Reasoning about quantifiers in SMT: the QSMA algorithm[1]

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy, EU

Invited Keynote Speech, 23rd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)
Ames, IA, USA, 26 Oct. 2023

Also presented with title "The QSMA algorithm" at Dagstuhl Seminar 23471 "The Next Generation of Deduction Systems: from Composition to Compositionality" 21 Nov. 2023

[1] Based on joint work with S. Graham-Lengrand and Ch. Vauthier

Motivation

- Applications of automated reasoning
  (e.g., analysis, verification, synthesis of programs)
  need reasoners that
- Decide the satisfiability of formulas involving both
  - Quantifiers and
  - Defined symbols:
    Symbols defined in background theories

The big picture

Major research objectives:

1. Enriching theorem provers with built-in theories
2. Integrating theorem provers and SMT solvers
3. Endowing SMT solvers with quantifier reasoning

The QSMA algorithm contributes to Objective (3)

Quantifier elimination (QE)

- A theory $\mathcal{T}$ admits QE if
  for all formulas φ there exists a $\mathcal{T}$-equivalent quantifier-free (QF) formula F
- Reduce $\mathcal{T}$-satisfiability of formulas to that of QF formulas
- Few theories admit QE
- QE is prohibitively expensive:
  Exponential in LRA, doubly exponential in LIA
- Not a practical solution
- Practical solution: QSMA algorithm

General problem statement

Quantified satisfiability modulo theory and assignment

Given:

- A theory $\mathcal{T}$
- A formula φ with arbitrary quantification
- An initial assignment to Boolean or first-order subterms of φ

- Either find a $\mathcal{T}$-model of φ that extends the initial assignment
- Or report that none exists

The QSMA algorithm

A new algorithm for the satisfiability of a formula φ with arbitrary quantification (alternation is key) modulo:

- A theory $\mathcal{T}$ with unique $\mathcal{T}$-model $\mathcal{M}_0$
- An initial assignment to the free variables of φ

- $\mathcal{T}$ is complete:
  Consistent + for all sentences F either $\mathcal{T} \vdash F$ or $\mathcal{T} \vdash \neg F$
- $\mathcal{T}$-model: extension of $\mathcal{M}_0$ with an assignment to free variables

A satisfiable example in LRA

$$\varphi_1 = \exists x.\forall y.\exists z.\ z \geq 0 \ \wedge \ x \geq 0 \ \wedge \ y + z \geq 0$$

- Say we assign $x{\leftarrow}0$
- For all values for y there exists a satisfying z namely $z{\leftarrow}max(0, -y)$ (read $y + z \geq 0$ as $z \geq -y$)
- Therefore φ_1 is true in LRA

(Unique $\mathcal{T}$-model $\mathcal{M}_0$: for a sentence satisfiability, validity, and truth in $\mathcal{M}_0$ coincide)

An unsatisfiable example in LRA

$$\varphi_2 = \exists x . \forall y . \exists z .\ z \geq 0 \ \wedge \ x \geq 0 \ \wedge \ y + z \leq 0$$

- Say we assign $x \leftarrow n$ for some $n \geq 0$ (n is immaterial)
- For $y \leftarrow 1$ no value for z satisfies $z \geq 0 \wedge z \leq -1$
- Therefore φ_2 is false in LRA

(Unique $\mathcal{T}$-model $\mathcal{M}_0$: for a sentence unsatisfiability, invalidity, and falsity in $\mathcal{M}_0$ coincide)

High-level view of the QSMA algorithm

- Apply $\neg\neg$ to convert $\forall$ into $\exists$
- Use Boolean variables as proxies for quantified subformulas
- Recursive descent over tree structure of formula
- Remove one level/block of $\exists$-quantifier(s) and assign values to freed 1st-order variables and proxies
- Assignments come from underlying SMA solver that gets a formula and a prior assignment (initially the initial assignment)
- Model-based guidance to weed out large parts of the space of possible assignments

More about formula view and recursive descent

$$\varphi = \exists \bar{x}. F[\bar{z}, \bar{x}, \bar{p}]\{p_i \leftarrow \exists \bar{y}_i. G_i[\bar{z}, \bar{x}, \bar{y}_i]\}_{i=1}^{k} \text{ where } \bar{z} = FV(\varphi)$$

