Optimization Modulo Theories
An Introduction

Roberto Sebastiani

Dept. of Computer Science and Engineering, DISI
University of Trento, Italy
roberto.sebastiani@unitn.it
http://disi.unitn.it/rseba

– International SAT/SMT/AR School, Lisbon, PT, July 3-7th, 2019 –

Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Imptoving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)

Outline

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

SMT($\mathcal{T}$): the problem of deciding the satisfiability of a (typically) ground first-order formula wrt some background theory $\mathcal{T}$.

- $\mathcal{T}$ can be a combination of theories $\bigcup_i \mathcal{T}_i$
- Theories of Interest:
  - Linear arithmetic over the rationals ($\mathcal{LRA}$)
    $(T_\delta \rightarrow (s_1 = s_0 + 3.4 \cdot t - 3.4 \cdot t_0)) \wedge (\neg T_\delta \rightarrow (s_1 = s_0))$
  - Linear arithmetic over the integers ($\mathcal{LIA}$)
    $(x := x_l + 2^{16} x_h) \wedge (x \geq 0) \wedge (x \leq 2^{16} - 1)$
  - Arrays ($\mathcal{AR}$)
    $(i = j) \vee read(write(a, i, e), j) = read(a, j)$
  - Bit vectors ($\mathcal{BV}$)
    $x_{[16]}[15:0] = (y_{[16]}[15:8] :: z_{[16]}[7:0]) << w_{[8]}[3:0]$
  - Non-linear arithmetic ($\mathcal{NLA}$)
    $((c = a \cdot b) \wedge (a_1 = a - 1) \wedge (b_1 = a + 1)) \rightarrow (c = a_1 \cdot b_1 + 1)$
  - ...
- "Lazy" Approach: SMT solver = CDCL SAT solver + $\mathcal{T}$-solver(s)

Need for Satisfiability Modulo Theories (SMT)

SMT solvers widely used as backend engines in formal verification and many other applications

- SW verification
- verification of Timed and Hybrid Systems
- verification of RTL Circuit designs & of microcode
- static analysis of SW programs
- test-case generation
- program synthesis
- scheduling
- planning with resources
- compiler optimization
- ...

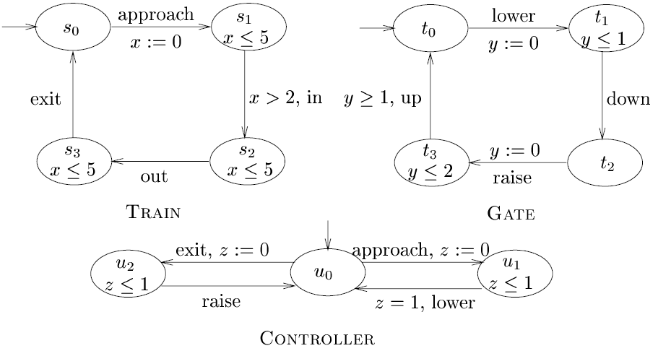
Need for Optimization Modulo Theories (SMT)

Many SMT-encodable problems require optimum solutions wrt. some objective function. E.g.:

- SW verification
- formal verification of parametric systems
- optimization of physical layout of circuit designs
- scheduling and temporal reasoning
- displacement of tools (e.g. strip-packing problem)
- planning with resources and retrofit planning
- radio link frequency assignment
- machine learning on hybrid domains
- goal modeling in requirement engineering
- ...

Ex.: FV of parametric systems

A (parametric version of a) timed system from [Alur, CAV-99] [8]:

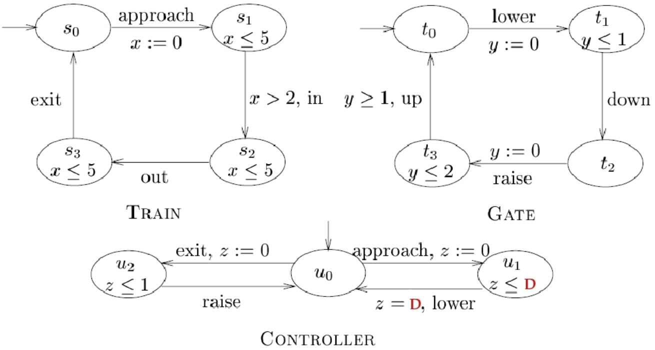


Decision Problem: check safety under fixed choices of the constants (e.g, the delay after which the controller orders the gate to lower the bar) ($M \models \mathbf{G}\neg(in \wedge up)$)

- BMC encodable into a SMT($\mathcal{LRA}$) problem (sat. $\Longrightarrow$ unsafe)

Ex.: FV of parametric systems

A (parametric version of a) timed system from [Alur, CAV-99] [8]:



Optimization Problem: find the minimum "unsafe" delay D after which the controller orders the gate to lower the bar, which doesn't guarantee safety ($M \not\models \mathbf{G}\neg(in \wedge up)$).

$\Longrightarrow$ Set the delay D strictly smaller

- BMC encodable into a OMT($\mathcal{LRA}$) problem (min. D s.t. satisf.)

Ex.: Formal Verification of Real-Time Systems

Model Checking: $M \models f$?

Bounded Model Checking (BMC) looks for an execution path of M of (increasing) length k

- satisfying the temporal property $\neg f$ (i.e. $M \models_k E\neg f$)
- minimizing the total elapsed time: $cost = \min(t^N - t^0)$

BMC is encoded into SMT($\mathcal{T}$) (e.g. $\mathcal{T} = \mathcal{LRA} \cup \mathcal{AR} \cup \ldots$):

- if φ_k is satisfiable, then $M \not\models f$

$$
\begin{array}{lccl}
 & DUMP^1 & \rightarrow & (A^1 = write(A^0, i^1, v_i^1)) \\
\wedge & \neg DUMP^1 & \rightarrow & (A^1 = A^0) \\
\wedge & DUMP^1 & \rightarrow & (t^1 - t^0 = 0) \\
\wedge & \ldots & & \\
\wedge & WAIT^1 & \rightarrow & (t^1 - t^0 > 0) \\
\wedge & \ldots & & \\
\wedge & DUMP^N & \rightarrow & \ldots \\
\wedge & \ldots & &
\end{array}
$$

Ex.: Formal Verification of Real-Time Systems

Model Checking: $M \models f$?

Bounded Model Checking (BMC) looks for an execution path of M of (increasing) length k

- satisfying the temporal property $\neg f$ (i.e. $M \models_k E\neg f$)
- minimizing the total elapsed time: $cost = \min(t^N - t^0)$

BMC is encoded into SMT($\mathcal{T}$) (e.g. $\mathcal{T} = \mathcal{LRA} \cup \mathcal{AR} \cup \ldots$):

- if φ_k is satisfiable, then $M \not\models f$

$$
\begin{array}{lllll}
 & DUMP^1 & \rightarrow & (A^1 = write(A^0, i^1, v_i^1)) \\
\wedge & \neg DUMP^1 & \rightarrow & (A^1 = A^0) \\
\wedge & DUMP^1 & \rightarrow & (t^1 - t^0 = 0) \\
\wedge & \ldots & & \\
\wedge & WAIT^1 & \rightarrow & (t^1 - t^0 > 0) \\
\wedge & \ldots & & \\
\wedge & DUMP^N & \rightarrow & \ldots \\
\wedge & \ldots & &
\end{array}
$$

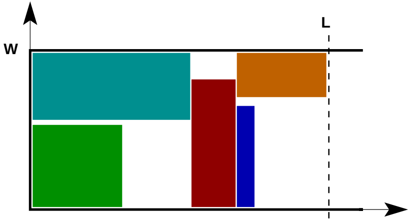
Ex.: Planning with Resources [62]

- SAT-based planning augmented with numerical constraints
- Straightforward to encode into into SMT($\mathcal{LRA}$)
- Goal: find a plan minimizing some resource consumption (time, money, gasoline, ...)

Example (sketch) [62]

$$
\begin{array}{ll}
(\mathit{Deliver}) & \wedge \;\texttt{// goal} \\
(\mathit{MaxLoad}) & \wedge \;\texttt{// load constraint} \\
(\mathit{MaxFuel}) & \wedge \;\texttt{// fuel constraint} \\
(\mathit{Move} \rightarrow \mathit{MinFuel}) & \wedge \;\texttt{// move requires fuel} \\
(\mathit{Move} \rightarrow \mathit{Deliver}) & \wedge \;\texttt{// move implies delivery} \\
(\mathit{GoodTrip} \rightarrow \mathit{Deliver}) & \wedge \;\texttt{// a good trip requires} \\
(\mathit{GoodTrip} \rightarrow \mathit{AllLoaded}) & \wedge \;\texttt{// a full delivery} \\
\hline
(\mathit{MaxLoad} \rightarrow (\mathit{load} \leq 30)) & \wedge \;\texttt{// load limit} \\
(\mathit{MaxFuel} \rightarrow (\mathit{fuel} \leq 15)) & \wedge \;\texttt{// fuel limit} \\
(\mathit{MinFuel} \rightarrow (\mathit{fuel} \geq 7 + 0.5\mathit{load})) & \wedge \;\texttt{// fuel constraint} \\
(\mathit{AllLoaded} \rightarrow (\mathit{load} = 45)) & \quad\;\texttt{//}
\end{array}
$$

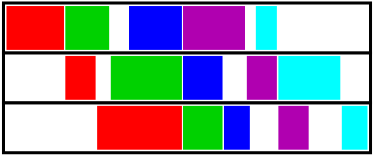
Ex.: (LGDP/MILP) Strip-packing & Carpet-cutting [29, 51, 53]



Strip-packing: Minimize the length L of a strip of width W while fitting N rectangles (no overlap, no rotation) [29]. Carpet-cutting: w. rotation.

$$
\begin{array}{rcl}
\varphi & \stackrel{\text{def}}{=} & (\text{cost} = L) \wedge \bigwedge_{i \in N}(L \geq x_i + L_i) \\
& \wedge & \bigwedge_{i,j \in N, i<j} \Big((x_i + L_i \leq x_j) \vee (x_j + L_j \leq x_i) \\
& & \vee (y_i - H_i \geq y_j) \vee (y_j - H_j \geq y_i) \Big) \\
& \wedge & \bigwedge_{i \in N}(x_i \leq \text{ub} - L_i) \wedge \bigwedge_{i \in N}(x_i \geq 0) \\
& \wedge & \bigwedge_{i \in N}(H_i \leq y_i) \wedge \bigwedge_{i \in N}(W \geq y_i) \wedge \bigwedge_{i \in N}(y_i \geq 0)
\end{array}
$$

Ex.: (LGDP/MILP) Zero-Wait Jobshop Scheduling [29, 51, 53]



Given a set I of jobs which must be scheduled sequentially on a set J of consecutive stages with zero-wait transfer between them, minimize the makespan M [47].

$$
\begin{aligned}
\varphi \quad \stackrel{\text{def}}{=} \quad & (\text{cost} = M) \wedge \bigwedge_{i \in I} (M \geq s_i + \textstyle\sum_{j \in J_i} t_{ij}) \wedge \bigwedge_{i \in I} (s_i \geq 0) \\
\wedge \quad & \bigwedge_{j \in C_{ik}, i,k \in I, i<k} \Big((s_i + \textstyle\sum_{m \in J_i, m \leq j} t_{im} \leq s_k + \sum_{m \in J_k, m<j} t_{km}) \\
& \qquad \vee (s_k + \textstyle\sum_{m \in J_k, m \leq j} t_{km} \leq s_i + \sum_{m \in J_i, m<j} t_{im}) \Big)
\end{aligned}
$$

Outline

Optimization Modulo Theories: General Case

Ingredients

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

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

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

- maximization dual

Optimization Modulo Theories with $\mathcal{LIRA}$ costs

Ingredients

- an SMT formula φ on $\mathcal{LIRA} \cup \mathcal{T}$
  - $\mathcal{LIRA}$ can be $\mathcal{LRA}$, $\mathcal{LIA}$ or a combination of both
  - $\mathcal{T} \stackrel{\text{def}}{=} \bigcup_i \mathcal{T}_i$, possibly empty
  - $\mathcal{LIRA}$ and $\mathcal{T}_i$ disjoint Nelson-Oppen theories
- a $\mathcal{LIRA}$ variable [term] "cost" occurring in φ
- (optionally) two constant numbers lb (lower bound) and ub (upper bound) s.t. lb $\leq$ *cost* $<$ ub (lb, ub may be $\mp\infty$)

Optimization Modulo Theories with $\mathcal{LIRA}$ costs (OMT($\mathcal{LIRA} \cup \mathcal{T}$))

Find a model for φ whose value of cost is minimum.

- maximization dual

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

Optimization Modulo Theories with $\mathcal{LRA}$ costs

Ingredients

- an SMT formula φ on $\mathcal{LRA} \cup \mathcal{T}$
  - $\mathcal{LIRA}$ can be $\mathcal{LRA}$, $\mathcal{LIA}$ or a combination of both
  - $\mathcal{T} \stackrel{\text{def}}{=} \bigcup_i \mathcal{T}_i$, possibly empty
  - $\mathcal{LRA}$ and $\mathcal{T}_i$ disjoint Nelson-Oppen theories
- a $\mathcal{LRA}$ variable [term] "cost" occurring in φ
- (optionally) two constant numbers lb (lower bound) and ub (upper bound) s.t. lb $\leq$ *cost* $<$ ub (lb, ub may be $\mp\infty$)

Optimization Modulo Theories with $\mathcal{LRA}$ costs (OMT($\mathcal{LRA} \cup \mathcal{T}$))

Find a model for φ whose value of cost is minimum.

- maximization dual

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

Solving OMT($\mathcal{LRA}$) [52, 53]

General idea

Combine standard SMT and LP minimization techniques.

Offline Schema

- Minimizer: based on the Simplex $\mathcal{LRA}$-solver by [25]
  - Handles strict inequalities
- Search Strategies:
  - Linear-Search strategy
  - Mixed Linear/Binary strategy

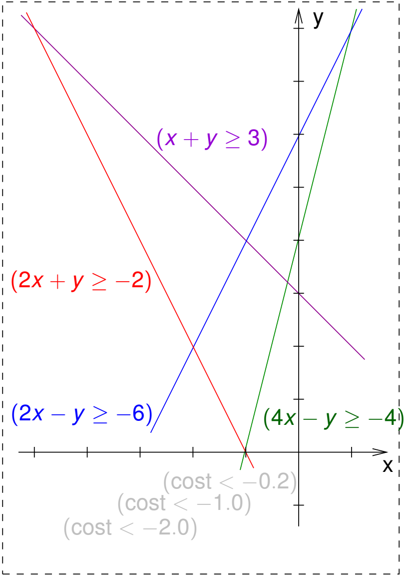
A toy example (linear search)

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

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

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

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



A toy example (linear search)

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

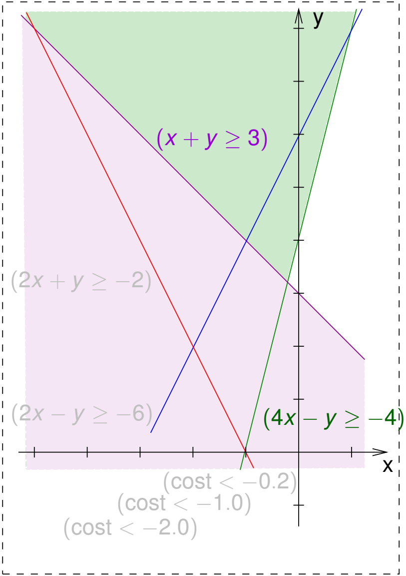
- OMT($\mathcal{LRA}$) problem:

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

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

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

$\Longrightarrow$ SAT, $min = -0.2$



A toy example (linear search)

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

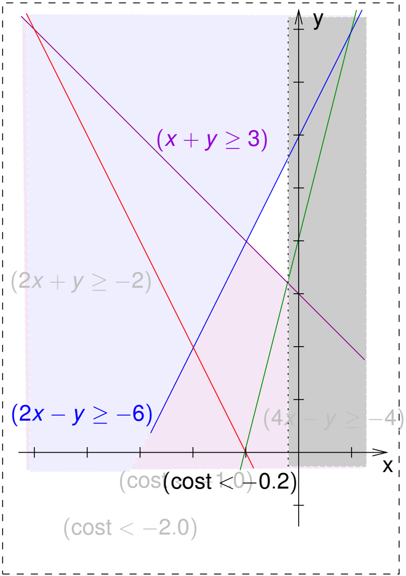
- OMT($\mathcal{LRA}$) problem:

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

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

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

$\Longrightarrow$ SAT, $min = -1.0$



A toy example (linear search)

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

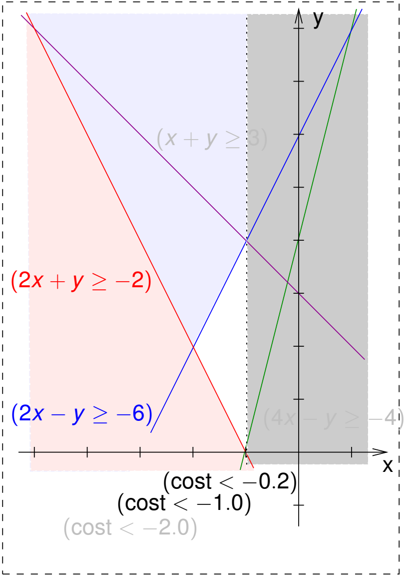
- OMT($\mathcal{LRA}$) problem:

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

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

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

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



A toy example (linear search)

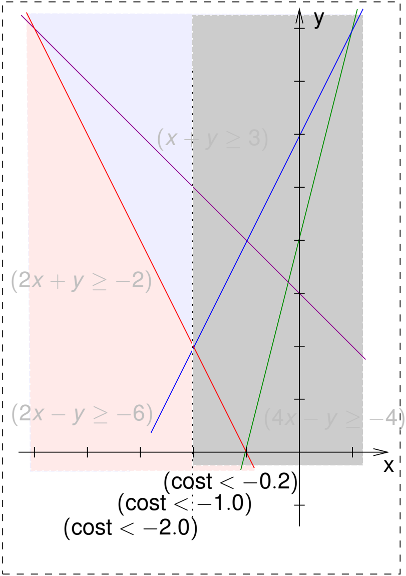
[w. pure-literal filt. $\Longrightarrow$ partial assignments]

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

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

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

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

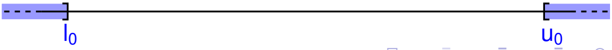


Offline Schema: Mixed Linear/Binary-Search Strategy

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

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

while $(\text{l} < \text{u})$ **do**



Offline Schema: Mixed Linear/Binary-Search Strategy

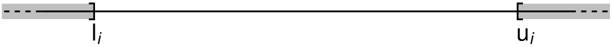
Input: $\langle \varphi, \mathsf{cost}, \mathsf{lb}, \mathsf{ub} \rangle$ // lb can be $-\infty$, ub can be $+\infty$

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

while $(\mathsf{l} < \mathsf{u})$ **do**

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

  else // Linear-search Mode



Offline Schema: Mixed Linear/Binary-Search Strategy

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

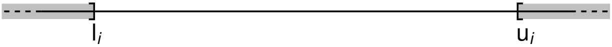
$\text{l} \leftarrow \text{lb}$; $\text{u} \leftarrow \text{ub}$; $\mathcal{M} \leftarrow \emptyset$; $\varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{lb}), (\text{cost} < \text{ub})\}$;

while $(\text{l} < \text{u})$ **do**

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

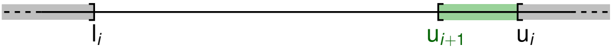
  else // Linear-search Mode

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



Offline Schema: Mixed Linear/Binary-Search Strategy

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



Offline Schema: Mixed Linear/Binary-Search Strategy

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

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

while $(\text{l} < \text{u})$ **do**

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

    else // Linear-search Mode

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

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

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

            $\text{l} \leftarrow \text{u}$;

return $\langle \mathcal{M}, \text{u} \rangle$

Offline Schema: Mixed Linear/Binary-Search Strategy

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

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

while $(\text{l} < \text{u})$ **do**

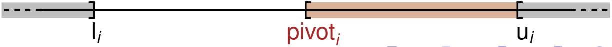
  if (BinSearchMode()) **then** // Binary-search Mode

    $\text{pivot} \leftarrow \text{ComputePivot}(\text{l}, \text{u})$;

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

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

  else // Linear-search Mode



Offline Schema: Mixed Linear/Binary-Search Strategy

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

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

while $(\text{l} < \text{u})$ **do**

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

    $\text{pivot} \leftarrow \text{ComputePivot}(\text{l}, \text{u})$;

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

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

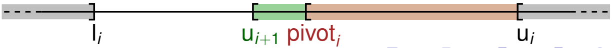
  else // Linear-search Mode

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

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

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

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



Offline Schema: Mixed Linear/Binary-Search Strategy

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

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

while $(\text{l} < \text{u})$ **do**

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

        $\text{pivot} \leftarrow \text{ComputePivot}(\text{l}, \text{u})$;

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

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

    else // Linear-search Mode

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

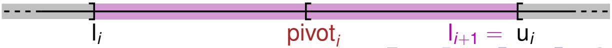
    else $\{\text{res} = \text{UNSAT}\}$

        if $((\text{cost} < \text{pivot}) \notin \text{SMT.ExtractUnsatCore}(\varphi))$ **then**

            $\text{l} \leftarrow \text{u}$;

        else

return $\langle \mathcal{M}, \text{u} \rangle$



Offline Schema: Mixed Linear/Binary-Search Strategy

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

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

while $(\text{l} < \text{u})$ **do**

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

    $\text{pivot} \leftarrow \text{ComputePivot}(\text{l}, \text{u});$

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

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

  else // Linear-search Mode

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

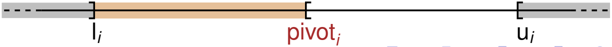
  else $\{\text{res} = \text{UNSAT}\}$

    if $((\text{cost} < \text{pivot}) \notin \text{SMT.ExtractUnsatCore}(\varphi))$ **then**

    else

      $\text{l} \leftarrow \text{pivot};$

      $\varphi \leftarrow (\varphi \setminus \{(\text{cost} < \text{pivot})\}) \cup \{\neg(\text{cost} < \text{pivot})\};$



The Minimizer

Minimizer embedded within the Simplex-based $\mathcal{LRA}$-solver by [25]

- Minimization by standard Simplex techniques

Strict Inequalities

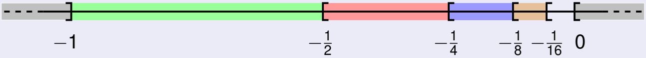
Temporally treated as non-strict inequalities:

- if minimum cost *min* lays only on non-strict inequalities, *min* is a solution
- otherwise, for some $\delta > 0$ there exists a solution for every cost $c \in]min, min + \delta]$

If *min* is a non-strict minimum, then $(cost \leq min)$ is added to φ.

Binary vs. Linear search

Beware of Zeno: pure binary search can cause infinite partitioning



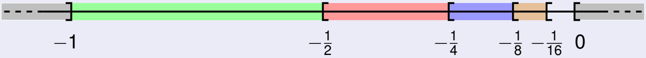
- E.g. if no solution in $[-1, 0[$, then $[-1, 0[, [-1/2, 0[, [-1/4, 0[, [-1/8, 0[, \ldots$
- SMT solver may find a conflict set $\eta \cup (\text{cost} < \text{pivot})$ even if $\varphi \setminus \{(\text{cost} < \text{pivot})\}$ is $\mathcal{LRA}$-inconsistent
- Solution: Binary-search interleaved with linear-search (Mixed Linear/Binary Search Strategy)

Note: Binary search not "obviously faster" than linear search

- Binary search: typically smaller number of range-restriction steps
- Linear search: average smaller cost of each range-restriction steps (unsatisfiable calls typically much harder than sat. ones)

Binary vs. Linear search

Beware of Zeno: pure binary search can cause infinite partitioning



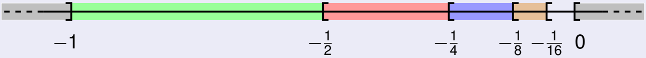
- E.g. if no solution in $[-1, 0[$, then $[-1, 0[, [-1/2, 0[, [-1/4, 0[, [-1/8, 0[, \ldots$
- SMT solver may find a conflict set $\eta \cup$ (cost $<$ pivot) even if $\varphi \setminus \{$(cost $<$ pivot)$\}$ is $\mathcal{LRA}$-inconsistent
- Solution: Binary-search interleaved with linear-search (Mixed Linear/Binary Search Strategy)

Note: Binary search not “obviously faster” than linear search

- Binary search: typically smaller number of range-restriction steps
- Linear search: average smaller cost of each range-restriction steps (unsatisfiable calls typically much harder than sat. ones)

Binary vs. Linear search

Beware of Zeno: pure binary search can cause infinite partitioning



- E.g. if no solution in $[-1, 0[$, then $[-1, 0[, [-1/2, 0[, [-1/4, 0[, [-1/8, 0[, \ldots$
- SMT solver may find a conflict set $\eta \cup (\text{cost} < \text{pivot})$ even if $\varphi \setminus \{(\text{cost} < \text{pivot})\}$ is $\mathcal{LRA}$-inconsistent
- Solution: Binary-search interleaved with linear-search (Mixed Linear/Binary Search Strategy)

Note: Binary search not "obviously faster" than linear search

- Binary search: typically smaller number of range-restriction steps
- Linear search: average smaller cost of each range-restriction steps (unsatisfiable calls typically much harder than sat. ones)

Termination & Correctness

Termination

The linear search procedure terminates:

- Finite number of satisfiable truth assignments μ_i
- No truth assignment μ_i generated twice
  - guaranteed by computing the minimum cost m_i of μ_i and learning $(\mathrm{cost} < \mathrm{m}_i)$

$\Longrightarrow$ also the mixed linear/binary search procedure terminates

Correctness

The procedure returns the minimum cost

- Explores the whole space of satisfiable truth assignments
- For every satisfiable truth assignment, Minimize finds the minimum cost

Some Enhancements [52, 53, 16]

- After invoking the minimizer and learning $(\text{cost} < \text{m}_i)$
  - Invoke $\mathcal{LRA}$-solver.solve$(\mu_i \wedge (\text{cost} < \text{m}_i)) \Rightarrow$ *conflict set* η_i and learn also $\neg\eta_i$
  - Binary mode: learn also $(\text{cost} < \text{pivot}_i)$ to reuse previously learned clauses in the form $\neg(\text{cost} < \text{pivot}_i) \vee C$
- Tightening of conflicts on binary search [52, 53, 16])
  - when $\varphi \wedge (\text{cost} < \mathit{pivot}_i)$ fails, look for tighter conflict $\neg(\text{cost} < M_i)$ s.t. $M_i > \mathit{pivot}_i$
- Adaptive Mixed Linear/Binary-Search Strategy: BinSearchMode() chooses according to $\frac{\Delta \text{ub}}{\Delta \# \mathit{conflicts}}$

From OMT($\mathcal{LRA}$) to OMT($\mathcal{LRA} \cup \mathcal{T}$)

OMT($\mathcal{LRA}$) procedure extended for handling $\mathcal{LRA} \cup \mathcal{T}$-formulas φ:

For free if SMT solver handles $\mathcal{LRA} \cup \mathcal{T}$-solving by *Delayed Theory Combination* [18] or Model-based Combination [23], splitting negated interface equalities $\neg(x_i = x_j)$ into $((x_i < x_j) \vee (x_i > x_j))$:

- Truth assignments $\mu' \stackrel{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{eid} \cup \mu_{\mathcal{T}}$ s.t. $\mu' \models \varphi$
  - μ_{eid} is a set containing interface equalities $(x_i = x_j)$, disequalities $\neg(x_i = x_j)$ and one inequality in $\{(x_i < x_j), (x_i > x_j)\}$ for every disequality in μ_{eid}
- $\mathcal{LRA}$-solver.solve invoked on $\mu'_{\mathcal{LRA}}$
  - $\mu'_{\mathcal{LRA}} \stackrel{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{ei}$ obtained from μ_{eid} by dropping disequalities

$\Longrightarrow$ $\mathcal{LRA}$-solver.minimize invoked on $\langle \text{cost}, \mu'_{\mathcal{LRA}} \rangle$

From OMT($\mathcal{LRA}$) to OMT($\mathcal{LRA} \cup \mathcal{T}$)

OMT($\mathcal{LRA}$) procedure extended for handling $\mathcal{LRA} \cup \mathcal{T}$-formulas φ:

For free if SMT solver handles $\mathcal{LRA} \cup \mathcal{T}$-solving by *Delayed Theory Combination* [18] or Model-based Combination [23], splitting negated interface equalities $\neg(x_i = x_j)$ into $((x_i < x_j) \vee (x_i > x_j))$:

- Truth assignments $\mu' \stackrel{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{eid} \cup \mu_{\mathcal{T}}$ s.t. $\mu' \models \varphi$
  - μ_{eid} is a set containing interface equalities $(x_i = x_j)$, disequalities $\neg(x_i = x_j)$ and one inequality in $\{(x_i < x_j), (x_i > x_j)\}$ for every disequality in μ_{eid}
- $\mathcal{LRA}$-solver.solve invoked on $\mu'_{\mathcal{LRA}}$
  - $\mu'_{\mathcal{LRA}} \stackrel{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{ei}$ obtained from μ_{eid} by dropping disequalities

$\Longrightarrow$ $\mathcal{LRA}$-solver.minimize invoked on $\langle \text{cost}, \mu'_{\mathcal{LRA}} \rangle$

From OMT($\mathcal{LRA} \cup \mathcal{T}$) to OMT($\mathcal{LIRA} \cup \mathcal{T}$) [55, 16]

- OMT($\mathcal{LRA} \cup \mathcal{T}$) procedures extended to $\mathcal{LIA}$ and mixed $\mathcal{LRA}$/$\mathcal{LIA}$ costs [16, 55]
- $\mathcal{LRA}$/$\mathcal{LIA}$-solvers enhanced with ILP minimization techniques (branch & bound, cutting planes, backjumping, ...)
- Note: with $\mathcal{LIA}$
  - ILP minimization often expensive
  - no "Zeno" problem for binary search
  - in principle, if problem is lower-bounded, the ILP minimizer is not necessary
- tradeoff between LP, (in)complete ILP minimization, binary search and Boolean Search [16, 55]

Truncated Branch and Bound

Observations:

- branch & bound can be expensive in degenerate cases
- optimality not truly necessary

Idea:

always stop B&B after first iteration, even if cost value is not guaranteed to be optimal.

Trade-off:

- less expensive minimization procedure on Integers
- risk of CDCL generating same μ multiple times

Outline

Incremental OMT [15, 55, 54]

Call OMT incrementally

- e.g., in BMC with parametric systems [53]

Intuition

In OMT, all learned clauses are either $\mathcal{T}$-lemmas, or derive from $\mathcal{T}$-lemmas and the original formulas , or are in the form $(\text{cost} < min)$
$\Longrightarrow$ exploit incrementality of SMT solvers, in two alternative ways:

(i) drop the $(\text{cost} < min)$ from one OMT call to the other

(ii) assert fresh variable S at each OMT call, and learn $\neg S \vee (\text{cost} < min)$ instead of $(\text{cost} < min)$

$\Longrightarrow$ can reuse learned clauses from OMT call to the other, (included these in the form $\neg(\text{cost} < min_{old}) \vee C$ as soon as $min_{cur} \leq min_{old}$.)

OMT with Independent Objectives (Boxed OMT) [38, 55]

The problem: $\langle \varphi, \{\mathsf{cost}_1, ..., \mathsf{cost}_k\} \rangle$ [38]

Given $\langle \varphi, \mathcal{C} \rangle$ s.t.:

- φ is the input formula
- $\mathcal{C} \stackrel{\text{def}}{=} \{\mathsf{cost}_1, ..., \mathsf{cost}_k\}$ is a set of $\mathcal{LIRA}$-terms on variables in φ,

$\langle \varphi, \mathcal{C} \rangle$ is the problem of finding a set of independent $\mathcal{LIRA}$-models $\mathcal{M}_1, ..., \mathcal{M}_k$ s.t. s.t. each $\mathcal{M}_i$ makes cost_i minimum.

Notes

- derives from SW verification problems [38]
- equivalent to k independent problems $\langle \varphi, \mathsf{cost}_1 \rangle, ..., \langle \varphi, \mathsf{cost}_k \rangle$
- intuition: share search effort for the different objectives
- generalizes to OMT($\mathcal{LIRA} \cup \mathcal{T}$) straightforwardly

OMT with Multiple Objectives [38, 16, 55]

Solution

- Intuition: when a $\mathcal{T}$-consistent satisfying assignment μ is found,

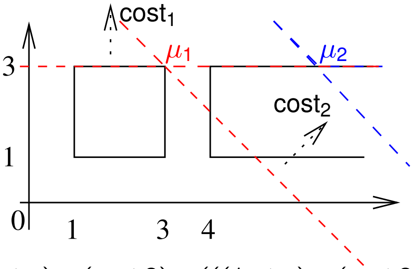
  $\texttt{foreach}\ \textbf{cost}_{\texttt{i}}$

  $\quad \texttt{min}_{\texttt{i}} := \texttt{min}\{\texttt{min}_{\texttt{i}}, \mathcal{T}\texttt{solver.minimize}(\mu, \textbf{cost}_{\texttt{i}})\};$

  $\texttt{learn}\ \bigvee_{\texttt{i}}(\textbf{cost}_{\texttt{i}} < \texttt{min}_{\texttt{i}}); \qquad // \ (\textbf{cost}_{\texttt{i}} < -\infty) \equiv \bot$

  proceed until UNSAT;
- Notice:
  - for each μ, guaranteed improvement of at least one min_i
  - in practice, for each μ, multiple cost_i minima are improved
- Implemented improvements:
  - (a) drop previous clauses $\bigvee_i(\mathsf{cost}_i < min_i)$
  - (b) $(\mathsf{cost}_i < min_i)$ pushed in μ first: if $\mathcal{T}$-inconsistent, skip minimization
  - (c) learn $\neg(\mathsf{cost}_i < min_i) \vee (\mathsf{cost}_i < min_i^{old})$, s.t. min_i^{old} previous min_i
    $\Longrightarrow$ reuse previously-learned clauses like $\neg(\mathsf{cost}_i < min_i^{old}) \vee C$

Boxed OMT: Example [38, 55]



$$\varphi \quad = \quad (1 \leq y) \wedge (y \leq 3) \wedge (((1 \leq x) \wedge (x \leq 3)) \vee (x \geq 4))$$
$$\wedge \quad (\text{cost}_1 = -y) \wedge (\text{cost}_2 = -x - y)$$

$$\mu_1 \quad = \quad \{(1 \leq y), (y \leq 3), (1 \leq x), (x \leq 3)\} \Longrightarrow \text{SAT} \Longrightarrow [-3, -6]$$

$\Longrightarrow$ *learn* $\{(\text{cost}_1 < -3) \vee (\text{cost}_2 < -6)\}$

$$\mu_2 \quad = \quad \{(1 \leq y), (y \leq 3), (x \geq 4)\} \Longrightarrow \text{SAT} \Longrightarrow [-3, -\infty]$$

$\Longrightarrow$ *learn* $\{(\text{cost}_1 < -3)\}$

$\Longrightarrow$ UNSAT

OMT with Lexicographic Combination of Objectives [16]

The problem

Find one optimal model $\mathcal{M}$ minimizing *costs* $\stackrel{\text{def}}{=}$ $\text{cost}_1, \text{cost}_2, ..., \text{cost}_k$ lexicographically.

