4}$$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$
$\neg A_4$	c_6^l

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$
$\neg A_4$	c_6^l
A_5	c_2

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
$\neg A_3$	$-$
$\neg A_4$	c_6^l
A_5	c_2

$$\frac{c_3 : A_4 \vee \neg A_5 \qquad c_2 : A_1 \vee A_5 \vee A_4}{c_7^l : A_1 \vee A_4}$$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
A_4	c_7^l

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\boxed{\neg A_3}$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
A_4	c_7^l
$\boxed{A_3}$	c_6^l

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
A_4	c_7^l
A_3	c_6^l

$c_6^l : A_3 \vee \neg A_4$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
A_4	c_7^l
A_3	c_6^l

$c_6^l : A_3 \vee \neg A_4$ $\quad$ $c_7^l : A_1 \vee A_4$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\neg A_1$

$\neg A_2$

$\neg A_3$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
A_4	c_7^l
A_3	c_6^l

$$\frac{c_6^l : A_3 \vee \neg A_4 \qquad c_7^l : A_1 \vee A_4}{A_1 \vee A_3}$$

Example

Input clauses

$c_1 : \neg A_2 \vee A_1 \vee A_5$

$c_2 : A_1 \vee A_5 \vee A_4$

$c_3 : A_4 \vee \neg A_5$

$c_4 : A_3 \vee A_5 \vee \neg A_4$

$c_5 : \neg A_4 \vee \neg A_5$

$c_6^l : A_3 \vee \neg A_4$

$c_7^l : A_1 \vee A_4$

Assumptions

$\boxed{\neg A_1}$

$\neg A_2$

$\boxed{\neg A_3}$

Trail

Lit	Reason
$\neg A_1$	$-$
$\neg A_2$	$-$
A_4	c_7^l
A_3	c_6^l

$$\frac{c_6^l : A_3 \vee \neg A_4 \qquad c_7^l : A_1 \vee A_4}{\boxed{A_1} \vee \boxed{A_3}}$$

Arrays (A)

- **Read** (rd) and **write** (wr) operations over arrays
- **Equality** over array variables (extensionality)
- Example: $\mathsf{wr}(a, i, x) = \mathsf{wr}(b, i, \mathsf{rd}(a, j, y)) \land \neg(a = b)$
- Common approach: reduction to EUF via **lazy axiom instantiation**
  - read-over-write: $\forall a.\forall i.\forall x.(\mathsf{rd}(\mathsf{wr}(a, i, x), i) = x)$
    $\forall a.\forall i.\forall j.\forall x.((i \neq j) \rightarrow \mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j))$
  - extensionality: $\forall a.\forall b.((a \neq b) \rightarrow \exists i.(\mathsf{rd}(a, i) \neq \mathsf{rd}(b, i))$
  - Add **lemmas on-demand** by instantiating the quantified variables with terms occurring in the input formula
    - Using smart “frugal” strategies: **check** candidate solution, instantiate only **(potentially) violated axioms**

Example

$$\neg(j = k), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j)), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, i, j\}, \{k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Example

$$\neg(j = k), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j)), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, i, j\}, \{k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Add violated lemma: $(i \neq k) \rightarrow (\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$

Example

$$\neg(j = k), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j)), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, i, j\}, \{k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Add violated lemma: $(i \neq k) \rightarrow (\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, j\}, \{i, k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Example

$$\neg(j = k), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j)), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, i, j\}, \{k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Add violated lemma: $(i \neq k) \rightarrow (\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\boxed{\mathsf{rd}(\mathsf{wr}(a, i, x), j)}\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, \boxed{j}\}, \{\boxed{i,}\ k\}, \{\boxed{\mathsf{rd}(a, j)}\}, \{\mathsf{rd}(a, k)\}$

Add violated lemma: $(i \neq j) \rightarrow (\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j))$

Example

$$\neg(j = k), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j)), \neg(\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, i, j\}, \{k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Add violated lemma: $(i \neq k) \rightarrow (\mathsf{rd}(\mathsf{wr}(a, i, x), k) = \mathsf{rd}(a, k))$

EUF solution (equivalence classes):

$\{a, \mathsf{wr}(a, i, x))\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), j)\}, \{\mathsf{rd}(\mathsf{wr}(a, i, x), k)\},$
$\{x, j\}, \{i, k\}, \{\mathsf{rd}(a, j)\}, \{\mathsf{rd}(a, k)\}$

Add violated lemma: $(i \neq j) \rightarrow (\mathsf{rd}(\mathsf{wr}(a, i, x), j) = \mathsf{rd}(a, j))$

EUF solver returns UNSAT

Outline

Introduction

CDCL-based SAT solvers

The DPLL(T) architecture

Some relevant T-solvers

Combination of theories

Combination of theories

- Very often in practice more than one theory is needed
  - Example (from intro):

$$\varphi \stackrel{\text{def}}{=} (x_1 \geq 0) \wedge (x_1 < 1) \wedge$$
$$((f(x_1) = f(0)) \rightarrow (\mathsf{rd}(\mathsf{wr}(P, x_2, x_3), x_2 + x_1) = x_3 + 1))$$

- How to build solvers for SMT(T_1 … T_n) that are both **efficient** and **modular**?
  - Can we **reuse** T_i-solvers and **combine** them?
  - Under what conditions?
  - How do we go from DPLL(T) to DPLL(T_1 … T_n)?