- F is QF as proxies p_i replace subformulas $\varphi_i = \exists \bar{y}_i. G_i[\bar{z}, \bar{x}, \bar{y}_i]$
- $FV(\varphi) = \emptyset$: quantified SMA problems when working subformulas under assignment to higher-level variables
- $p_i \leftarrow$true/false: try to show φ_i true / false
- p_i undefined: can be ignored
- φ is true under the initial assignment iff QSMA can extend the initial assignment to one satisfying
$$F[\bar{z}, \bar{x}, \bar{p}] \wedge \bigwedge_i (p_i \Leftrightarrow \varphi_i)$$

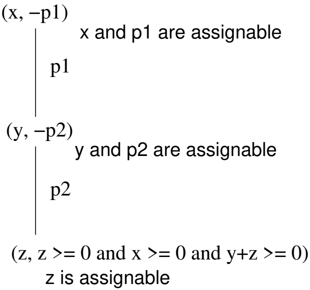
The satisfiable example in LRA done by QSMA

$$\varphi_1 = \exists x. \neg \exists y. \neg \exists z.\ z \geq 0 \ \wedge \ x \geq 0 \ \wedge \ y + z \geq 0$$

- $\exists x. \neg p_1$

  $p_1 = \exists y. \neg \exists z.\ z \geq 0 \ \wedge \ x \geq 0 \ \wedge \ y + z \geq 0$
- $x \leftarrow 0,\ p_1 \leftarrow \mathsf{false}$
- $\exists y. \neg p_2$

  $p_2 = \exists z.\ z \geq 0 \ \wedge \ x \geq 0 \ \wedge \ y + z \geq 0$
- $y \leftarrow n,\ p_2 \leftarrow \mathsf{true}$
- $\exists z. z \geq 0 \wedge x \geq 0 \wedge y{+}z \geq 0$
- $z \leftarrow max(0, -n)$
- *True*



The unsatisfiable example in LRA done by QSMA

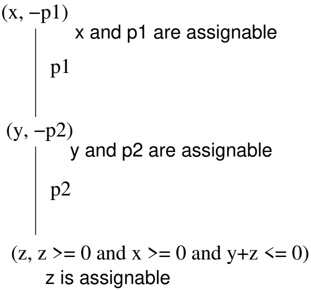
$$\varphi_2 = \exists x.\neg\exists y.\neg\exists z.\ z \geq 0\ \wedge\ x \geq 0\ \wedge\ y + z \leq 0$$

- $\exists x.\neg p_1$

  $p_1 = \exists y.\neg\exists z.\ z \geq 0\ \wedge\ x \geq 0\ \wedge\ y + z \leq 0$
- $x \leftarrow n\ (n \geq 0),\ p_1 \leftarrow \text{false}$
- $\exists y.\neg p_2$

  $p_2 = \exists z.\ z \geq 0\ \wedge\ x \geq 0\ \wedge\ y + z \leq 0$
- $y \leftarrow 1,\ p_2 \leftarrow \text{true}$
- $\exists z. z \geq 0 \wedge x \geq 0 \wedge y{+}z \leq 0$
- No z satisfies

  $z \geq 0 \wedge z \leq -1$
- *False*



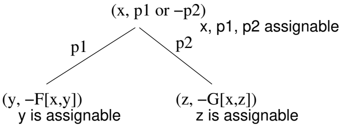
More general example

- QSMA handles arbitrary formulas
- Quantifiers in arbitrary positions:
  no need of prenex normal form
- $\varphi = \exists x.((\forall y.F[x,y]) \Rightarrow (\forall z.G[x,z]))$
  where F and G are QF
- Eliminating implication and universal quantifiers yields:
  $\varphi = \exists x.((\exists y.\neg F[x,y]) \vee (\neg\exists z.\neg G[x,z]))$

The more general example as done by QSMA

$$\varphi = \exists x.((\exists y.\neg F[x,y]) \vee (\neg\exists z.\neg G[x,z]))$$

- $\exists x.(p_1 \vee \neg p_2)$
  $p_1 = \exists y.\neg F[x,y]$
  $p_2 = \exists z.\neg G[x,z]$
- Assign x, p_1, p_2
- $p_1 \leftarrow$true: find a y satisfying $\neg F[x,y]$
- $p_2 \leftarrow$false: show that there is no z satisfying $\neg G[x,z]$



From formula to QSMA-tree

$$\varphi = \exists \bar{x}. F[\bar{z}, \bar{x}, \bar{p}]\{p_i \leftarrow \exists \bar{y}_i. G_i[\bar{z}, \bar{x}, \bar{y}_i]\}_{i=1}^{k}$$