Solution

- Intuition:

  $\{$*minimize* $\text{cost}_1\}$
  when UNSAT
  $\{$*substitute unit clause* $(\text{cost}_1 < min_1)$ *with* $(\text{cost}_1 = min_1)\}$
  $\{$*minimize* $\text{cost}_2\}$
  ...

OMT with Other forms of Objective Combination

OMT with Min-Max [Max-Min] optimization

Given $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_k\}\rangle$, find a solution which minimizes the maximum value among $\{\text{cost}_1, ..., \text{cost}_k\}$. (Max-Min dual.)

- Frequent in some applications (e.g. [53, 59])

$\Longrightarrow$ encode into OMT($\mathcal{LIRA} \cup \mathcal{T}$) problem
$\{\varphi \wedge \bigwedge_i (\text{cost}_i \leq \text{cost}), \text{cost}\}$ s.t. cost fresh.

OMT with linear combinations of costs

Given $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_k\}\rangle$ and a set of weights $\{w_1, ..., w_k\}$, find a solution which minimizes $\sum_i w_i \cdot \text{cost}_i$.

$\Longrightarrow$ encode into OMT($\mathcal{LIRA} \cup \mathcal{T}$) problem
$\{\varphi \wedge (\text{cost} = \sum_i w_i \cdot \text{cost}_i), \text{cost}\}$ s.t. cost fresh.

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

Outline

OMT($\mathcal{LRA} \cup \mathcal{T}$) vs. SMT with PB costs (& MaxSMT)

SMT + PB costs (& MaxSMT) can be encoded into OMT($\mathcal{LRA} \cup \mathcal{T}$):

$$\begin{array}{ll} \text{minimize} & \sum_j w_j \cdot A_j \quad //(\sum_j ite(A_j, w_j, 0)) \\ \text{s.t.} & \varphi \\ \Downarrow & \\ \text{minimize} & \sum_j x_j \\ \text{s.t.} & \varphi \wedge \bigwedge_j (A_j \rightarrow (x_j = w_j)) \wedge (\neg A_j \rightarrow (x_j = 0)) \\ & \quad \wedge \bigwedge_j ((x_j \geq 0) \wedge (x_j \leq w_j)) \end{array}$$

but not vice versa!

- SMT + PB costs finds the minimum-cost $\mathcal{T}$-satisfiable assignment
  $\Longrightarrow$ search for minimum is purely Boolean
- OMT($\mathcal{LIRA} \cup \mathcal{T}$) finds the $\mathcal{T}$-satisfiable assignment whose minimum cost is minimum
  $\Longrightarrow$ search for minimum involves two dimensions: Boolean and arithmetical

OMT($\mathcal{LRA} \cup \mathcal{T}$) vs. SMT with PB costs (& MaxSMT)

SMT + PB costs (& MaxSMT) can be encoded into OMT($\mathcal{LRA} \cup \mathcal{T}$):

$$\begin{array}{ll} \text{minimize} & \sum_j w_j \cdot A_j \quad //(\sum_j ite(A_j, w_j, 0)) \\ \text{s.t.} & \varphi \end{array}$$

$$\Downarrow$$

$$\begin{array}{ll} \text{minimize} & \sum_j x_j \\ \text{s.t.} & \varphi \wedge \bigwedge_j (A_j \rightarrow (x_j = w_j)) \wedge (\neg A_j \rightarrow (x_j = 0)) \\ & \quad \wedge \bigwedge_j ((x_j \geq 0) \wedge (x_j \leq w_j)) \end{array}$$

but not vice versa!

- SMT + PB costs finds the minimum-cost $\mathcal{T}$-satisfiable assignment
  $\Longrightarrow$ search for minimum is purely Boolean
- OMT($\mathcal{LIRA} \cup \mathcal{T}$) finds the $\mathcal{T}$-satisfiable assignment whose minimum cost is minimum
  $\Longrightarrow$ search for minimum involves two dimensions: Boolean and arithmetical

Remark: range constraints "$(x_j \geq 0) \wedge (x_j \leq w_j)$"

$$\begin{array}{ll} OMT+PB: & \sum_j w_j \cdot A_j,\ w_i > 0\ \ //(\sum_j ite(A_j, w_j, 0)) \\ & \Downarrow \\ & \sum_j x_j,\ x_j\ fresh \\ \text{s.t.} & \ldots \wedge \bigwedge_j (A_j \to (x_j = w_j)) \wedge (\neg A_j \to (x_j = 0)) \\ & \wedge (x_j \geq 0) \wedge (x_j \leq w_j) \end{array}$$

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

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

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]

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

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

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

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

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]

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

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

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

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

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]

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

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

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

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

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]

SMT/OMT with Pseudo-Boolean Costraints & Costs:

Alternative Solution: conversion into SMT($\mathcal{T}$)

- SAT + PB can be efficiently encoded into SAT [26]

$\Longrightarrow$ encode SMT($\mathcal{T}$) + PB into SMT($\mathcal{T}$)

- similar idea implemented in [16, 15] for cardinality constraints

Alternative Solution: Leverage SAT+PB

- develop a “modulo theory” version of your favourite PB-solver
- afaik, no implementation available

Alternative Solution: SMT($\mathcal{T} \cup \mathcal{C}$) [20]

- $\mathcal{C}$ is an ad-hoc “theory of costs”
- a specialized very-fast theory-solver for $\mathcal{C}$ added
  - very fast & aggressive search pruning and theory-propagation

SMT/OMT with Pseudo-Boolean Costraints & Costs:

Alternative Solution: conversion into SMT($\mathcal{T}$)

- SAT + PB can be efficiently encoded into SAT [26]

$\Longrightarrow$ encode SMT($\mathcal{T}$) + PB into SMT($\mathcal{T}$)

- similar idea implemented in [16, 15] for cardinality constraints

Alternative Solution: Leverage SAT+PB

- develop a “modulo theory” version of your favourite PB-solver
- afaik, no implementation available

Alternative Solution: SMT($\mathcal{T} \cup \mathcal{C}$) [20]

- $\mathcal{C}$ is an ad-hoc “theory of costs”
- a specialized very-fast theory-solver for $\mathcal{C}$ added
  - very fast & aggressive search pruning and theory-propagation

SMT/OMT with Pseudo-Boolean Costraints & Costs:

Alternative Solution: conversion into SMT($\mathcal{T}$)

- SAT + PB can be efficiently encoded into SAT [26]

$\Longrightarrow$ encode SMT($\mathcal{T}$) + PB into SMT($\mathcal{T}$)

- similar idea implemented in [16, 15] for cardinality constraints

Alternative Solution: Leverage SAT+PB

- develop a "modulo theory" version of your favourite PB-solver
- afaik, no implementation available

Alternative Solution: SMT($\mathcal{T} \cup \mathcal{C}$) [20]

- $\mathcal{C}$ is an ad-hoc "theory of costs"
- a specialized very-fast theory-solver for $\mathcal{C}$ added
    - very fast & aggressive search pruning and theory-propagation

A "Theory of cost" $\mathcal{C}$

A "theory of costs" $\mathcal{C}$

- M variables $cost^i$
- predicate "bound cost" $BC(cost^i, k)$ ("$cost^i \leq k$")
- predicate "incur cost" $IC(cost^i, j, \mathrm{c}^i_j)$ ("the jth addend of $cost^i$ is c^i_j")
- "$cost^i = \sum_{j=1}^{N^i} \mathrm{c}^i_j \cdot A^i_j, \quad s.t.\ cost^i \ \in \ (l^i, u^i]$"
  encoded as:
  $\neg BC(cost^i, l^i) \wedge BC(cost^i, u^i) \wedge \bigwedge_{j=1}^{N^i} (A^i_j \leftrightarrow IC(cost^i, j, \mathrm{c}^i_j))$

$\mathcal{C}$-solver

for each i, $\mathcal{C}$-solver mantains the current values of the incurred costs $cost^i \stackrel{\text{def}}{=} \sum_{IC(cost^i,j,c^i_j) \leftarrow \top} c^i_j$, the total cost of all unassigned IC's $\Delta cost^i \stackrel{\text{def}}{=} \sum_{\{IC(cost^i,j,c^i_j)\ unassigned\}} c^i_j$, and of the range $]lb_i, ub^i]$

1. $BC(cost^i, c) \leftarrow \top/\bot \Longrightarrow$ update $]lb_i, ub^i]$
2. $IC(cost^i, j, c^i_j) \leftarrow \top \Longrightarrow cost^i \leftarrow cost^i + c^i_j$
   $IC(cost^i, j, c^i_j) \leftarrow \bot \Longrightarrow \Delta cost^i \leftarrow \Delta cost^i - c^i_j$
3. $cost^i > ub^i \Longrightarrow$ conflict
4. $cost^i + \Delta cost^i \leq lb^i \Longrightarrow$ conflict
5. $IC(cost^i, j, c^i_j) \leftarrow \top$ causes 3. $\Longrightarrow$ propagate $\neg IC(cost^i, j, c^i_j)$
6. $IC(cost^i, j, c^i_j) \leftarrow \bot$ causes 4. $\Longrightarrow$ propagate $IC(cost^i, j, c^i_j)$

- very fast:
  - add one constraint & solve: 1 sum + 1 comparison
  - theory propagation: linear in the number of propagated literals

$\mathcal{C}$-solver

for each i, $\mathcal{C}$-solver mantains the current values of the incurred costs $cost^i \stackrel{\text{def}}{=} \sum_{IC(cost^i,j,c^i_j)\leftarrow\top} c^i_j$, the total cost of all unassigned IC's $\Delta cost^i \stackrel{\text{def}}{=} \sum_{\{IC(cost^i,j,c^i_j)\ unassigned\}} c^i_j$, and of the range $]lb_i, ub^i]$

1. $BC(cost^i, c) \leftarrow \top/\bot \Longrightarrow$ update $]lb_i, ub^i]$
2. $IC(cost^i, j, c^i_j) \leftarrow \top \Longrightarrow cost^i \leftarrow cost^i + c^i_j$
   $IC(cost^i, j, c^i_j) \leftarrow \bot \Longrightarrow \Delta cost^i \leftarrow \Delta cost^i - c^i_j$
3. $cost^i > ub^i \Longrightarrow$ conflict
4. $cost^i + \Delta cost^i \leq lb^i \Longrightarrow$ conflict
5. $IC(cost^i, j, c^i_j) \leftarrow \top$ causes 3. $\Longrightarrow$ propagate $\neg IC(cost^i, j, c^i_j)$
6. $IC(cost^i, j, c^i_j) \leftarrow \bot$ causes 4. $\Longrightarrow$ propagate $IC(cost^i, j, c^i_j)$

- very fast:
  - add one constraint & solve: 1 sum + 1 comparison
  - theory propagation: linear in the number of propagated literals

MaxSAT Modulo Theories (MaxSMT) I

[Partial Weighted] MaxSMT: The problem

Input: $\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$: resp. sets of hard and (weighted) soft $\mathcal{T}$-clauses;

Output: a maximum-weight set of soft $\mathcal{T}$-clauses $\psi_s^{\mathcal{T}}$ s.t. $\psi_s^{\mathcal{T}} \subseteq \varphi_s^{\mathcal{T}}$ and $\varphi_h^{\mathcal{T}} \cup \psi_s^{\mathcal{T}}$ is $\mathcal{T}$-satisfiable

MaxSMT vs. SMT with PB cost functions

MaxSMT $\langle \varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}} \rangle$ encodable into SMT with PB costs $\langle \varphi^{\mathcal{T}\prime}, \text{cost} \rangle$:

$$\varphi^{\mathcal{T}\prime} \stackrel{\text{def}}{=} \varphi_h^{\mathcal{T}} \cup \bigcup_{C_j^{\mathcal{T}} \in \varphi_s^{\mathcal{T}}} \{(A_j \vee C_j^{\mathcal{T}})\}; \quad \text{cost} \stackrel{\text{def}}{=} \sum_{C_j^{\mathcal{T}} \in \varphi_s^{\mathcal{T}}} w_j \cdot A_j,$$

SMT with PB costs $\langle \varphi^{\mathcal{T}\prime}, \text{cost} \stackrel{\text{def}}{=} \sum_j w_j \cdot A_j \rangle$ encodable into MaxSMT:

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \varphi^{\mathcal{T}\prime}; \quad \varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \bigcup_j \{\underbrace{(\neg A_j)}_{w_j}\};$$

MaxSAT Modulo Theories (MaxSMT) I

[Partial Weighted] MaxSMT: The problem

Input: $\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$: resp. sets of hard and (weighted) soft $\mathcal{T}$-clauses;

Output: a maximum-weight set of soft $\mathcal{T}$-clauses $\psi_s^{\mathcal{T}}$ s.t. $\psi_s^{\mathcal{T}} \subseteq \varphi_s^{\mathcal{T}}$ and $\varphi_h^{\mathcal{T}} \cup \psi_s^{\mathcal{T}}$ is $\mathcal{T}$-satisfiable

MaxSMT vs. SMT with PB cost functions

MaxSMT $\langle \varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}} \rangle$ encodable into SMT with PB costs $\langle \varphi^{\mathcal{T}\prime}, \text{cost} \rangle$:

$$\varphi^{\mathcal{T}\prime} \stackrel{\text{def}}{=} \varphi_h^{\mathcal{T}} \cup \bigcup_{C_j^{\mathcal{T}} \in \varphi_s^{\mathcal{T}}} \{(A_j \vee C_j^{\mathcal{T}})\}; \quad \text{cost} \stackrel{\text{def}}{=} \sum_{C_j^{\mathcal{T}} \in \varphi_s^{\mathcal{T}}} w_j \cdot A_j,$$

SMT with PB costs $\langle \varphi^{\mathcal{T}\prime}, \text{cost} \stackrel{\text{def}}{=} \sum_j w_j \cdot A_j \rangle$ encodable into MaxSMT:

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \varphi^{\mathcal{T}\prime}; \quad \varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \bigcup_j \{\underbrace{(\neg A_j)}_{w_j}\};$$

MaxSAT Modulo Theories (MaxSMT) II

Solution: encode into OMT($\mathcal{LRA}$) [44, 52, 53]

- can be composed with other objective functions

Alternative Solution: Leverage MaxSAT

- develop a “modulo theory” version of your favourite MaxSAT solver
- a few implementations available [4, 5, 15]

A “Modular” Approach to MaxSMT [21]

- Idea: Combine an SMT and a MaxSAT solver:
  MaxSMT = MaxSAT + SMT

MaxSAT Modulo Theories (MaxSMT) II

Solution: encode into OMT($\mathcal{LRA}$) [44, 52, 53]

- can be composed with other objective functions

Alternative Solution: Leverage MaxSAT

- develop a “modulo theory” version of your favourite MaxSAT solver
- a few implementations available [4, 5, 15]

A “Modular” Approach to MaxSMT [21]

- Idea: Combine an SMT and a MaxSAT solver:
  MaxSMT = MaxSAT + SMT

MaxSAT Modulo Theories (MaxSMT) II

Solution: encode into OMT($\mathcal{LRA}$) [44, 52, 53]

- can be composed with other objective functions

Alternative Solution: Leverage MaxSAT

- develop a “modulo theory” version of your favourite MaxSAT solver
- a few implementations available [4, 5, 15]

A “Modular” Approach to MaxSMT [21]

- Idea: Combine an SMT and a MaxSAT solver:
  MaxSMT = MaxSAT + SMT

A Modular Approach for MaxSMT$(\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}})$ [21]