The Nelson-Oppen method

- A **general technique** for combining T_i-solvers
- Requires:
  - T_i's to have disjoint signatures, i.e. no symbols in common (other than $=$)
  - T_i's to be stably-infinite, i.e. every quantifier-free T_i-satisfiable formula is satisfiable in an infinite model of T_i
    - Examples: EUF, LIA, LRA, A
    - Counterexample: BV
    - *(Extensions exist to deal with some non-stably-infinite theories)*

The Nelson-Oppen method

How it works (for $T_1 \cup T_2$)

- Preprocessing **purification** step on the input formula φ
  - Pure formula: no atom containing symbols of different T_i's (except $=$)
  - By labeling subterms
  - Example:

$$\varphi \stackrel{\text{def}}{=} f(\overbrace{x+3y}^{l_1})+4 \leq \overbrace{g(w)}^{l_2} \mapsto$$

$$(\overbrace{f(l_1)}^{l_3}+4 \leq l_2) \wedge (l_1 = x+3y) \wedge (l_2 = g(w)) \mapsto$$

$$(l_3+4 \leq l_2) \wedge (l_1 = x+3y) \wedge (l_2 = g(w)) \wedge (f(l_1) = l_3)$$

The Nelson-Oppen method

How it works (for $T_1 \cup T_2$)

- Preprocessing **purification** step on the input formula φ
  - Pure formula: no atom containing symbols of different T_i's (except $=$)
  - By labeling subterms
  - Example:

$$\varphi \stackrel{\text{def}}{=} f(\overbrace{x+3y}^{l_1})+4 \leq \overbrace{g(w)}^{l_2} \mapsto$$

$$(\overbrace{f(l_1)}^{l_3}+4 \leq l_2) \wedge (l_1 = x+3y) \wedge (l_2 = g(w)) \mapsto$$

$$(l_3+4 \leq l_2) \wedge (l_1 = x+3y) \wedge (l_2 = g(w)) \wedge (f(l_1) = l_3)$$

- T_i-solvers cooperate by **exchanging (disjunctions of) entailed interface equalities**
  - I.e., equalities between shared variables

The Nelson-Oppen method

How it works (for $T_1 \cup T_2$)

- Preprocessing **purification** step on the input formula φ
  - Pure formula: no atom containing symbols of different T_i's (except $=$)
  - By labeling subterms
  - Example:

$$\varphi \stackrel{\text{def}}{=} f(\overbrace{x+3y}^{l_1}) + 4 \leq \overbrace{g(w)}^{l_2} \mapsto$$

$$(\overbrace{f(l_1)}^{l_3} + 4 \leq l_2) \wedge (l_1 = x + 3y) \wedge (l_2 = g(w)) \mapsto$$

$$(\boxed{l_3} + 4 \leq \boxed{l_2}) \wedge (\boxed{l_1} = x + 3y) \wedge (\boxed{l_2} = g(w)) \wedge (f(\boxed{l_1}) = \boxed{l_3})$$

Interface variables

- T_i-solvers cooperate by **exchanging (disjunctions of) entailed interface equalities**
  - I.e., equalities between shared variables

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$$\models$$

$$(x_1 = x_3) \vee (x_1 = x_4)$$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$$\models$$

$$(x_1 = x_3) \vee (x_1 = x_4)$$

$\neg(f(x_1) = f(x_2))$, EUF
$\neg(f(x_2) = f(x_4))$,
$(f(x_3) = x_5), (f(x_1) = x_6)$

$$(x_1 = x_3)$$

$$\models$$

$$(x_5 = x_6)$$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

$$\models$$

$$(x_1 = x_3) \vee (x_1 = x_4)$$

$$(x_1 = x_3)$$

$$\models$$

$$(x_5 = x_6)$$

$$(x_5 = x_6)$$

$$\models$$

$$(x_2 = x_3) \vee (x_2 = x_4)$$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

$$\models$$

$$(x_1 = x_3) \vee (x_1 = x_4)$$

$$(x_1 = x_3)$$

$$\models$$

$$(x_5 = x_6)$$

$$(x_5 = x_6)$$

$$\models$$

$$(x_2 = x_3) \vee (x_2 = x_4)$$

$$(x_2 = x_3)$$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

$$\models$$

$$(x_1 = x_3) \vee (x_1 = x_4)$$

$$(x_1 = x_3)$$

$$\models$$

$$(x_5 = x_6)$$

$$(x_5 = x_6)$$

$$\models$$

$$(x_2 = x_3) \vee (x_2 = x_4)$$

$$(x_2 = x_3) \quad (x_2 = x_4)$$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