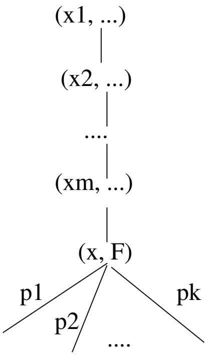
- QSMA-tree $\mathcal{G} = (\bar{z}, T)$ with rigid variables $\bar{z}$
- $k = 0$: T is a node labeled $(\bar{x}, F[\bar{z}, \bar{x}])$
- $k > 0$:
  - T has root labeled $(\bar{x}, F[\bar{z}, \bar{x}, \bar{p}])$ with k arcs labeled $p_1, \ldots, p_k$ to children $b_1, \ldots, b_k$
  - Child b_i labeled $(\bar{y}_i, G_i[\bar{z}, \bar{x}, \bar{y}_i])$ is root of QSMA-tree $\mathcal{G}_i = ((\bar{z}, \bar{x}), T_i)$ with rigid variables $\bar{z} \uplus \bar{x}$ for $\varphi_i = \exists \bar{y}_i. G_i[\bar{z}, \bar{x}, \bar{y}_i]$

Rigid variables and assignable variables

Rigid and assignable variables at a node

QSMA-tree $\mathcal{G} = (\bar{z}, T)$

- Node n labeled $(\bar{x}, F)$
- Local variables $n.\bar{x}$
- QF formula $n.F$
- k outgoing arcs labeled $p_1, \ldots, p_k$
- Assignable vars at n: $Var(n) = \bar{x} \uplus \{p_1, \ldots, p_k\}$
- Rigid vars at n: $Rigid(n) = \bar{z} \uplus \bar{x}_1 \uplus \ldots \uplus \bar{x}_m$
- $\mathcal{G}_n = (Rigid(n), T_n)$



From QSMA-tree back to formula

QSMA-tree: $\mathcal{G} = (\bar{z}, T)$

For all nodes n of T, the formula $n.\psi$ at node n:

- Node n is leaf labeled $(\bar{x}, F[\bar{z}, \bar{x}])$:
  $n.\psi = \exists \bar{x}. F[\bar{z}, \bar{x}]$
- Node n has label $(\bar{x}, F[\bar{z}, \bar{x}, \bar{p}])$ and children $b_1, \ldots, b_k$ via arcs (n, b_i) labeled p_i:
  $n.\psi = \exists \bar{x}. F[\bar{z}, \bar{x}, \bar{p}]\{p_i \leftarrow b_i.\psi\}_{i=1}^{k}$
  for $b_i.\psi$ the formula at node b_i

If $\mathcal{G}$ is the QSMA-tree for φ and r is the root of $\mathcal{G}$ then $r.\psi = \varphi$

Satisfaction of QSMA-tree

QSMA-tree $\mathcal{G} = (\bar{z}, T)$ with root r

- $\mathcal{M}$: extension of $\mathcal{M}_0$ to $Rigid(r) = \bar{z}$
- $\mathcal{M} \models \mathcal{G}$ if there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $Var(r)$ s.t.
  1. $\mathcal{M}' \models r.F$
  2. For all children b of r via arc (r, b) labeled p
     $\mathcal{M}'(p) =$ true iff $\mathcal{M}' \models \mathcal{G}_b$

If $\mathcal{M}'(p) =$ true: try to show $b.\psi$ true
If $\mathcal{M}'(p) =$ false: try to show $b.\psi$ false
If $\mathcal{M}'$ is partial and does not assign p: ignore $b.\psi$

Thm: $\mathcal{G}$ is the QSMA-tree for formula φ: $\mathcal{M} \models \mathcal{G}$ iff $\mathcal{M} \models \varphi$

Assigning variables in QSMA

Assume to have a solver for theory $\mathcal{T}$ (and model $\mathcal{M}_0$) offering a model extension function SMA:

- Given formula $\exists \bar{x}.F[\bar{z}, \bar{x}, \bar{p}]$, where F is QF and extension $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$,
- SMA$(F[\bar{z}, \bar{x}, \bar{p}], \mathcal{M})$ returns:
  - Either extension $\mathcal{M}'$ of $\mathcal{M}$ to $\bar{x} \uplus \bar{p}$ such that $\mathcal{M}' \models F[\bar{z}, \bar{x}, \bar{p}]$
  - Or nil if no such extension exists

- Testing all possible assignments: impossible, infinitely many
- Needed: model-based guidance

Under-approximations and over-approximations

Formula φ with $FV(\varphi) = \bar{z}$ $\llbracket\varphi\rrbracket$: set of models of φ