Input: $\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$ // sets of hard and (weighted) soft $\mathcal{T}$-clauses
$\langle \varphi_h^{\mathcal{B}}, \varphi_s^{\mathcal{B}} \rangle \leftarrow \mathcal{T}2\mathcal{B}\ (\langle \varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}} \rangle)$;
$\Theta^{\mathcal{T}} \leftarrow \emptyset$; // current set of $\mathcal{T}$-lemmas
$\psi_s^{\mathcal{T}} \leftarrow \varphi_s^{\mathcal{T}}$; // current approximation of the result
while (SMT.Solve$(\varphi_h^{\mathcal{T}} \cup \psi_s^{\mathcal{T}} \cup \Theta^{\mathcal{T}})$ = UNSAT) **do**
  $\Theta^{\mathcal{T}} \leftarrow \Theta^{\mathcal{T}} \cup$ SMT.GetTLemmas(); $\Theta^{\mathcal{B}} \leftarrow \mathcal{T}2\mathcal{B}\ (\Theta^{\mathcal{T}})$;
  $\psi_s^{\mathcal{B}} \leftarrow$ MaxSAT$(\varphi_h^{\mathcal{B}} \cup \Theta^{\mathcal{B}}, \varphi_s^{\mathcal{B}})$; $\psi_s^{\mathcal{T}} \leftarrow \mathcal{B}2\mathcal{T}\ (\psi_s^{\mathcal{B}})$;
return $\psi_s^{\mathcal{T}}$;

Based on the cyclic interaction of an SMT and a MaxSAT solver:

- SMT.Solve used as a generator of sets of $\mathcal{T}$-lemmas $\Theta_0^{\mathcal{T}}, \Theta_1^{\mathcal{T}}, \dots$
  $\Longrightarrow$ provide the information to rule-out $\mathcal{T}$-inconsistent solutions
- MaxSAT used to extract minimum-cost clause sets $\psi_{s,0}^{\mathcal{B}}, \psi_{s,1}^{\mathcal{B}}, \dots$
  - works on Boolean abstractions $\varphi_h^{\mathcal{B}}, \varphi_s^{\mathcal{B}}$ plus the $\mathcal{T}$-lemmas $\Theta_i^{\mathcal{B}}$

A Modular Approach for MaxSMT($\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$) [21]

Input: $\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$ // sets of hard and (weighted) soft $\mathcal{T}$-clauses
$\langle \varphi_h^{\mathcal{B}}, \varphi_s^{\mathcal{B}} \rangle \leftarrow \mathcal{T}2\mathcal{B}\,(\langle \varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}} \rangle)$;
$\Theta^{\mathcal{T}} \leftarrow \emptyset$; // current set of $\mathcal{T}$-lemmas
$\psi_s^{\mathcal{T}} \leftarrow \varphi_s^{\mathcal{T}}$; // current approximation of the result
while (SMT.Solve($\varphi_h^{\mathcal{T}} \cup \psi_s^{\mathcal{T}} \cup \Theta^{\mathcal{T}}$) = UNSAT) **do**
    $\Theta^{\mathcal{T}} \leftarrow \Theta^{\mathcal{T}} \cup$ SMT.GetTLemmas(); $\Theta^{\mathcal{B}} \leftarrow \mathcal{T}2\mathcal{B}\,(\Theta^{\mathcal{T}})$;
    $\psi_s^{\mathcal{B}} \leftarrow$ MaxSAT($\varphi_h^{\mathcal{B}} \cup \Theta^{\mathcal{B}}, \varphi_s^{\mathcal{B}}$); $\psi_s^{\mathcal{T}} \leftarrow \mathcal{B}2\mathcal{T}\,(\psi_s^{\mathcal{B}})$;
return $\psi_s^{\mathcal{T}}$;

Based on the cyclic interaction of an SMT and a MaxSAT solver:

- SMT.Solve used as a generator of sets of $\mathcal{T}$-lemmas $\Theta_0^{\mathcal{T}}, \Theta_1^{\mathcal{T}}, \ldots$
  $\Longrightarrow$ provide the information to rule-out $\mathcal{T}$-inconsistent solutions
- MaxSAT used to extract minimum-cost clause sets $\psi_{s,0}^{\mathcal{B}}, \psi_{s,1}^{\mathcal{B}}, \ldots$
  - works on Boolean abstractions $\varphi_h^{\mathcal{B}}, \varphi_s^{\mathcal{B}}$ plus the $\mathcal{T}$-lemmas $\Theta_i^{\mathcal{B}}$

A Modular Approach for MaxSMT($\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$) [21]

Input: $\varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}}$ // sets of hard and (weighted) soft $\mathcal{T}$-clauses
$\langle \varphi_h^{\mathcal{B}}, \varphi_s^{\mathcal{B}} \rangle \leftarrow \mathcal{T}2\mathcal{B}\ (\langle \varphi_h^{\mathcal{T}}, \varphi_s^{\mathcal{T}} \rangle)$;
$\Theta^{\mathcal{T}} \leftarrow \emptyset$; // current set of $\mathcal{T}$-lemmas
$\psi_s^{\mathcal{T}} \leftarrow \varphi_s^{\mathcal{T}}$; // current approximation of the result
while (SMT.Solve($\varphi_h^{\mathcal{T}} \cup \psi_s^{\mathcal{T}} \cup \Theta^{\mathcal{T}}$) = UNSAT) **do**
  $\Theta^{\mathcal{T}} \leftarrow \Theta^{\mathcal{T}} \cup$ SMT.GetTLemmas(); $\Theta^{\mathcal{B}} \leftarrow \mathcal{T}2\mathcal{B}\ (\Theta^{\mathcal{T}})$;
  $\psi_s^{\mathcal{B}} \leftarrow$ MaxSAT($\varphi_h^{\mathcal{B}} \cup \Theta^{\mathcal{B}}, \varphi_s^{\mathcal{B}}$); $\psi_s^{\mathcal{T}} \leftarrow \mathcal{B}2\mathcal{T}\ (\psi_s^{\mathcal{B}})$;
return $\psi_s^{\mathcal{T}}$;

Based on the cyclic interaction of an SMT and a MaxSAT solver:

- SMT.Solve used as a generator of sets of $\mathcal{T}$-lemmas $\Theta_0^{\mathcal{T}}, \Theta_1^{\mathcal{T}}, \ldots$
  $\Longrightarrow$ provide the information to rule-out $\mathcal{T}$-inconsistent solutions
- MaxSAT used to extract minimum-cost clause sets $\psi_{s,0}^{\mathcal{B}}, \psi_{s,1}^{\mathcal{B}}, \ldots$
  - works on Boolean abstractions $\varphi_h^{\mathcal{B}}, \varphi_s^{\mathcal{B}}$ plus the $\mathcal{T}$-lemmas $\Theta_i^{\mathcal{B}}$

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \leq 0)) & [4] \\ C_1 : & ((x \leq 1)) & [3] \\ C_2 : & ((x \geq 2)) & [2] \\ C_3 : & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2 : & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3 : & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4 : & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5 : & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6 : & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \leq 0)) & [4] \\ C_1: & ((x \leq 1)) & [3] \\ C_2: & ((x \geq 2)) & [2] \\ C_3: & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2: & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3: & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4: & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5: & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6: & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ , \ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \le 0)) & [4] \\ C_1: & ((x \le 1)) & [3] \\ C_2: & ((x \ge 2)) & [2] \\ C_3: & ((x \ge 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \le 0) \vee (x \le 1)) \\ \theta_2: & (\neg(x \ge 3) \vee (x \ge 2)) \\ \theta_3: & (\neg(x \le 0) \vee \neg(x \ge 2)) \\ \theta_4: & (\neg(x \le 0) \vee \neg(x \ge 3)) \\ \theta_5: & (\neg(x \le 1) \vee \neg(x \ge 2)) \\ \theta_6: & (\neg(x \le 1) \vee \neg(x \ge 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2,\ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \le 0)) & [4] \\ C_1 : & ((x \le 1)) & [3] \\ C_2 : & ((x \ge 2)) & [2] \\ C_3 : & ((x \ge 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \le 0) \vee (x \le 1)) \\ \theta_2 : & (\neg(x \ge 3) \vee (x \ge 2)) \\ \theta_3 : & (\neg(x \le 0) \vee \neg(x \ge 2)) \\ \theta_4 : & (\neg(x \le 0) \vee \neg(x \ge 3)) \\ \theta_5 : & (\neg(x \le 1) \vee \neg(x \ge 2)) \\ \theta_6 : & (\neg(x \le 1) \vee \neg(x \ge 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ , \ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \le 0)) & [4] \\ C_1: & ((x \le 1)) & [3] \\ C_2: & ((x \ge 2)) & [2] \\ C_3: & ((x \ge 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \le 0) \vee (x \le 1)) \\ \theta_2: & (\neg(x \ge 3) \vee (x \ge 2)) \\ \theta_3: & (\neg(x \le 0) \vee \neg(x \ge 2)) \\ \theta_4: & (\neg(x \le 0) \vee \neg(x \ge 3)) \\ \theta_5: & (\neg(x \le 1) \vee \neg(x \ge 2)) \\ \theta_6: & (\neg(x \le 1) \vee \neg(x \ge 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ , \ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \leq 0)) & [4] \\ C_1 : & ((x \leq 1)) & [3] \\ C_2 : & ((x \geq 2)) & [2] \\ C_3 : & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2 : & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3 : & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4 : & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5 : & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6 : & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \le 0)) & [4] \\ C_1: & ((x \le 1)) & [3] \\ C_2: & ((x \ge 2)) & [2] \\ C_3: & ((x \ge 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \le 0) \vee (x \le 1)) \\ \theta_2: & (\neg(x \ge 3) \vee (x \ge 2)) \\ \theta_3: & (\neg(x \le 0) \vee \neg(x \ge 2)) \\ \theta_4: & (\neg(x \le 0) \vee \neg(x \ge 3)) \\ \theta_5: & (\neg(x \le 1) \vee \neg(x \ge 2)) \\ \theta_6: & (\neg(x \le 1) \vee \neg(x \ge 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2,\ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \leq 0)) & [4] \\ C_1: & ((x \leq 1)) & [3] \\ C_2: & ((x \geq 2)) & [2] \\ C_3: & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2: & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3: & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4: & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5: & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6: & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2,\ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \leq 0)) & [4] \\ C_1: & ((x \leq 1)) & [3] \\ C_2: & ((x \geq 2)) & [2] \\ C_3: & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2: & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3: & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4: & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5: & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6: & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\ \ , C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \ \ \}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\ \ , \ \ , C_2, C_3\}$	8	SAT

A toy example I

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad\qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \le 0)) & [4] \\ C_1: & ((x \le 1)) & [3] \\ C_2: & ((x \ge 2)) & [2] \\ C_3: & ((x \ge 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \le 0) \vee (x \le 1)) \\ \theta_2: & (\neg(x \ge 3) \vee (x \ge 2)) \\ \theta_3: & (\neg(x \le 0) \vee \neg(x \ge 2)) \\ \theta_4: & (\neg(x \le 0) \vee \neg(x \ge 3)) \\ \theta_5: & (\neg(x \le 1) \vee \neg(x \ge 2)) \\ \theta_6: & (\neg(x \le 1) \vee \neg(x \ge 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

An "unlucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_4\}$	$\{\quad, C_1, C_2, C_3\}$	11	UNSAT
2	$\{\theta_4, \theta_6\}$	$\{C_0, C_1, C_2, \quad\}$	9	UNSAT
3	$\{\theta_4, \theta_6, \theta_3\}$	$\{\quad, \quad, C_2, C_3\}$	8	SAT

A toy example II

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \leq 0)) & [4] \\ C_1 : & ((x \leq 1)) & [3] \\ C_2 : & ((x \geq 2)) & [2] \\ C_3 : & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2 : & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3 : & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4 : & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5 : & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6 : & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_1, \theta_2, \theta_5\}$	$\{\quad,\quad, C_2, C_3\}$	8	SAT

A toy example II

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \leq 0)) & [4] \\ C_1 : & ((x \leq 1)) & [3] \\ C_2 : & ((x \geq 2)) & [2] \\ C_3 : & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2 : & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3 : & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4 : & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5 : & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6 : & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_1, \theta_2, \theta_5\}$	$\{\quad , \quad , C_2, C_3\}$	8	SAT

A toy example II

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad\qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \leq 0)) & [4] \\ C_1 : & ((x \leq 1)) & [3] \\ C_2 : & ((x \geq 2)) & [2] \\ C_3 : & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2 : & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3 : & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4 : & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5 : & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6 : & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_1, \theta_2, \theta_5\}$	$\{\quad, \quad, C_2, C_3\}$	8	SAT

A toy example II

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0 : & ((x \leq 0)) & [4] \\ C_1 : & ((x \leq 1)) & [3] \\ C_2 : & ((x \geq 2)) & [2] \\ C_3 : & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1 : & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2 : & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3 : & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4 : & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5 : & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6 : & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_1, \theta_2, \theta_5\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example II

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \leq 0)) & [4] \\ C_1: & ((x \leq 1)) & [3] \\ C_2: & ((x \geq 2)) & [2] \\ C_3: & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2: & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3: & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4: & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5: & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6: & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight($\psi_{s,i}^{\mathcal{T}}$)	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_1, \theta_2, \theta_5\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

A toy example II

$$\varphi_h^{\mathcal{T}} \stackrel{\text{def}}{=} \emptyset \qquad\qquad \varphi_h^{\mathcal{B}} \stackrel{\text{def}}{=} \emptyset$$

$$\varphi_s^{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} C_0: & ((x \leq 0)) & [4] \\ C_1: & ((x \leq 1)) & [3] \\ C_2: & ((x \geq 2)) & [2] \\ C_3: & ((x \geq 3)) & [6] \end{array} \right\} \qquad \varphi_s^{\mathcal{B}} \stackrel{\text{def}}{=} \left\{ \begin{array}{ll} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi_h^{\mathcal{T}} \cup \varphi_s^{\mathcal{T}}$ is:

$$\Theta_*^{\mathcal{T}} = \left\{ \begin{array}{ll} \theta_1: & (\neg(x \leq 0) \vee (x \leq 1)) \\ \theta_2: & (\neg(x \geq 3) \vee (x \geq 2)) \\ \theta_3: & (\neg(x \leq 0) \vee \neg(x \geq 2)) \\ \theta_4: & (\neg(x \leq 0) \vee \neg(x \geq 3)) \\ \theta_5: & (\neg(x \leq 1) \vee \neg(x \geq 2)) \\ \theta_6: & (\neg(x \leq 1) \vee \neg(x \geq 3)) \end{array} \right\} \qquad \Theta_*^{\mathcal{B}} = \left\{ \begin{array}{l} (\neg A_0 \vee A_1) \\ (\neg A_3 \vee A_2) \\ (\neg A_0 \vee \neg A_2) \\ (\neg A_0 \vee \neg A_3) \\ (\neg A_1 \vee \neg A_2) \\ (\neg A_1 \vee \neg A_3) \end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

i	$\Theta_i^{\mathcal{T}}$	$\psi_{s,i}^{\mathcal{T}}$	Weight$(\psi_{s,i}^{\mathcal{T}})$	$SMT(\varphi_h^{\mathcal{T}} \cup \psi_{s,i}^{\mathcal{T}} \cup \Theta_i^{\mathcal{T}})$
0	$\{\}$	$\{C_0, C_1, C_2, C_3\}$	15	UNSAT
1	$\{\theta_1, \theta_2, \theta_5\}$	$\{\ \ ,\ \ , C_2, C_3\}$	8	SAT

Outline

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)				
OPTIMIZATION with PB cost function and constraints				
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)				SMT($\mathcal{T}$)
OPTIMIZATION with PB cost function and constraints				
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)				
OPTIMIZATION with PB cost function and constraints	(Weighted) MaxSAT PB Opt.			
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)				
OPTIMIZATION with PB cost function and constraints		MaxSMT and SMT($\mathcal{T}$) with PB cost funct.		
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)		LP		
OPTIMIZATION with PB cost function and constraints				
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)			ILP, MILP, DP, LGDP	
OPTIMIZATION with PB cost function and constraints				
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)				
OPTIMIZATION with PB cost function and constraints				
OPTIMIZATION with linear cost function				

OMT($\mathcal{LIRA} \cup \mathcal{T}$)

OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

	Boolean formulas	Sets of $\mathcal{LIRA}$ constraints	SMT($\mathcal{LIRA}$)	SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)
DECISION (Satisfiability)		LP	ILP, MILP, DP, LGDP	SMT($\mathcal{T}$)
OPTIMIZATION with PB cost function and constraints	(Weighted) MaxSAT PB Opt.	MaxSMT and SMT($\mathcal{T}$) with PB cost funct.		
OPTIMIZATION with linear cost function		OMT($\mathcal{LIRA} \cup \mathcal{T}$)		

(Finite Domain) Constraint Programming

back

FDCP/MILP

- Very efficient on (integer) linear arithmetic / combinatorial reasoning
- Very efficient handling of global constraints (e.g. all-different)
- Booleans typically represented as 0-1 integers
- (typically) finite precision arithmetic

SMT/OMT

- Very efficient on Boolean reasoning
- Supports other theories (*Array, Bit-Vectors, Strings, ...*)
- Incremental
- infinite precision arithmetic
- Other functionalities: all-smt, proofs, unsat-cores, interpolants, ...