$$\models$$

$$(x_1 = x_3) \vee (x_1 = x_4)$$

$\Rightarrow$

$(x_1 = x_3)$ $\quad$ $(x_1 = x_4)$

$$\models$$

$(x_5 = x_6)$

No more deductions possible

$\Leftarrow$

$$(x_5 = x_6)$$

$$\models$$

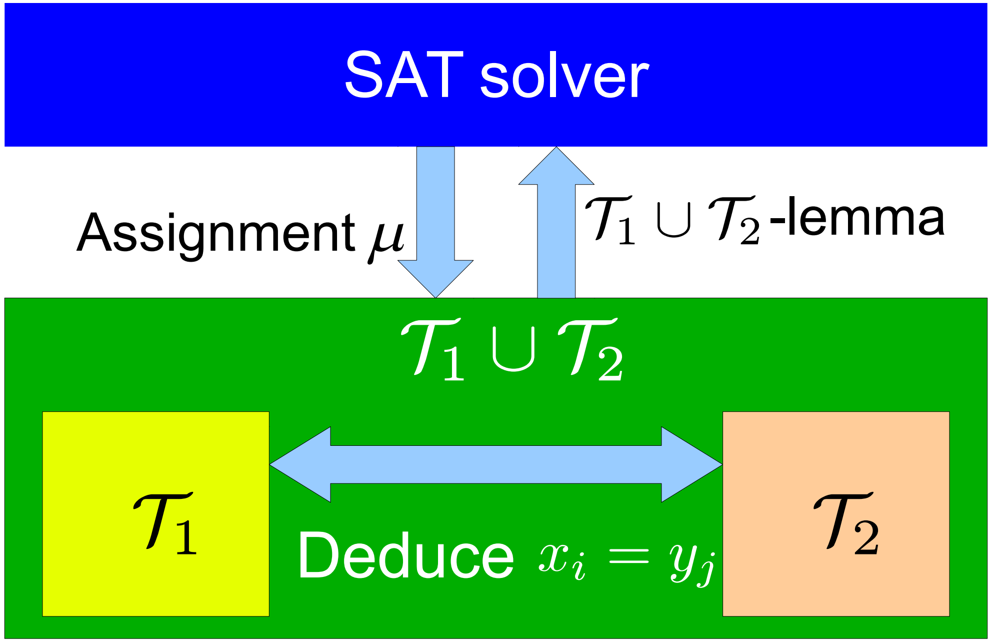
$$(x_2 = x_3) \vee (x_2 = x_4)$$

$\Rightarrow$

$(x_2 = x_3)$ $\quad$ $(x_2 = x_4)$

DPLL(T) for combined theories

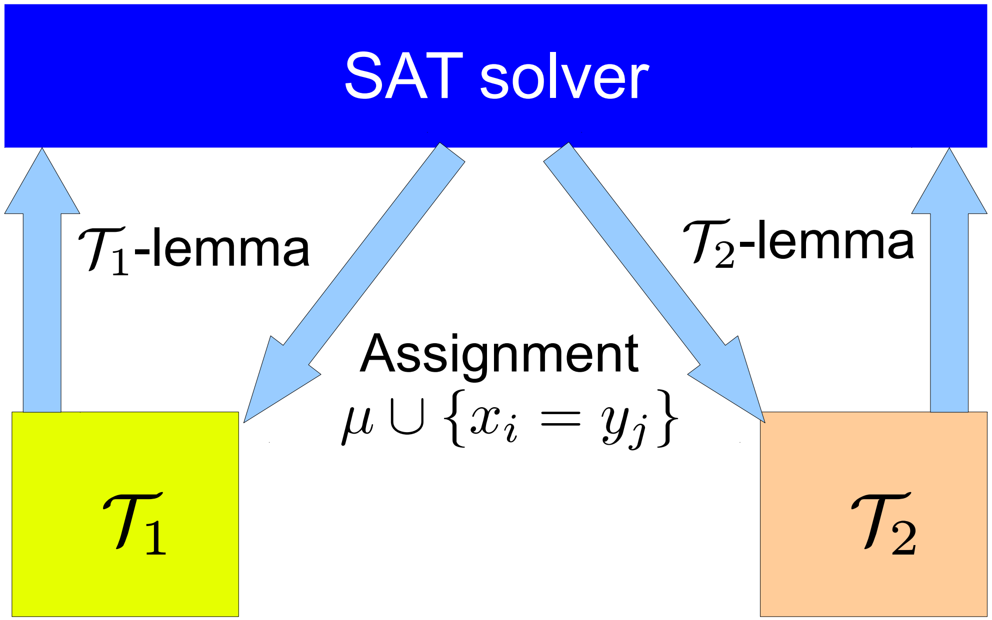
- Traditional approach: a **single** combined Nelson-Oppen T-solver
  - T_i-solvers exchange (disjunctions of) implied interface equalities **internally**
    - Interface equalities invisible to the SAT solver



- Drawbacks: T_i-solvers need to:
  - be deduction complete for interface equalities
  - be able to perform case splits internally

Delayed Theory Combination

- Alternative to traditional approach
  - Each T_i-solver interacts directly and only with the SAT solver
  - SAT solver takes care of (all or part of) the combination
    - Augment the Boolean search space with the possible interface equalities $(x_i = y_j)$
- Advantages:
  - No need of complete deduction of interface equalities
  - Case analysis via splitting on-demand



Delayed theory combination in practice

- **Model-based** heuristic:
  - Initially, no interface equalities generated
  - When a solution is found, check against all the possible interface equalities
    - If T_1 and T_2 agree on the implied equalities, return **SAT**
    - Otherwise, branch on equalities implied by T_1-model but not by T_2-model
  - Optimistic approach, similar to axiom instantiation

- Still allow T_i-solvers to **exchange equalities internally**
  - But no requirement of completeness
  - Avoids “polluting” the SAT space with equality deductions leading to conflicts