Some OMT tools

- BCLT [44, 35]
  `http://www.cs.upc.edu/~oliveras/bclt-main.html`
- OPTIMATHSAT [52, 53, 55, 54, 57], on top of MATHSAT [22]
  `http://optimathsat.disi.unitn.it`
- SYMBA [38], on top of Z3 [24]
  `https://bitbucket.org/arieg/symba/src`
- Z3 [16, 15], on top of Z3 [24]
  `http://z3.codeplex.com`

More Recently:

- HAZEL [40]. $\Longrightarrow$ *$\mathcal{BV}$, incremental*
- CEGIO [7, 9] $\Longrightarrow$ *counterexample guided inductive optimization*
- MAXHS-MSAT [27] $\Longrightarrow$ *MaxSMT with* Implicit Hitting Set (IHS) algorithm
- PULI [33]. $\Longrightarrow$ *$\mathcal{LIA}$ cost functions, (based on* linear regression*)*

OMT Applications (OPTIMATHSAT)

Real-Time Systems. Worst-Case Execution Time (WCET) of programs [28]
$\Longrightarrow$ reproduced with OPTIMATHSAT [3]

Requirements Engineering. Constrained Goal Models with resources, preferences and goals [41, 42, 43].
$\Longrightarrow$ OPTIMATHSAT backend engine of CGM-TOOL [1]

Machine Learning. Inference & Learning in Hybrid domains [46, 60].
$\Longrightarrow$ OPTIMATHSAT backend engine of LMT tool [2]

Quantum Annealing. Solving SAT and MaxSAT with D-Wave 2000Q QAs [12, 13]
$\Longrightarrow$ offline used of OPTIMATHSAT to generate optimal QUBO encodings of Boolean functions

Formal Verification & Model Checking. Synthesis of Barrier Certificates for Hybrid Dynamical Systems [48]
$\Longrightarrow$ OPTIMATHSAT used as oracle to separate safe/unsafe regions starting from a simulation

Scheduling. Optimal sleep/wake-up scheduling for WSNs [32, 34, 33]
$\Longrightarrow$ OPTIMATHSAT used to deal with increasingly denser WSNs [34]

OMT Applications (Other tools)

Static Analysis.

- Generation of Invariants and Proving Termination via *Constraint-based* method [19]
- Finding Inductive Invariants via *Local Policy Iteration* [30, 31]

Formal Verification & Model Checking.

- Computing Loop Iterations for Bounded Program Verification [39]

Scheduling and Planning with Resources.

- Optimal plans for multi-robot systems [36, 37]
- Task planning for smart factories [14]
- Optimal Job-Shop Scheduling with OMT [50]
- Synthesis Communication Schedules for Time Sensitive Networks [45]

Software Security Engineering.

- Multi-Objective Workflow Satisfiability Problem [11]

Outline

Ongoing Work & Research Directions on OMT

Field still far from maturity, lots of possible research directions:

- Improve efficiency!
- OMT on different theories, e.g.:
  - Bit vectors ([16, 40])
  - $\mathcal{NLA}(\mathbb{R})$
  - $\mathcal{NLA}(\mathbb{Z})$ ([35])
  - Floating point ([61])
- Exploit alternative SMT schemas (e.g., Model-Construction SMT)
- Hybrid techniques, integration with techniques in neighbour fields (MaxSAT, PB, CSP, MILP, CA, ...)
- Extensive empirical comparison wrt. techniques in neighbour fields
  (MaxSAT, PB, CSP, MILP, ...)
- Bridge SMT/OMT with CSP/COP (Minizinc)

To this extent....

Announcement

PHD POSITION available in Trento on
"Advancing Optimization Modulo Theories"
The call will expire in a couple of months.

Please contact me if interested: roberto.sebastiani@unitn.it.
(Se also flier on the desk.)


"That's all Folks!"

References I

[1] CGM-Tool.
www.cgm-tool.eu.

[2] LMT.
http://disi.unitn.it/~teso/lmt/lmt.tgz.

[3] WCET OMT.
https://github.com/PatrickTrentin88/wcet_omt.

[4] Yices.
http://yices.csl.sri.com/.

[5] Z3.
http://research.microsoft.com/en-us/um/redmond/projects/z3/ml/z3.html.

[6] I. Abío, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell.
A Parametric Approach for Smaller and Better Encodings of Cardinality Constraints.
In *19th International Conference on Principles and Practice of Constraint Programming*, CP'13, 2013.

[7] H. F. Albuquerque, R. F. Araujo, I. V. de Bessa, L. C. Cordeiro, and E. B. de Lima Filho.
OptCE: A Counterexample-Guided Inductive Optimization Solver.
In *SBMF*, volume 10623 of *Lecture Notes in Computer Science*, pages 125–141. Springer, 2017.

[8] R. Alur.
Timed Automata.
In *Proc. CAV'99*, pages 8–22, 1999.

[9] R. F. Araujo, H. F. Albuquerque, I. V. de Bessa, L. C. Cordeiro, and J. E. C. Filho.
Counterexample guided inductive optimization based on satisfiability modulo theories.
Sci. Comput. Program., 165:3–23, 2018.

[10] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell.
Cardinality Networks: a theoretical and empirical study.
Constraints, 16(2):195–221, 2011.

References II

[11] C. Bertolissi, D. R. dos Santos, and S. Ranise.
Solving Multi-Objective Workflow Satisfiability Problems with Optimization Modulo Theories Techniques.
In *SACMAT*, pages 117–128. ACM, 2018.

[12] Z. Bian, F. Chudak, W. Macready, A. Roy, R. Sebastiani, and S. Varotti.
Solving SAT and MaxSAT with a Quantum Annealer: Foundations and a Preliminary Report.
In *Frontiers of Combining Systems*, volume 10483 of *LNCS*, pages 153–171. Springer, 2017.

[13] Z. Bian, F. A. Chudak, W. G. Macready, A. Roy, R. Sebastiani, and S. Varotti.
Solving SAT and maxsat with a quantum annealer: Foundations, encodings, and preliminary results.
CoRR, abs/1811.02524, 2018.
Under submission for journal publication.

[14] A. Bit-Monnot, F. Leofante, L. Pulina, E. Ábrahám, and A. Tacchella.
SMarTplan: a Task Planner for Smart Factories.
CoRR, abs/1806.07135, 2018.

[15] N. Bjørner, A. Phan, and L. Fleckenstein.
νz - an optimizing SMT solver.
In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, pages 194–199, 2015.

[16] N. Bjorner and A.-D. Phan.
νZ - Maximal Satisfaction with Z3.
In *Proc International Symposium on Symbolic Computation in Software Science*, Gammart, Tunisia, December 2014. EasyChair Proceedings in Computing (EPiC).
`http://www.easychair.org/publications/?page=862275542`.

[17] N. Bjorner, A.-D. Phan, and L. Fleckenstein.
Z3 - An Optimizing SMT Solver.
In *Proc. TACAS*, volume 9035 of *LNCS*. Springer, 2015.

References III

[18] M. Bozzano, R. Bruttomesso, A. Cimatti, T. A. Junttila, S. Ranise, P. van Rossum, and R. Sebastiani.
Efficient Theory Combination via Boolean Search.
Information and Computation, 204(10):1493–1525, 2006.

[19] L. Candeago, D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio.
Speeding up the Constraint-Based Method in Difference Logic.
In *SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 284–301. Springer, 2016.

[20] A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, and C. Stenico.
Satisfiability modulo the theory of costs: Foundations and applications.
In *TACAS*, volume 6015 of *LNCS*, pages 99–113. Springer, 2010.

[21] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani.
A Modular Approach to MaxSAT Modulo Theories.
In *International Conference on Theory and Applications of Satisfiability Testing, SAT*, volume 7962 of *LNCS*, July 2013.

[22] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani.
The MathSAT 5 SMT Solver.
In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'13.*, volume 7795 of *LNCS*, pages 95–109. Springer, 2013.

[23] L. M. de Moura and N. Bjørner.
Z3: An efficient smt solver.
In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[24] L. M. de Moura and N. Bjørner.
Z3: an efficient SMT solver.
In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, pages 337–340, 2008.

[25] B. Dutertre and L. de Moura.
A Fast Linear-Arithmetic Solver for DPLL(T).
In *CAV*, volume 4144 of *LNCS*, 2006.

References IV

[26] N. Eén and N. Sörensson.
Translating Pseudo-Boolean Constraints into SAT.
JSAT, 2(1-4):1–26, 2006.

[27] K. Fazekas, F. Bacchus, and A. Biere.
Implicit Hitting Set Algorithms for Maximum Satisfiability Modulo Theories.
In *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, pages 134–151. Springer, 2018.

[28] J. Henry, M. Asavoae, D. Monniaux, and C. Maïza.
How to Compute Worst-case Execution Time by Optimization Modulo Theory and a Clever Encoding of Program Semantics.
In *Proceedings of the 2014 SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems*, LCTES '14, pages 43–52, New York, NY, USA, 2014. ACM.

[29] M. Hifi.
Exact algorithms for the guillotine strip cutting/packing problem.
Computers & OR, 25(11):925–940, 1998.

[30] E. G. Karpenkov, K. Friedberger, and D. Beyer.
JavaSMT: A Unified Interface for SMT Solvers in Java.
In *VSTTE*, volume 9971 of *Lecture Notes in Computer Science*, pages 139–148, 2016.

[31] G. E. Karpenkov.
Finding inductive invariants using satisfiability modulo theories and convex optimization.
Theses, Université Grenoble Alpes, Mar. 2017.

[32] G. Kovásznai, C. Biró, and B. Erdélyi.
Generating Optimal Scheduling for Wireless Sensor Networks by Using Optimization Modulo Theories Solvers.
2017.

[33] G. Kovásznai, C. Biró, and B. Erdélyi.
Puli - a problem-specific omt solver.
EasyChair Preprint no. 371, EasyChair, 2018.

References V

[34] G. Kovásznai, B. Erdélyi, and C. Biró.
Investigations of graph properties in terms of wireless sensor network optimization.
In *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, pages 1–8, Jan 2018.

[35] D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio.
Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions.
In *SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2014.

[36] F. Leofante, E. Ábrahám, T. Niemueller, G. Lakemeyer, and A. Tacchella.
On the Synthesis of Guaranteed-Quality Plans for Robot Fleets in Logistics Scenarios via Optimization Modulo Theories.
In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 403–410, Aug 2017.

[37] F. Leofante, E. Abraham, T. Niemueller, G. Lakemeyer, and A. Tacchella.
Integrated Synthesis and Execution of Optimal Plans for Multi-Robot Systems in Logistics.
Information Systems Frontiers, pages 1–21, May 2018.

[38] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik.
Symbolic optimization with smt solvers.
In *POPL*, pages 607–618, 2014.

[39] T. Liu, S. S. Tyszberowicz, B. Beckert, and M. Taghdiri.
Computing Exact Loop Bounds for Bounded Program Verification.
In *SETTA*, volume 10606 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 2017.

[40] A. Nadel and V. Ryvchin.
Bit-Vector Optimization.
In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2016*, volume 9636 of *LNCS*. Springer, 2016.

[41] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos.
Multi-objective reasoning with constrained goal models.
Requirements Engineering, 2016.
In print. Published online 24 December 2016. DOI: `http://dx.doi.org/10.1007/s00766-016-0263-5`.

References VI

[42] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos.
Requirements Evolution and Evolution Requirements with Constrained Goal Models.
In *Proceedings of the 37nd International Conference on Conceptual Modeling - ER16*, LNCS. Springer, 2016.

[43] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos.
Modeling and Reasoning on Requirements Evolution with Constrained Goal Models.
In A. Cimatti and M. Sirjani, editors, *Software Engineering and Formal Methods - 15th International Conference, SEFM 2017, Trento, Italy, September 4-8, 2017, Proceedings*, volume 10469 of *Lecture Notes in Computer Science*, pages 70–86. Springer, 2017.

[44] R. Nieuwenhuis and A. Oliveras.
On SAT Modulo Theories and Optimization Problems.
In *Proc. Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *LNCS*. Springer, 2006.

[45] R. S. Oliver, S. S. Craciunas, and W. Steiner.
IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding.
In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 13–24, April 2018.

[46] A. Passerini.
Learning Modulo Theories.
In C. Bessiere, L. D. Raedt, L. Kotthoff, S. Nijssen, B. O'Sullivan, and D. Pedreschi, editors, *Data Mining and Constraint Programming - Foundations of a Cross-Disciplinary Approach*, volume 10101 of *Lecture Notes in Computer Science*, pages 113–146. Springer, 2016.

[47] R. Raman and I. Grossmann.
Modelling and computational techniques for logic based integer programming.
Computers and Chemical Engineering, 18(7):563 – 578, 1994.

[48] S. Ratschan.
Simulation Based Computation of Certificates for Safety of Dynamical Systems.
In A. Abate and G. Geeraerts, editors, *Formal Modeling and Analysis of Timed Systems - 15th International Conference, FORMATS 2017, Berlin, Germany, September 5-7, 2017, Proceedings*, volume 10419 of *Lecture Notes in Computer Science*, pages 303–317. Springer, 2017.

References VII

[49] D. Rayside, H.-C. Estler, and D. Jackson.
The Guided Improvement Algorithm for Exact, General-Purpose, Many-Objective Combinatorial Optimization.
Technical report, Massachusetts Institute of Technology, Cambridge, 07 2009.

[50] S. F. Roselli, K. Bengtsson, and K. Åkesson.
SMT Solvers for Job-Shop Scheduling Problems: Models Comparison and Performance Evaluation.
In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 547–552, Aug 2018.

[51] N. W. Sawaya and I. E. Grossmann.
A cutting plane method for solving linear generalized disjunctive programming problems.
Comput Chem Eng, 29(9):1891–1913, 2005.

[52] R. Sebastiani and S. Tomasi.
Optimization in SMT with LA(Q) Cost Functions.
In *IJCAR*, volume 7364 of *LNAI*, pages 484–498. Springer, July 2012.

[53] R. Sebastiani and S. Tomasi.
Optimization Modulo Theories with Linear Rational Costs.
ACM Transactions on Computational Logics, 16(2), March 2015.

[54] R. Sebastiani and P. Trentin.
OptiMathSAT: A Tool for Optimization Modulo Theories.
In *Proc. International Conference on Computer-Aided Verification, CAV 2015*, volume 9206 of *LNCS*. Springer, 2015.

[55] R. Sebastiani and P. Trentin.
Pushing the Envelope of Optimization Modulo Theories with Linear-Arithmetic Cost Functions.
In *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'15*, volume 9035 of *LNCS*. Springer, 2015.

[56] R. Sebastiani and P. Trentin.
On Optimization Modulo Theories, MaxSMT and Sorting Networks.
In *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'17*, volume 10205 of *LNCS*. Springer, 2017.

References VIII

[57] R. Sebastiani and P. Trentin.
OptiMathSAT: A Tool for Optimization Modulo Theories.
Journal of Automated Reasoning, Dec 2018.

[58] C. Sinz.
Towards an Optimal CNF Encoding of Boolean Cardinality Constraints.
In P. van Beek, editor, *CP*, volume 3709 of *LNCS*, pages 827–831. Springer, 2005.

[59] S. Teso, R. Sebastiani, and A. Passerini.
Structured Learning Modulo Theories.
Artificial Intelligence Journal, 2015.
To appear.

[60] S. Teso, R. Sebastiani, and A. Passerini.
Structured learning modulo theories.
Artif. Intell., 244:166–187, 2017.

[61] P. Trentin and R. Sebastiani.
Optimization Modulo the Theory of Floating-Point Numbers.
In *Proc. Int. Conference on Automated Deduction, CADE 27*, LNCS/LNAI. Springer, 2019.
To appear.

[62] S. Wolfman and D. Weld.
The LPSAT Engine & its Application to Resource Planning.
In *Proc. IJCAI*, 1999.

Outline

Outline

Solving OMT($\mathcal{LRA}$) [52, 53]

General idea

Combine standard SMT and LP minimization techniques.

Offline Schema

SMT solver and LP minimizer used as blackbox procedures.

$\Longrightarrow$ no need to hack the code of the SMT solver

Inline Schema

Search for minimum integrated inside the CDCL loop of the SMT solver.

Solving OMT($\mathcal{LRA}$) [52, 53]

General idea

Combine standard SMT and LP minimization techniques.

Offline Schema

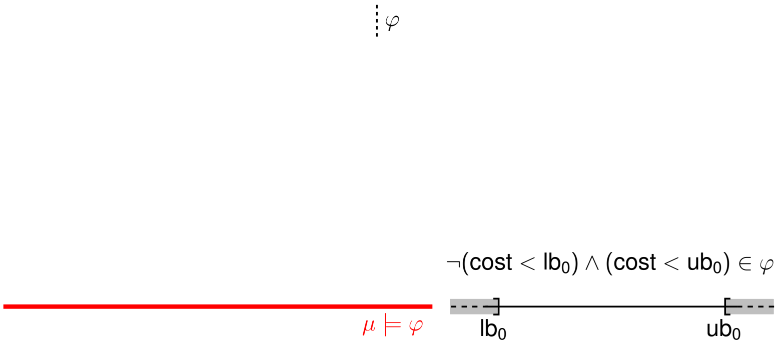
SMT solver and LP minimizer used as blackbox procedures.

$\Longrightarrow$ no need to hack the code of the SMT solver

Inline Schema

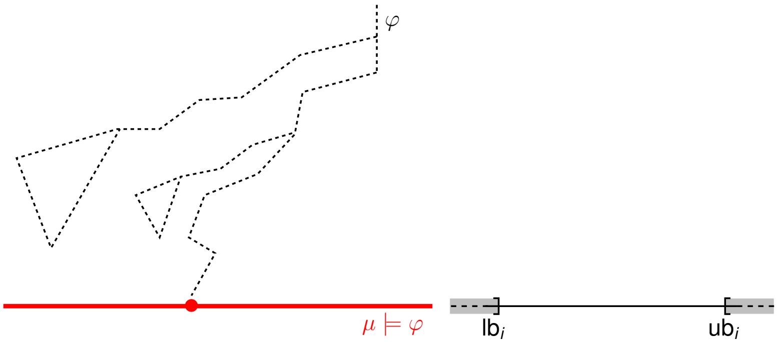
Search for minimum integrated inside the CDCL loop of the SMT solver.

Inline Version: Linear-Search Strategy



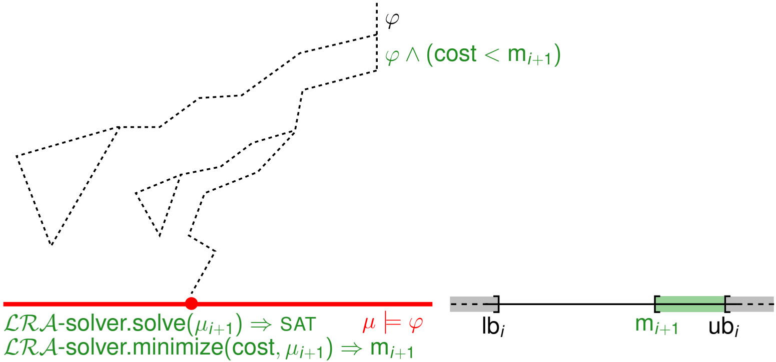
- Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of $\mathcal{LRA}$-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $\mathcal{LRA}$-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion

Inline Version: Linear-Search Strategy



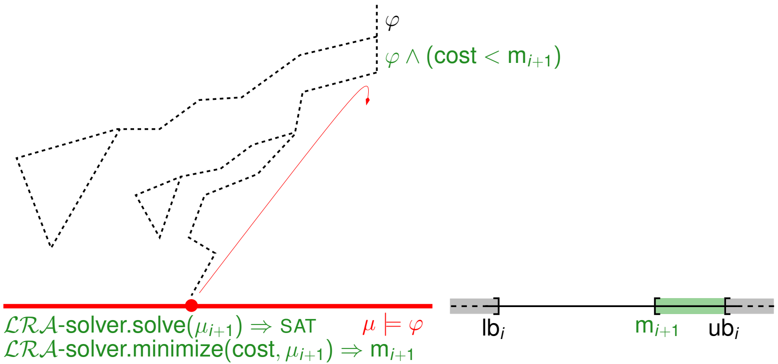
- Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of $\mathcal{LRA}$-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $\mathcal{LRA}$-checking (early-pruning) plays the role of "bounding" in a Branch & Bound fashion

Inline Version: Linear-Search Strategy



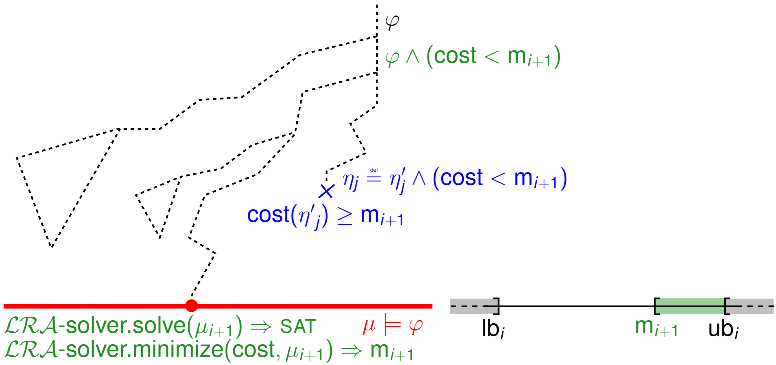
- Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of $\mathcal{LRA}$-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $\mathcal{LRA}$-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion

Inline Version: Linear-Search Strategy



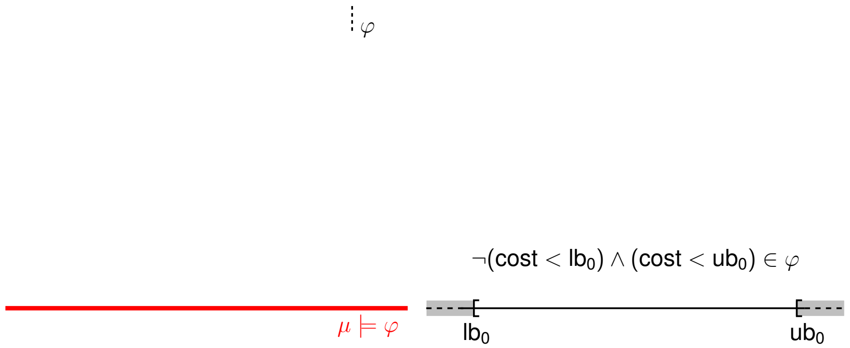
- Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of $\mathcal{LRA}$-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $\mathcal{LRA}$-checking (early-pruning) plays the role of "bounding" in a Branch & Bound fashion

Inline Version: Linear-Search Strategy



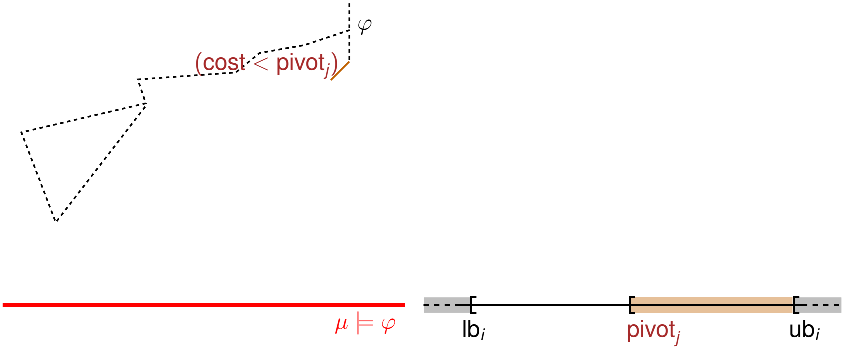
- Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of $\mathcal{LRA}$-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $\mathcal{LRA}$-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion

Inline Version: Binary-Search Strategy



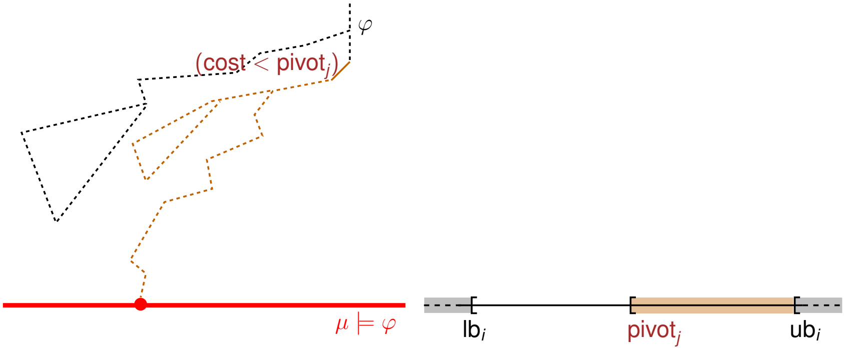
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Inline Version: Binary-Search Strategy



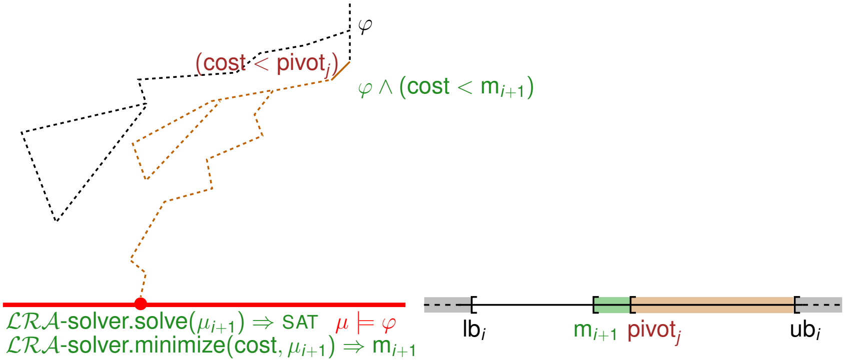
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot$_j$ and decide (cost $<$ pivot$_j$)

Inline Version: Binary-Search Strategy



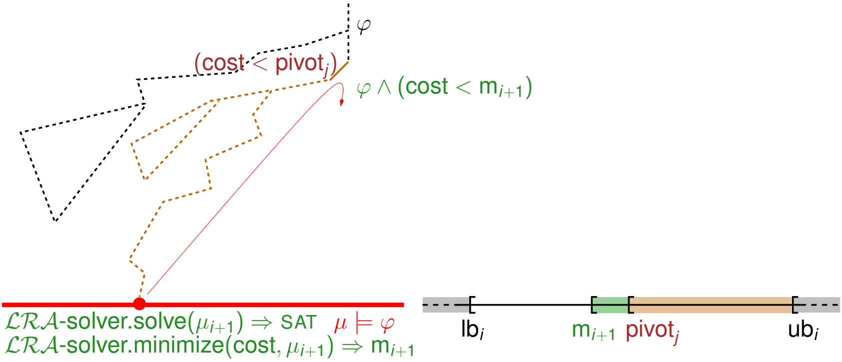
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Inline Version: Binary-Search Strategy



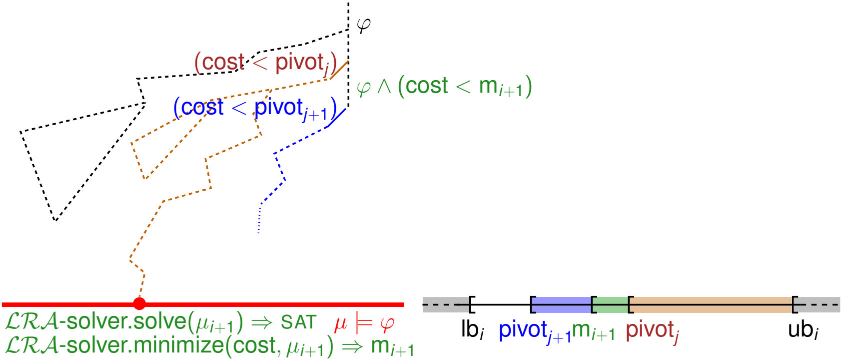
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Inline Version: Binary-Search Strategy



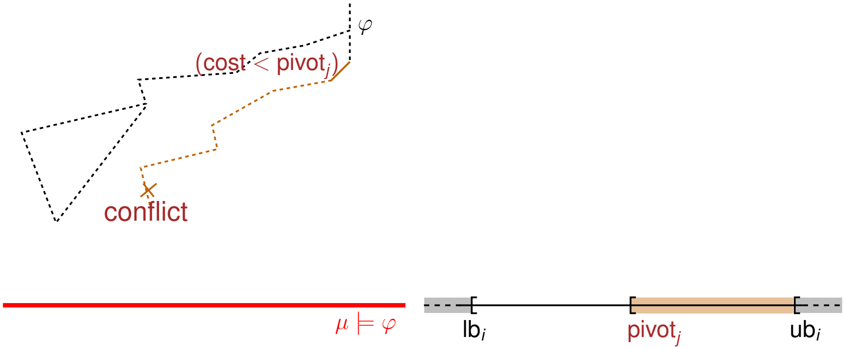
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Inline Version: Binary-Search Strategy



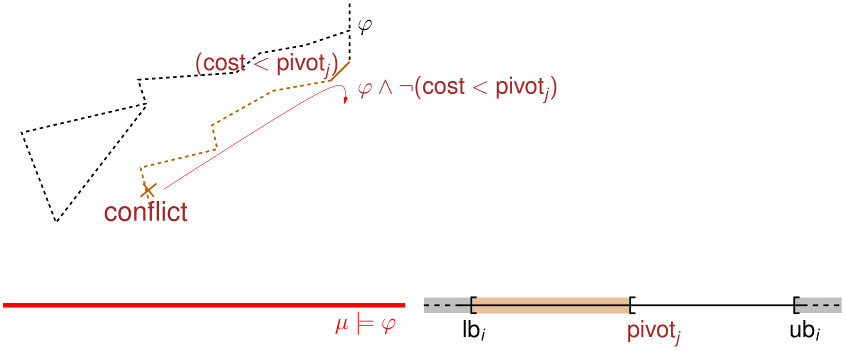
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Inline Version: Binary-Search Strategy



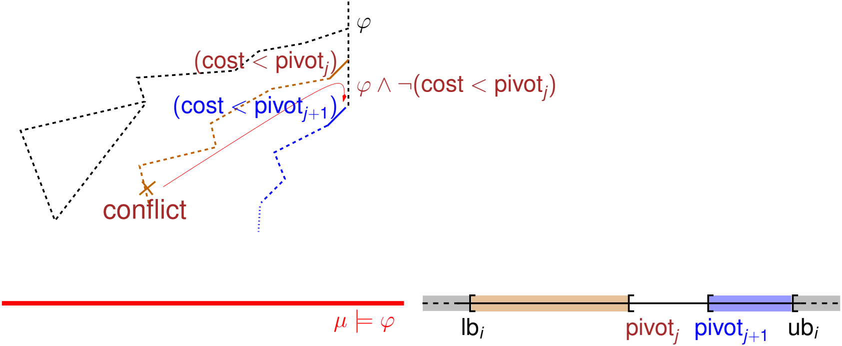
- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Inline Version: Binary-Search Strategy



- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot$_j$ and decide (cost < pivot$_j$)

Inline Version: Binary-Search Strategy



- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot_j and decide $(\text{cost} < \text{pivot}_j)$

Outline

OMT($\mathcal{BV}$)

back

Minimization of an unsigned Bit-Vector

Given a pair $\langle\varphi, \text{cost}\rangle$, where cost $\stackrel{\text{def}}{=}$ [cost[0], ..., cost[$n-1$]] is an **unsigned** $\mathcal{BV}$ of n bits:

- **Reduction to:**
  - Lexicographic OMT: $\langle\varphi, \{\text{cost}[0] \neq 0, ..., \text{cost}[n-1] \neq 0\}\rangle_{\mathcal{L}}$
  - MaxSMT [16, 17]: $\langle\varphi, \bigcup_{i=0}^{i=n-1}\langle\text{cost}[i] \neq 0, 1\rangle\rangle$
- **OMT-based Approach:** linear-search, binary-search and adaptive-search
- **Ad-Hoc Algorithms:**
  - OBV-WA [40]
    - each cost[i] transformed into a *high-priority* decision variable
    - the *phase-saving* of each cost[i] initialized to 0
  - OBV-BS [40]
    - binary search over the bits [cost[0], ..., cost[$n-1$]]
    - at most *n incremental* calls to the underlying SMT solver

Question:
How to deal with other $\mathcal{BV}$ goals?

- signed vs. unsigned
- maximization vs minimization

OMT($\mathcal{BV}$) - Signed/Unsigned $\mathcal{BV}$ [61]

Example: encoding of a 8-bits Bit-Vector

Unsigned:

0	1	2	3	4	5	6	7		
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	255	Positive
1	1	1	1	1	1	1	0	254	
...									
1	0	0	0	0	0	0	1	129	
1	0	0	0	0	0	0	0	128	
0	1	1	1	1	1	1	1	127	
0	1	1	1	1	1	1	0	126	
...									
0	0	0	0	0	0	0	1	1	
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	

Signed: (Two's complement)

0	1	2	3	4	5	6	7		
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	127	Positive
0	1	1	1	1	1	1	0	126	
...									
0	0	0	0	0	0	0	1	1	
0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	-1	Negative
1	1	1	1	1	1	1	0	-2	
...									
1	0	0	0	0	0	0	1	-127	
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	-128	

Attractor *attr* **for** cost**:** when minimizing, it's the **smallest** $\mathcal{BV}$-value of the same sort of cost.

- it's the ideal result of the optimization search
- depends on signed/unsigned

[Dual for Maximization]

OMT($\mathcal{BV}$) - Signed/Unsigned $\mathcal{BV}$ [61]

Reduction to unsigned $\mathcal{BV}$ (minimization)

Given an *attractor attr* for cost, both $\mathcal{BV}$s of n bits, replace cost with

$$\text{cost } \mathbf{xor}_n \ attr$$

Example: maximization of a signed 8-bits Bit-Vector

Before: cost

0	1	2	3	4	5	6	7		
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	127	Positive
0	1	1	1	1	1	1	0	126	
...									
0	0	0	0	0	0	0	1	1	
0	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	−1	Negative
1	1	1	1	1	1	1	0	−2	
...									
1	0	0	0	0	0	0	1	−127	
1	0	0	0	0	0	0	0	−128	

After: cost $\mathbf{xor}_8$ #*b*0111111

0	1	2	3	4	5	6	7		
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	127	Positive
0	0	0	0	0	0	0	1	126	
...									
0	1	1	1	1	1	1	0	1	
0	1	1	1	1	1	1	1	0	
1	0	0	0	0	0	0	0	−1	Negative
1	0	0	0	0	0	0	1	−2	
...									
1	1	1	1	1	1	1	0	−127	
1	1	1	1	1	1	1	1	−128	

OMT($\mathcal{FP}$) [61]

back

Goal: find a model $\mathcal{M}$ of φ for which the value of cost is minimum.