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2))$, EUF
$\neg(f(x_2) = f(x_4))$,
$(f(x_3) = x_5), (f(x_1) = x_6)$

LIA-model: $x_1 \mapsto 1 \quad x_2 \mapsto 2$
$x_3 \mapsto 0 \quad x_4 \mapsto 1$
$x_5 \mapsto 0 \quad x_6 \mapsto 1$

$\models$

$(x_1 = x_4)$

EUF-model: $\{x_1\} \; \{x_2\} \; \{x_3\} \; \{x_4\}$
$\{x_5, f(x_3)\} \; \{x_6, f(x_1)\}$
$\{f(x_2)\} \; \{f(x_4)\}$

$\not\models$

$(x_1 = x_4)$

Branch on $(x_1 = x_4)$

$(x_1 = x_4)$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

LIA-model:
$$\begin{array}{ll} x_1 \mapsto 1 & x_2 \mapsto 2 \\ x_3 \mapsto 0 & x_4 \mapsto 1 \\ x_5 \mapsto 0 & x_6 \mapsto 1 \end{array}$$

$$\models$$

$(x_3 = x_5)$
$(x_4 = x_6)$

EUF-model:
$$\begin{array}{l} \{x_1, x_4\}\ \{x_2\}\ \{x_3\} \\ \{x_5, f(x_3)\} \\ \{x_6, f(x_1), f(x_4)\}\{f(x_2)\} \end{array}$$

$$\not\models$$

$(x_3 = x_5)$
$(x_4 = x_6)$

...

$(x_1 = x_4)$

$(x_3 = x_5)$

$(x_4 = x_6)$

Example

LIA $(x_1 \geq 0), (x_1 \leq 1), (x_2 \geq x_6)$
$(x_2 \leq x_6 + 1), (x_5 = x_4 - 1)$
$(x_3 = 0), (x_4 = 1)$

LIA-model: $x_1 \mapsto 1 \quad x_2 \mapsto 2$
$x_3 \mapsto 0 \quad x_4 \mapsto 1$
$x_5 \mapsto 0 \quad x_6 \mapsto 1$

$\neg(f(x_1) = f(x_2)),$ EUF
$\neg(f(x_2) = f(x_4)),$
$(f(x_3) = x_5), (f(x_1) = x_6)$

EUF-model: $\{x_1, x_4, x_6, f(x_1), f(x_4)\}$
$\{x_3, x_5, f(x_3)\}$
$\{f(x_2)\}\{x_2\}$

...

$(x_1 = x_4)$

$(x_3 = x_5)$

$(x_4 = x_6)$

Selected bibliography

DISCLAIMER: this is not meant to be complete, just a starting point. Apologies to missing authors/works

- SMT in general and DPLL(T)
  - Nieuwenhuis, Oliveras, Tinelli. **Solving SAT and SAT Modulo Theories: From an abstract Davis--Putnam--Logemann--Loveland procedure to DPLL(T)**. J. ACM 2006
  - Sebastiani. **Lazy Satisfiability Modulo Theories.** JSAT 2007
  - Barrett, Sebastiani, Seshia, Tinelli. **Satisfiability Modulo Theories.** SAT handbook 2009
- Theory solvers
  - Detlefs, Nelson, Saxe. **Simplify: a theorem prover for program checking**. J. ACM 2005
  - Nieuwenhuis, Oliveras. **Fast congruence closure and extensions**. Inf. Comput. 2007

Selected bibliography

- Theory solvers (cont'd)

  - Dutertre, de Moura. **A Fast Linear-Arithmetic Solver for DPLL(T)**. CAV 2006
  - de Moura, Bjørner. **Model-based Theory Combination**. Electr. Notes Theor. Comput. Sci. 2008
  - Brummayer, Biere. **Lemmas on Demand for the Extensional Theory of Arrays**. JSAT 2009
  - de Moura, Bjørner. **Generalized, efficient array decision procedures**. FMCAD 2009
  - Jovanovic, de Moura. **Cutting to the Chase - Solving Linear Integer Arithmetic**. J. Autom. Reasoning 2013
  - Hadarean, Bansal, Jovanovic, Barrett, Tinelli. **A Tale of Two Solvers: Eager and Lazy Approaches to Bit-Vectors**. CAV 2014

Thank You

VTSA summer school 2015

Exploiting SMT for Verification of Infinite-State Systems

2. Interpolation in SMT and in Verification

Alberto Griggio
Fondazione Bruno Kessler – Trento, Italy

Outline

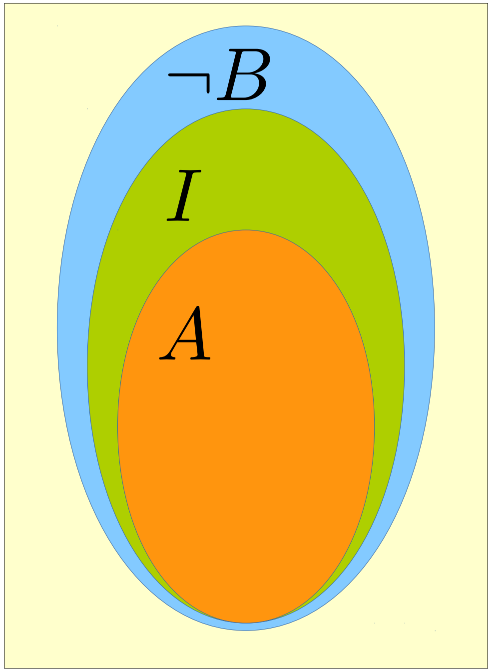
Introduction

Interpolants in Formal Verification

Computing interpolants in SMT

Introduction

- (Craig) Interpolant for an ordered pair (A, B) of formulae s.t. $A \wedge B \models_T \bot$ (or: $A \models_T \neg B$) is a formula I s.t.
  - $A \models_T I$
  - $I \wedge B \models_T \bot$ $(I \models_T \neg B)$
  - All the uninterpreted (in T) symbols of I are shared between A and B