Simplification: $\exists\mathcal{M}$ s.t. $\mathcal{M} \models \varphi$ and $\mathcal{M}(\text{cost}) \neq$ NaN.
$\Longrightarrow$ replace φ with $\varphi \wedge \text{cost} \neq$ NaN

$\mathcal{FP}$ Minimization Approaches

- **Reduction to Bit-Vector Optimization:**
  - $\mathcal{BV}$ and $\mathcal{FP}$ are not Nelson-Oppen disjoint!
    $\Longrightarrow$ can only use eager $\mathcal{BV}/\mathcal{FP}$ SMT-solving approach
- **OMT-based Approach:** linear-search, binary-search and adaptive-search
- **Ad-Hoc Algorithms:**
  - OFP-BS (based on OBV-BS [40])
    - binary search over the bits [cost[0], ..., cost[$n-1$]]
    - at most *n incremental* calls to the underlying SMT solver

OMT($\mathcal{FP}$) [61]

Example: Encoding of a $\mathcal{FP}_{\langle 3,5 \rangle}$

0	1	2	3	4	5	6	7		
0	1	1	1	0	0	0	0	$+\infty$	
0	1	1	0	1	1	1	1	$^{31}/_{2}$	
				…					Positive
0	0	0	0	0	0	0	1	$^{1}/_{64}$	
0	0	0	0	0	0	0	0	$+0$	
1	0	0	0	0	0	0	0	-0	
1	0	0	0	0	0	0	1	$-^{1}/_{64}$	
				…					Negative
1	1	1	0	1	1	1	1	$-^{31}/_{2}$	
1	1	1	1	0	0	0	0	$-\infty$	

Minimization in the

- **Positive Domain**, go towards

0	0	0	0	0	0	0	0	$+0$

- **Negative Domain**, go towards

1	1	1	1	1	1	1	1	NaN

- **unless** the exponent is all **1s**, then go towards

?	1	1	1	0	0	0	0	$+\infty/-\infty$

Dynamic Attractor $attr_{\tau_k}$ **for** cost**:** given an assignment τ_k to the first k bits of cost, it's the **smallest** $\mathcal{FP}$-value different from NaN s.t.

$$\forall_{i=0}^{i=k-1} attr_{\tau_k}[i] = \tau_k[i]$$

- The ideal result of the optimization wrt. **current** search horizon

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k])$ **then**
  SMT.INCREMENTAL_CHECK($\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k]$) // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		$\tau_\mathbf{k}$	**attr**$_{\tau_\mathbf{k}}$		
0	0 1 1 0 1 1 1 1	$^{31}/_{2}$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_{2}$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ **skip**
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **skip**
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **end.**

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** ($\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k]$) **then**
  SMT.INCREMENTAL_CHECK($\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k]$) // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		$\tau_\mathbf{k}$	**attr**$_{\tau_\mathbf{k}}$		
0	0 1 1 0 1 1 1 1	$^{31}/_{2}$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_{2}$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ skip
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ skip
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ end.

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k])$ **then**
  SMT.INCREMENTAL_CHECK($\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k]$) // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		τ_k	$attr_{\tau_k}$		
0	0 1 1 0 1 1 1 1	$^{31}/_{2}$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_{2}$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ skip
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ skip
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ end.

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k])$ **then**
  SMT.INCREMENTAL_CHECK($\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k]$) // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		τ_k	$attr_{\tau_k}$		
0	0 1 1 0 1 1 1 1	$^{31}/_{2}$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_{2}$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ **skip**
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **skip**
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **end.**

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k])$ **then**
  SMT.INCREMENTAL_CHECK$(\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k])$ // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		τ_k	$attr_{\tau_k}$		
0	0 1 1 0 1 1 1 1	$^{31}/_{2}$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_{2}$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ **skip**
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ skip
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ end.

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k])$ **then**
  SMT.INCREMENTAL_CHECK$(\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k])$ // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		τ_k	$attr_{\tau_k}$		
0	0 1 1 0 1 1 1 1	$^{31}/_2$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_2$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ **skip**
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **skip**
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ end.

OMT($\mathcal{FP}$) - OFP-BS [61]

Idea: Use $attr_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq attr_{\tau_k}[k])$ **then**
  SMT.INCREMENTAL_CHECK($\varphi \wedge \tau_k \wedge \text{cost}[k] = attr_{\tau_k}[k]$) // try improve cost
  - UNSAT $\Longrightarrow$ update τ_k and $attr_{\tau_k}$
  - SAT $\Longrightarrow$ update τ_k and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

k	$\mathcal{M}$(cost)		τ_k	$attr_{\tau_k}$		
0	0 1 1 0 1 1 1 1	$^{31}/_{2}$		1 1 1 1 0 0 0 0	$-\infty$	$\Longrightarrow$ UNSAT
1	0 1 1 0 1 1 1 1	$^{31}/_{2}$	0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ SAT
2	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ **skip**
6	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0	0 0 0 0 0 0 0 0	$+0$	$\Longrightarrow$ UNSAT
7	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **skip**
8	0 0 0 0 0 0 1 0	$^{1}/_{32}$	0 0 0 0 0 0 1 0	0 0 0 0 0 0 1 0	$^{1}/_{32}$	$\Longrightarrow$ **end.**

Outline

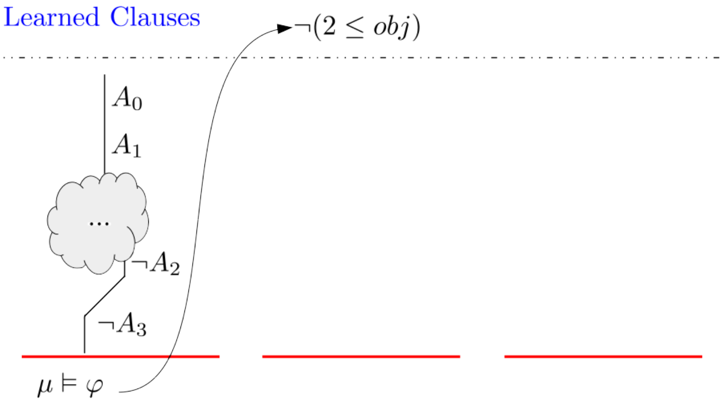
Running Example: performance bottleneck

back

Problem:

- $\langle \varphi, \texttt{min}(\text{cost}) \rangle$, where cost $:= w \cdot \sum_{i=0}^{n-1} A_i$, currently $obj = k \cdot w$
- OPTIMIZATION STEP: learn $\neg(k \cdot w \leq \text{cost})$ and restart/jump to *level* 0

Example: with $k = 2$, $w = 1$ and $n = 4$



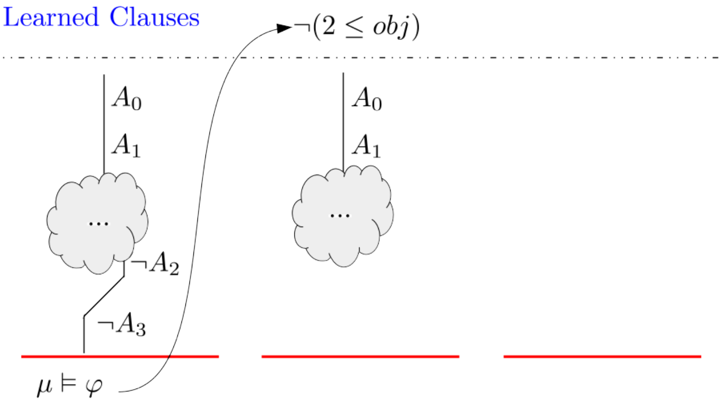
Running Example: performance bottleneck

back

Problem:

- $\neg(k \leq \text{cost})$ causes the inconsistency of $\binom{n}{k}$ truth assignments satisfying exactly k variables in $A_0, ..., A_{n-1}$

Example: with $k = 2$, $w = 1$ and $n = 4$



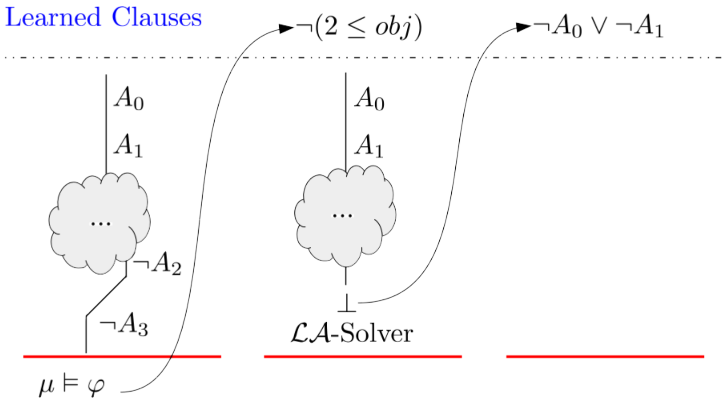
Running Example: performance bottleneck

back

Problem:

- $\neg(k \leq \text{cost})$ causes the inconsistency of $\binom{n}{k}$ truth assignments satisfying exactly k variables in $A_0, ..., A_{n-1}$
  $\Longrightarrow$ inconsistency is not revealed by Boolean Constraint Propagation

Example: with $k = 2$, $w = 1$ and $n = 4$



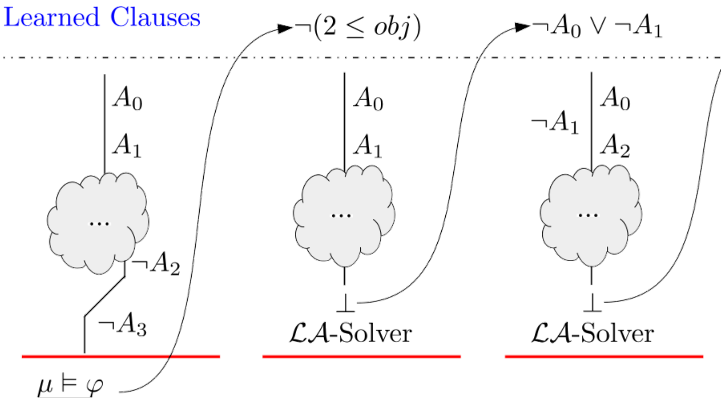
Running Example: performance bottleneck

back

Problem:

- up to $\binom{n}{k}$ (expensive) calls to the $\mathcal{LA}$-Solver required

Example: with $k = 2$, $w = 1$ and $n = 4$



Solution: OMT + sorting networks [56]

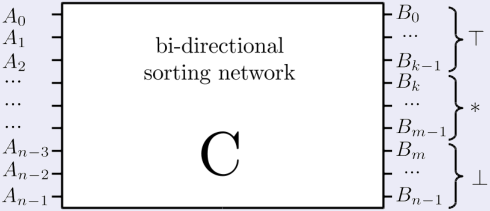
Contribution:

Enriched OMT encoding with bidirectional **sorting networks** [58, 10].

Approach:

Given $\langle \varphi, \text{cost} \rangle$, cost $:= w \cdot \sum_{i=0}^{n-1} A_i$, and a bi-directional **sorting network** relation $C(A_0, ..., A_{n-1}, B_0, ..., B_{n-1})$ s.t.

- k A_i's are $\top \iff \{B_0, ..., B_{k-1}\}$ are $\top$,
- $m - k$ A_i's are $* \iff \{B_k, ..., B_{m-1}\}$ are $*$,
- $n - m$ A_i's are $\bot \iff \{B_m, ..., B_{n-1}\}$ are $\bot$



then we encode it as $\langle \varphi', \text{cost} \rangle$, where

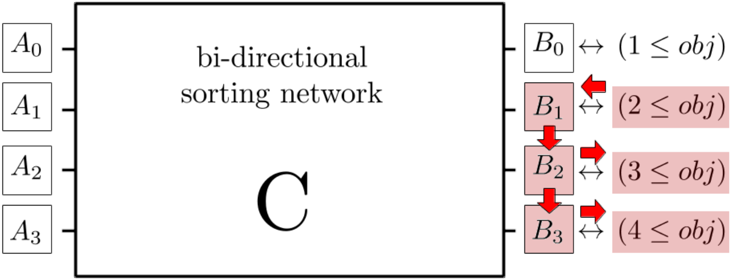
$$\varphi' := \varphi \wedge C(A_0, ..., A_{n-1}, B_0, ..., B_{n-1}) \wedge \bigwedge_{i=0}^{n-1} B_i \leftrightarrow ((i+1) \cdot w \leq \text{cost}) \wedge \bigwedge_{i=0}^{n-2} B_{i+1} \rightarrow B_i$$

Properties: OMT + sorting networks [56]

Properties:

- if $(k \cdot w \leq \text{cost}) = \bot$, then by BCP $\forall i \in [k, n].B_{i-1} = \bot$

Example: with $k = 2$, $w = 1$ and $n = 4$



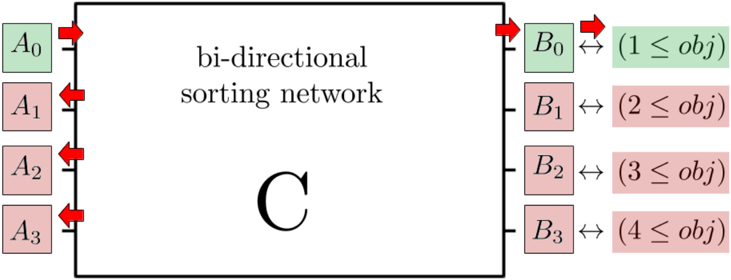
Properties: OMT + sorting networks [56]

Properties:

- if $(k \cdot w \leq \text{cost}) = \perp$, then by BCP $\forall i \in [k, n]. B_{i-1} = \perp$
- as soon as $k-1$ A_i are assigned $\top$
  $\Longrightarrow$ all others are unit-propagated to $\perp$

Dual if $(k \cdot w \leq \text{cost}) = \top$.

Example: with $k = 2$, $w = 1$ and $n = 4$

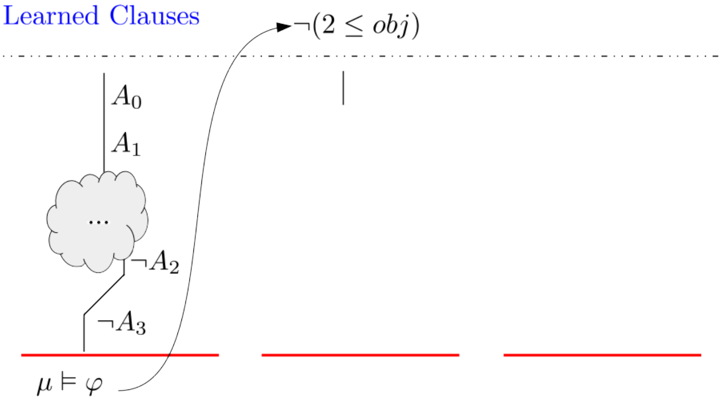


Example: OMT with sorting networks

back

- Optimization Step: learn $\neg(k \cdot w \leq \text{cost})$ and restart/jump to *level* 0

Example: with $k = 2$, $w = 1$ and $n = 4$

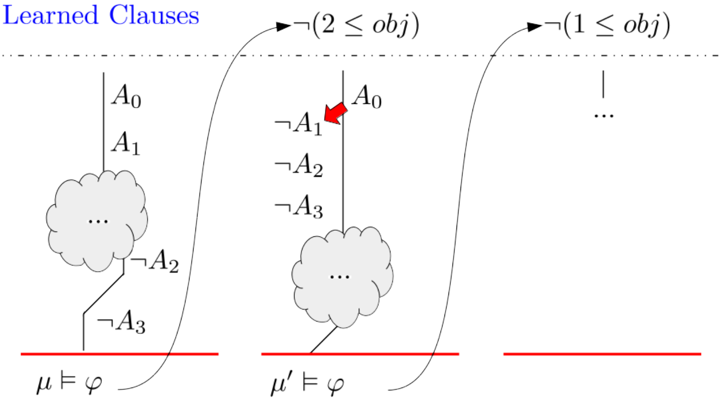


Example: OMT with sorting networks

back

- OPTIMIZATION STEP: learn $\neg(k \cdot w \leq \text{cost})$ and restart/jump to *level* 0
- as soon as $k-1$ A_i are assigned $\top$
  $\Longrightarrow$ all others are unit-propagated to $\bot$

Example: with $k = 2$, $w = 1$ and $n = 4$



Solution: Combine OMT with Sorting Networks

OPTIMATHSAT: sorting networks implemented

- **Bi-directional Sequential Counter** [58], in $O(n^2)$ but incremental
  sum of A_i's, unary representation
- **Bi-directional Cardinality Network** [10, 6], in $O(n\,log^2 n)$
  based on *merge-sort* algorithm

Generalization

The same performance issue occurs for $\langle \varphi, \text{cost} \rangle$, where

$$\text{cost} = \tau_1 + ... + \tau_m,$$

$$\forall_j \in [1, m].\ (\tau_j = w_j \cdot \sum_{i=0}^{i=k_j} A_{ji})\ \wedge\ (0 \leq \tau_j) \wedge (\tau_j \leq w_j \cdot k_j)$$

Solution:

- use a separate sorting circuit for each term τ_j
- add clauses in the form $(w_j \cdot i \leq \tau_j) \rightarrow (w_j \cdot i \leq \text{cost})$

Outline

MaxRes: Maximum Resolution [16]

back

Idea: given a MaxSMT $\langle \varphi_h, \varphi_s \rangle$, treat both φ_h and φ_s as hard clauses.