- Why are interpolants useful?
  - Overapproximation of A relative to B
    - Overapprox. of $\exists_{\{x \notin B\}} \vec{x}.A$
  - “Local” explanation of why A is inconsistent with B

Importance of interpolation

Several important applications in formal verification:

- Approximate image computation for model checking of infinite-state systems
- Predicate discovery for Counterexample-Guided Abstraction Refinement
- Approximation of transition relation for infinite-state systems
- An alternative to (lazy) predicate abstraction for program verification
- Automatic generation of loop invariants
- ...

Outline

Introduction

Interpolants in Formal Verification

Computing interpolants in SMT

Background

Symbolic transition systems

- State variables X
- Initial states formula $I(X)$
- Transition relation formula $T(X, X')$

- A state σ is an assignment to the state vars $\bigwedge_{x_i \in X} x_i = v_i$
- A path of the system S is a sequence of states $\sigma_0, \ldots, \sigma_k$ such that $\sigma_0 \models I$ and $\sigma_i, \sigma'_{i+1} \models T$
- A *k*-step (symbolic) unrolling of S is a formula

$$I(X^0) \wedge \bigwedge_{i=0}^{k-1} T(X^i, X^{i+1})$$

  - Encodes all possible paths of length up to *k*
- A state property is a formula P over X
  - Encodes all the states σ such that $\sigma \models P$

Forward reachability checking

- Forward image computation
  - Compute all states reachable from σ in one transition:

$$\mathrm{Img}(\sigma(X)) := \exists X.\sigma(X) \wedge T(X, X')[X/X']$$

- Prove that a set of states $\mathrm{Bad}(X)$ is not reachable:

$$R(X) := I(X)$$

$$\mathrm{Bad}(X)$$

$$\mathrm{Img}(R(X))$$

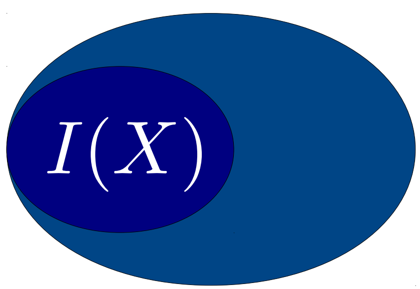
Forward reachability checking

- Forward image computation
  - Compute all states reachable from σ in one transition:

$$\mathrm{Img}(\sigma(X)) := \exists X.\sigma(X) \wedge T(X, X')[X/X']$$

- Prove that a set of states $\mathrm{Bad}(X)$ is not reachable:

$R(X) :=$



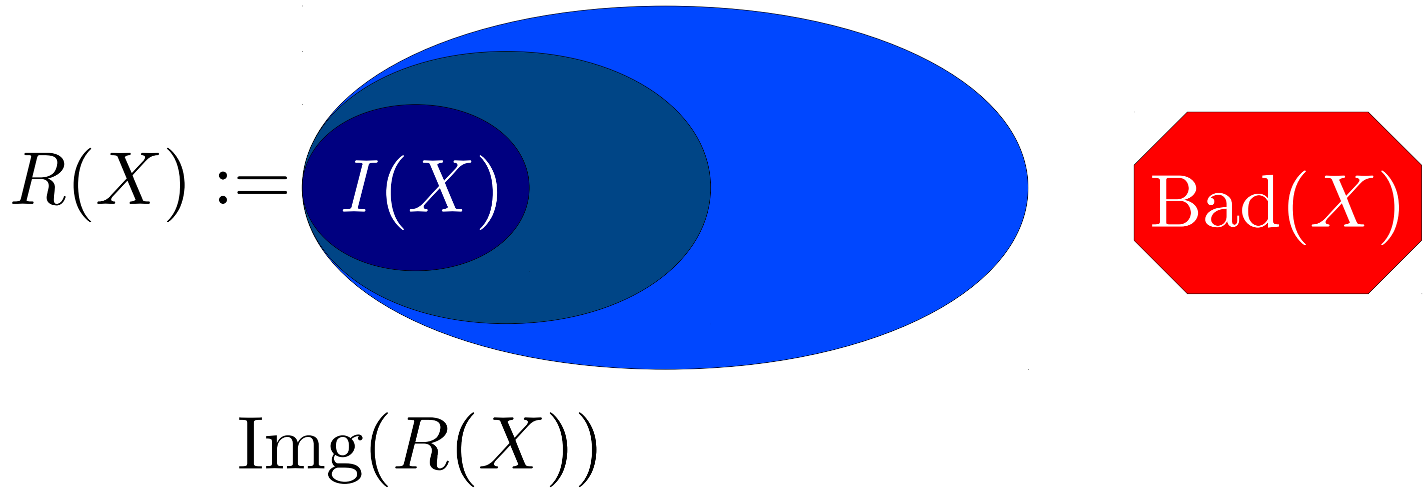
$\mathrm{Img}(R(X))$

Forward reachability checking

- Forward image computation
  - Compute all states reachable from σ in one transition:

$$\mathrm{Img}(\sigma(X)) := \exists X.\sigma(X) \wedge T(X, X')[X/X']$$

- Prove that a set of states $\mathrm{Bad}(X)$ is not reachable:

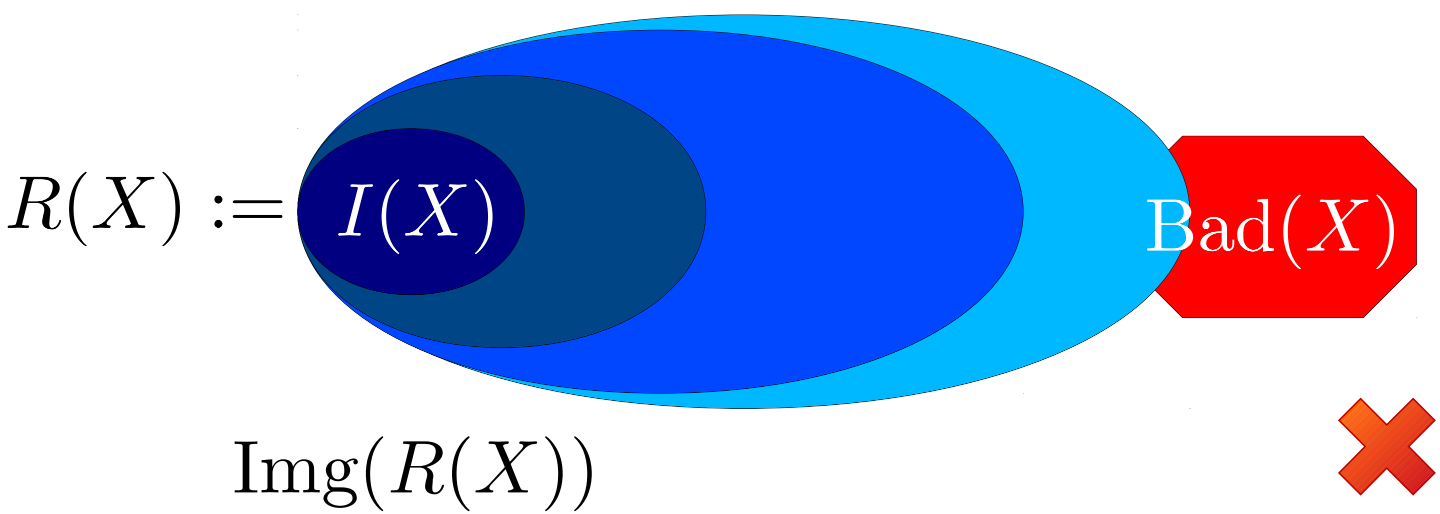


Forward reachability checking

- Forward image computation
  - Compute all states reachable from σ in one transition:

$$\boxed{\mathrm{Img}(\sigma(X)) := \exists X.\sigma(X) \wedge T(X, X')[X/X']}$$

- Prove that a set of states $\mathrm{Bad}(X)$ is not reachable:

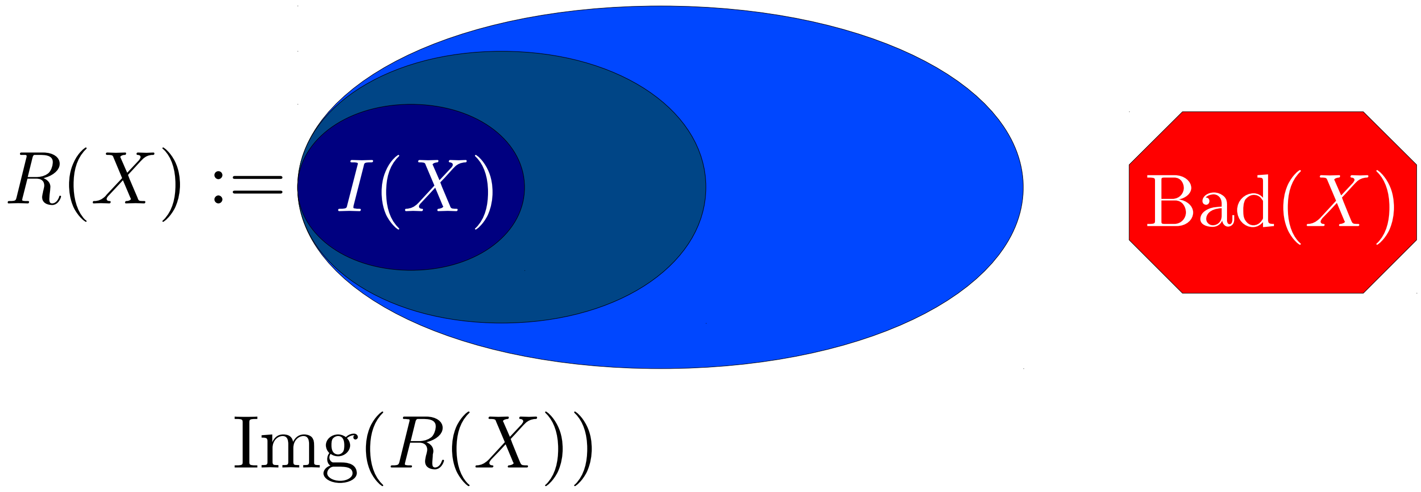


Forward reachability checking

- Forward image computation
  - Compute all states reachable from σ in one transition:

$$\mathrm{Img}(\sigma(X)) := \exists X.\sigma(X) \wedge T(X, X')[X/X']$$

- Prove that a set of states $\mathrm{Bad}(X)$ is not reachable:

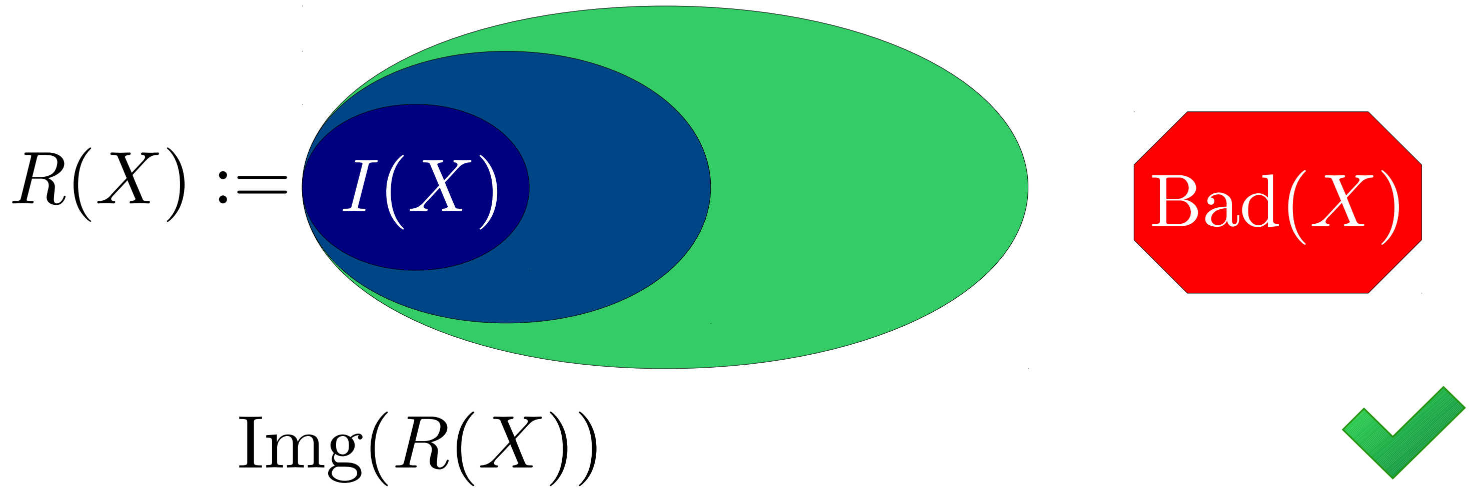


Forward reachability checking

- Forward image computation
  - Compute all states reachable from σ in one transition:

$$\mathrm{Img}(\sigma(X)) := \exists X.\sigma(X) \wedge T(X, X')[X/X']$$

- Prove that a set of states $\mathrm{Bad}(X)$ is not reachable:



Interpolation-based reachability

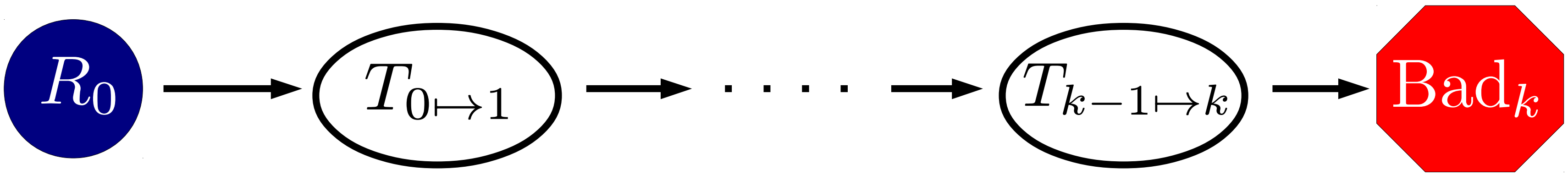
- Image computation requires quantifier elimination, which is typically very expensive (both in theory and in practice)

- Interpolation-based algorithm (McMillan CAV'03): use interpolants to overapproximate image computation
  - much more efficient than the previous algorithm
    - interpolation is often much cheaper than quantifier elimination
    - abstraction (overapproximation) accelerates convergence