Analyze conflict τ, where $\tau \stackrel{\text{def}}{=} \tau_h \cup \tau_s$, $\tau_h \subseteq \varphi_h$ and $\tau_s \subseteq \varphi_s$

- if $\tau_s = \emptyset \Longrightarrow$ input problem is unsatisfiable
- else **let** $w_{min} \stackrel{\text{def}}{=} \min(w_i \mid \langle C_i, w_i \rangle \in \tau_s)$ and **relax** the problem:
  - Learn conflict-clause and replace soft-clauses

$$\begin{aligned} \varphi_h \quad &:= \quad \varphi_h \cup \bigvee\nolimits_{\langle C_i, w_i \rangle \in \tau_s} \neg C_i \\ \varphi_s \quad &:= \quad \varphi_s \setminus \tau_s \cup \bigcup\nolimits_{\langle C_i, w_i \rangle \in \tau_s} \langle C_i, w_i - w_{min} \rangle \ \textbf{if} \ \ w_i - w_{min} > 0 \end{aligned}$$

  - if $\mid \tau_s \mid > \ 1 \Longrightarrow$ add compensation clauses

$$\begin{aligned} \varphi_h \quad &:= \quad \varphi_h \cup \bigcup\nolimits_{\langle C_i, w_i \rangle \in \tau_s} . B_i \rightarrow (B_{i-1} \wedge C_i) \\ & \qquad \qquad // \ B_0 := \top, \forall_{i>0}. B_i \text{ is fresh Boolean var} \\ \varphi_s \quad &:= \quad \varphi_s \cup \bigcup\nolimits_{\langle C_i, w_i \rangle \in \{\tau_s \setminus \langle C_1, w_1 \rangle\}} . \langle B_{i-1} \vee C_i, w_{min} \rangle \end{aligned}$$

No Conflict: optimal solution

Outline

Extended SMT-LIBv2 Interface [57]

back

```
(minimize <term> [:id <string>] [:signed]
            [:lower <const_term>] [:upper <const_term>])
(maximize <term> [:id <string>] [:signed]
            [:lower <const_term>] [:upper <const_term>])

(minmax <term> ... <term> [:id <string>] [:signed]
            [:lower <const_term>] [:upper <const_term>])
(maxmin <term> ... <term> [:id <string>] [:signed]
            [:lower <const_term>] [:upper <const_term>])

(assert-soft <term> [:id <string>] [:weight <const_term>])

(check-sat)
(check-allsat (<const_term> ... <const_term>))

(get-objectives)
(load-objective-model <numeral>)
```

Outline

Pareto OMT

back

Definitions:

- A model $\mathcal{M}$ *Pareto-dominates* $\mathcal{M}'$ iff

  $$\forall i.\mathcal{M}(\text{cost}_i) \leq \mathcal{M}'(\text{cost}_i)$$

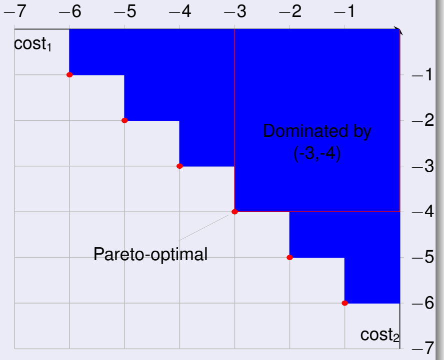
  and

  $$\exists j.\mathcal{M}(\text{cost}_j) < \mathcal{M}'(\text{cost}_j)$$

  (dual for maximization)

- $\mathcal{M}$ is *Pareto-optimal* iff it is not Pareto-dominated by any $\mathcal{M}'$.

Example: $\langle \varphi, \{\text{cost}_1, \text{cost}_2\}\rangle_{\mathcal{P}}$



Goal: given a pair $\langle \varphi, \mathcal{O}\rangle_{\mathcal{P}}$, where $\mathcal{O} \stackrel{\text{def}}{=} \{\text{cost}_1, ..., \text{cost}_N\}$

- find the set of Pareto-optimal models $\{\mathcal{M}_1, ..., \mathcal{M}_M\}$ (i.e. the *Pareto front*)

Pareto OMT: Guided Improvement Algorithm (GIA)

Guided Improvement Algorithm [49, 16]

Given a pair $\langle \varphi, \mathcal{O} \rangle_{\mathcal{P}}$, where $\mathcal{O} \stackrel{\text{def}}{=} \{\text{cost}_1, ..., \text{cost}_N\}$:

- start from random model $\mathcal{M}$ of φ
- **loop:** look for a model $\mathcal{M}'$ of φ that *Pareto-dominates* $\mathcal{M}$
  $\Longrightarrow$ if any, replace $\mathcal{M}$ with $\mathcal{M}'$ and keep looking
- block solutions Pareto-dominated by $\mathcal{M}$
- **repeat**

Infinite Loop:

- some cost_i is unbounded
- some cost_j can always be improved by an *infinitesimal value* (e.g. OMT($\mathcal{LRA}$))

Also: $\mathcal{T}$-minimization procedure not used
$\Longrightarrow$ the same μ may be visited multiple times by CDCL/SAT engine

Pareto OMT: Lexicographic GIA

Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\} \rangle_{\mathcal{L}}$, then $\mathcal{M}$ is also Pareto-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\} \rangle_{\mathcal{P}}$.

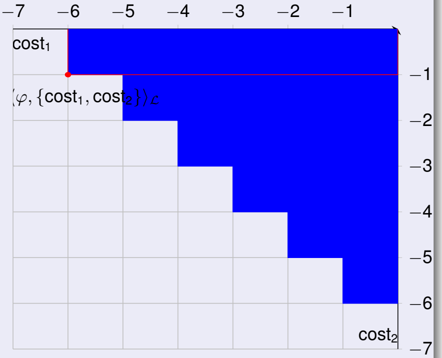
Idea:

- Shuffle $\{\text{cost}_1, ..., \text{cost}_N\}$
  $\Longrightarrow$ *explore from different directions*
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn

  $$\bigvee_{i=1}^{i=N} (\text{cost}_i < \mathcal{M}[\text{cost}_i])$$

  to block Pareto-dominated solutions
- **repeat**

Example: $\langle \varphi, \{\text{cost}_1, \text{cost}_2\} \rangle_{\mathcal{P}}$



Pareto OMT: Lexicographic GIA

Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\} \rangle_{\mathcal{L}}$, then $\mathcal{M}$ is also Pareto-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\} \rangle_{\mathcal{P}}$.

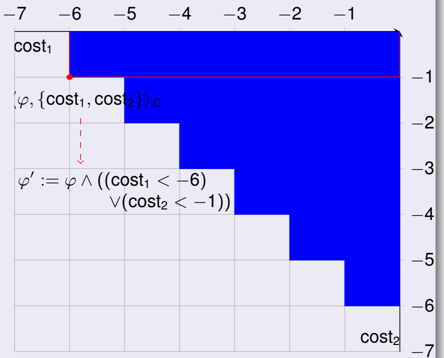
Idea:

- Shuffle $\{\text{cost}_1, ..., \text{cost}_N\}$
  $\Longrightarrow$ *explore from different directions*
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn

  $$\bigvee_{i=1}^{i=N} (\text{cost}_i < \mathcal{M}[\text{cost}_i])$$

  to block Pareto-dominated solutions
- **repeat**

Example: $\langle \varphi, \{\text{cost}_1, \text{cost}_2\} \rangle_{\mathcal{P}}$



Pareto OMT: Lexicographic GIA

Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle\varphi, \{\text{cost}_1, ..., \text{cost}_N\}\rangle_{\mathcal{L}}$, then $\mathcal{M}$ is also Pareto-optimal for $\langle\varphi, \{\text{cost}_1, ..., \text{cost}_N\}\rangle_{\mathcal{P}}$.

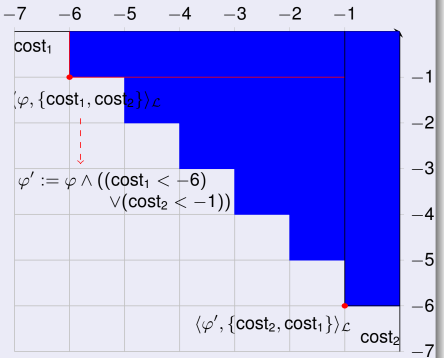
Idea:

- Shuffle $\{\text{cost}_1, ..., \text{cost}_N\}$
  $\Longrightarrow$ *explore from different directions*
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn

$$\bigvee_{i=1}^{i=N} (\text{cost}_i < \mathcal{M}[\text{cost}_i])$$

  to block Pareto-dominated solutions
- **repeat**

Example: $\langle\varphi, \{\text{cost}_1, \text{cost}_2\}\rangle_{\mathcal{P}}$



Pareto OMT: Lexicographic GIA

Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle \varphi, \{cost_1, ..., cost_N\} \rangle_{\mathcal{L}}$, then $\mathcal{M}$ is also Pareto-optimal for $\langle \varphi, \{cost_1, ..., cost_N\} \rangle_{\mathcal{P}}$.

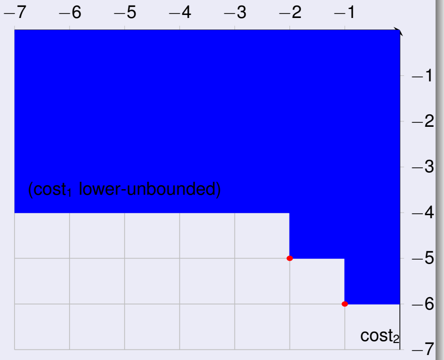
Idea:

- Shuffle $\{cost_1, ..., cost_N\}$
  $\Longrightarrow$ *explore from different directions*
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn

$$\bigvee_{i=1}^{i=N} (cost_i < \mathcal{M}[cost_i])$$

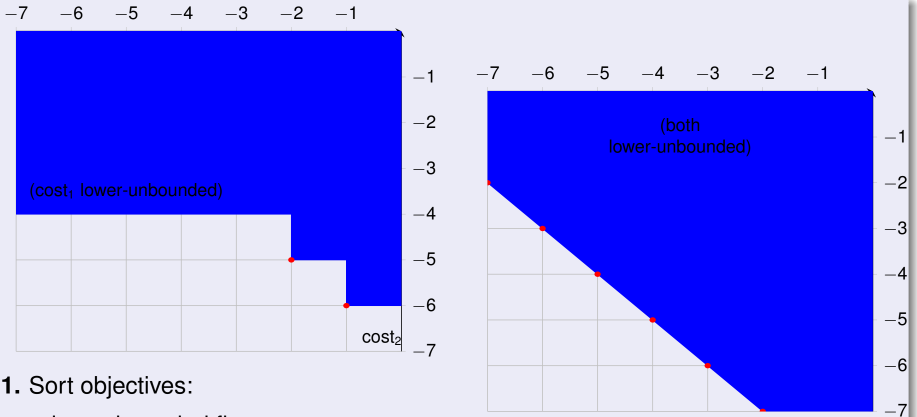
  to block Pareto-dominated solutions
- **repeat**

Example: $\langle \varphi, \{cost_1, cost_2\} \rangle_{\mathcal{P}}$



Problem: how to deal with unbounded objectives?

Pareto OMT: dealing with unbounded objectives

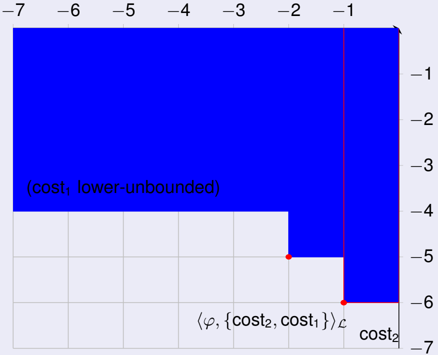


1. Sort objectives:

- lower-bounded first
- lower-unbounded last

before Lex. OMT.

Pareto OMT: dealing with unbounded objectives

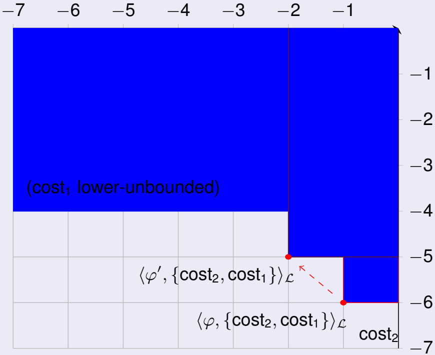


1. Sort objectives:

- lower-bounded first
- lower-unbounded last

before Lex. OMT.

Pareto OMT: dealing with unbounded objectives

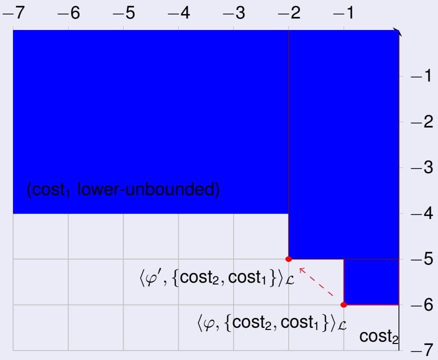


1. Sort objectives:

- lower-bounded first
- lower-unbounded last

before Lex. OMT.

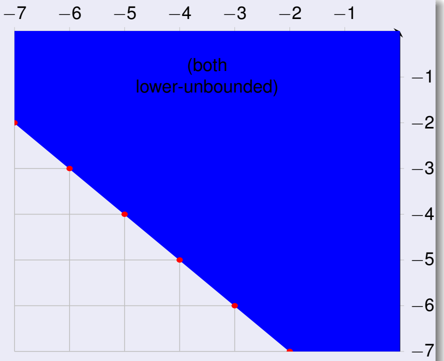
Pareto OMT: dealing with unbounded objectives



1. Sort objectives:

- lower-bounded first
- lower-unbounded last

before Lex. OMT.

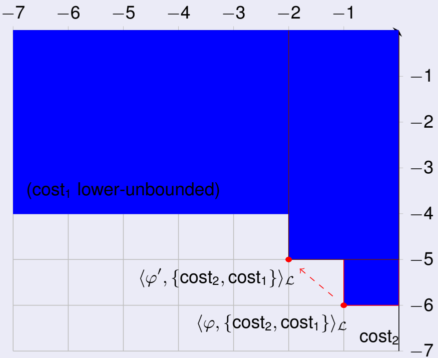


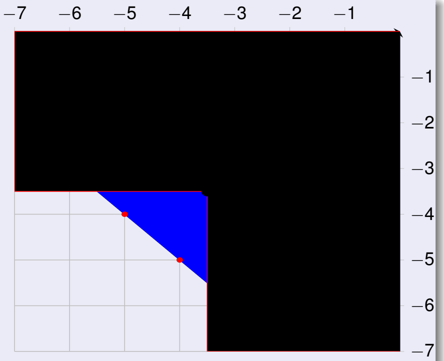
2. If Lex. OMT unbounded, (temporarily) learn:

$$\bigwedge_{i=1}^{i=N} (\text{cost}_i \leq \mathcal{M}[\text{cost}_i])$$

and try again.

Pareto OMT: dealing with unbounded objectives





1. Sort objectives:

- lower-bounded first
- lower-unbounded last

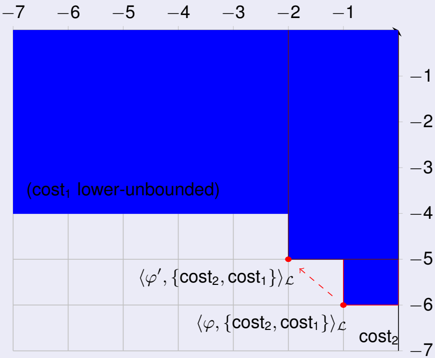
before Lex. OMT.

2. If Lex. OMT unbounded, (temporarily) learn:

$$\bigwedge_{i=1}^{i=N} (\text{cost}_i \leq \mathcal{M}[\text{cost}_i])$$

and try again.

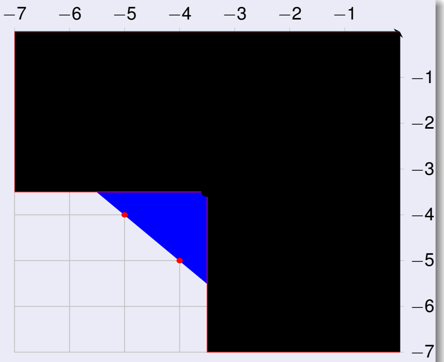
Pareto OMT: dealing with unbounded objectives



1. Sort objectives:

- lower-bounded first
- lower-unbounded last

before Lex. OMT.



2. If Lex. OMT unbounded, (temporarily) learn:

$$\bigwedge_{i=1}^{i=N} (\text{cost}_i \leq \mathcal{M}[\text{cost}_i])$$

and try again.

3. If Lex. OMT still unbounded, give up.