  - termination is still guaranteed for finite-state systems

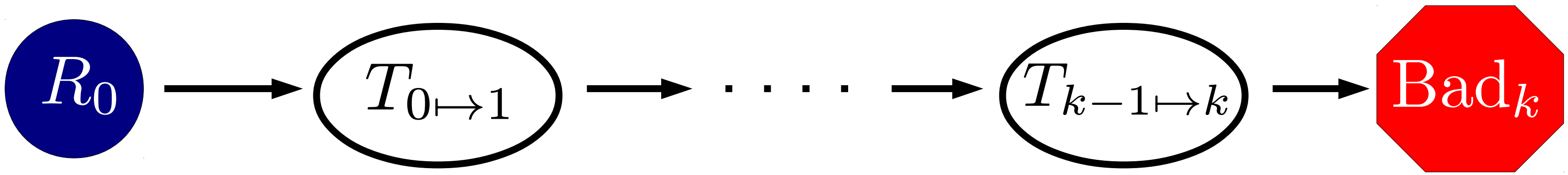
Interpolation-based reachability

- Set $R(X) := I(X)$
- Check satisfiability of $R_0 \wedge \bigwedge_{i=0}^{k-1} T_i \wedge \mathrm{Bad}_k$



Interpolation-based reachability

- Set $R(X) := I(X)$
- Check satisfiability of $R_0 \wedge \bigwedge_{i=0}^{k-1} T_i \wedge \mathrm{Bad}_k$

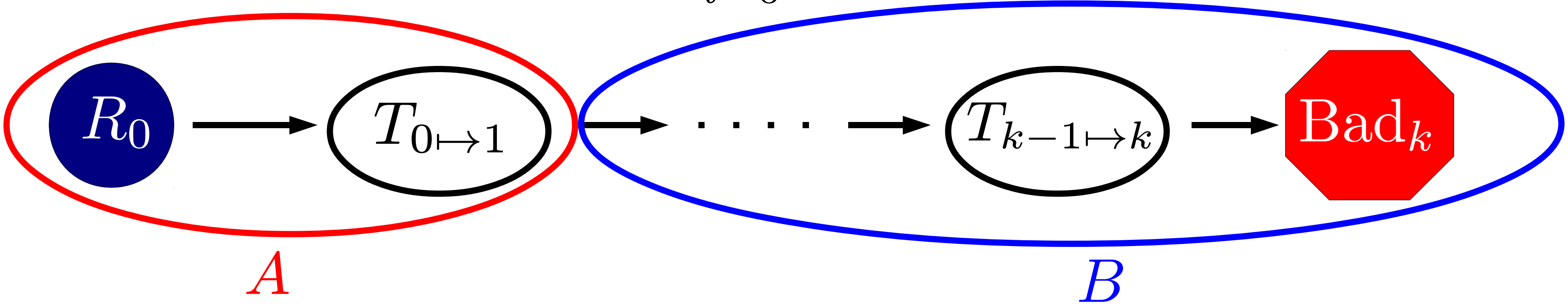


- If SAT:
  - If $R \equiv I$, return REACHABLE
  - else, increase *k* and repeat

the unrolling hits Bad

Interpolation-based reachability

- Set $R(X) := I(X)$
- Check satisfiability of $R_0 \wedge \bigwedge_{i=0}^{k-1} T_i \wedge \mathrm{Bad}_k$

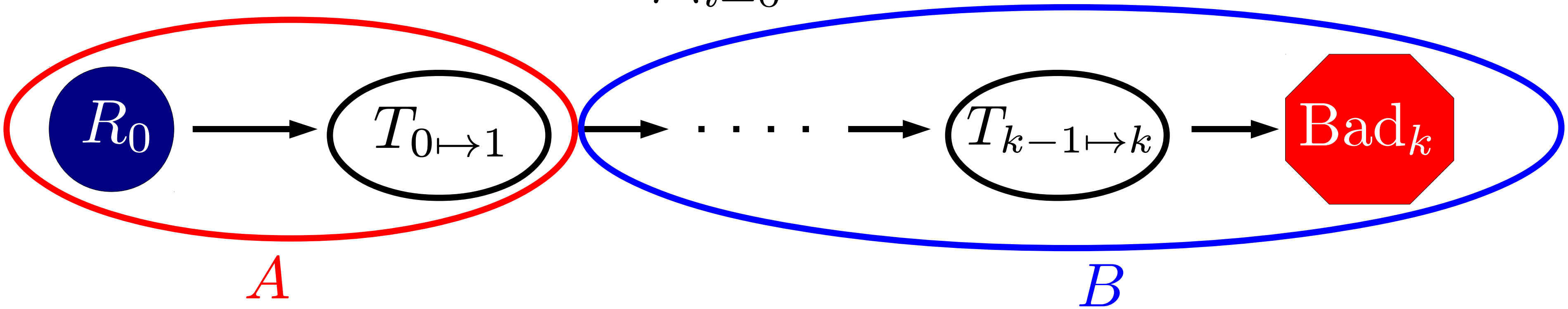


- If UNSAT:
  - Set $\varphi(X) := \mathrm{Interpolant}(A, B)[X'/X]$

φ is an abstraction of the forward image guided by the property

Interpolation-based reachability

- Set $R(X) := I(X)$
- Check satisfiability of $R_0 \wedge \bigwedge_{i=0}^{k-1} T_i \wedge \mathrm{Bad}_k$



- If UNSAT:
  - Set $\varphi(X) := \mathrm{Interpolant}(A, B)[X'/X]$

    φ is an abstraction of the forward image guided by the property

  - If $\varphi \models R$, return UNREACHABLE   fixpoint found
  - else, set $R(X) := R(X) \vee \varphi(X)$ and continue