- Under-approximation of φ: QF formula U with $FV(U) = \bar{z}$
  for all extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$
  $\mathcal{M} \models U$ implies $\mathcal{M} \models \varphi$
  under-approximations help to return true
- Over-approximation of φ: QF formula O with $FV(O) = \bar{z}$
  for all extensions $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}$
  $\mathcal{M} \models \varphi$ implies $\mathcal{M} \models O$
  $\mathcal{M} \not\models O$ implies $\mathcal{M} \not\models \varphi$
  over-approximations help to return false

$$\llbracket U\rrbracket \subseteq \llbracket\varphi\rrbracket \subseteq \llbracket O\rrbracket$$

Under- and over-approximations in QSMA

- QSMA-tree $\mathcal{G} = (\bar{z}, T)$ for formula φ
- Given $\mathcal{M}$ extending $\mathcal{M}_0$ to $\bar{z}$, the QSMA algorithm determines whether $\mathcal{M} \models \mathcal{G}$:
  - For all nodes n of T maintain under-approximation $n.U$ of $n.\psi$ and over-approximation $n.O$ of $n.\psi$
  - Goal: $\mathcal{M} \models n.U \vee \neg n.O$:
    if $\mathcal{M} \models n.U$ return true for $\mathcal{G}_n$ (i.e., for $n.\psi$)
    if $\mathcal{M} \not\models n.O$ return false for $\mathcal{G}_n$ (i.e., for $n.\psi$)

Formulas $n.\psi$ have the form $\exists \bar{x}.F[\bar{z}, \bar{x}, \bar{p}]$

Model-based under-approximations in QSMA

Assume to have a solver for theory $\mathcal{T}$ (and model $\mathcal{M}_0$) offering a model-based under-approximation function MBU:

- Given formula $\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]$, where F is QF
  and extension $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z}\uplus\bar{p}$ such that $\mathcal{M}\models\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]$
- MBU$(F[\bar{z},\bar{x},\bar{p}],\bar{x},\mathcal{M})$ returns
  an under-approximation of $\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]$ that is true in $\mathcal{M}$:
  a QF formula $U[\bar{z},\bar{p}]$ such that
  - $\mathcal{T}\models U[\bar{z},\bar{p}]\Rightarrow(\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}])$ and
  - $\mathcal{M}\models U[\bar{z},\bar{p}]$

$U[\bar{z},\bar{p}]$: $\mathcal{T}$-interpolant between model and formula

Model-based over-approximations in QSMA

Assume to have a solver for theory $\mathcal{T}$ (and model $\mathcal{M}_0$) offering a model-based over-approximation function MBO:

- Given formula $\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]$, where F is QF and extension $\mathcal{M}$ of $\mathcal{M}_0$ to $\bar{z} \uplus \bar{p}$ such that $\mathcal{M} \not\models \exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]$
- MBO$(F[\bar{z},\bar{x},\bar{p}],\bar{x},\mathcal{M})$ returns an over-approximation of $\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]$ that is false in $\mathcal{M}$: a QF formula $O[\bar{z},\bar{p}]$ such that
  - $\mathcal{T} \models (\exists\bar{x}.F[\bar{z},\bar{x},\bar{p}]) \Rightarrow O[\bar{z},\bar{p}]$ and
  - $\mathcal{M} \not\models O[\bar{z},\bar{p}]$

$O[\bar{z},\bar{p}]$: reverse $\mathcal{T}$- interpolant between formula and model

Examples of MBU and MBO

- Examples of MBU:
  - Model-based projection
    [Komuravelli, Gurfinkel, Chaki: CAV 2014,
    FMSD journal 2016]
    [Bjørner, Janota: LPAR 2015 (short)]
  - Model generalization for LRA [Dutertre: SMT 2015]
  - Model generalization for NRA [Jovanović, Dutertre: CAV 2021]
- Examples of MBO:
  - Unsatisfiable core (purely Boolean assignment)
  - Model interpolation for NRA [Jovanović, Dutertre: CAV 2021]

Weakening and strengthening approximations in QSMA

QSMA-tree $\mathcal{G} = (\bar{z}, T)$ for formula φ
For all nodes n of T:

- Weaken $n.U$ so that $[\![n.U]\!]$ inflates by introducing a disjunction with an MBU
- Strengthen $n.O$ so that $[\![n.O]\!]$ deflates by introducing a conjunction with an MBO
- Inflate $[\![n.U]\!]$ and deflate $[\![n.O]\!]$ to zoom in on either a model of $n.\psi$ or its non-existence

Main function of the QSMA algorithm

@pre: $\mathcal{G} = (\bar{z}, T)$: QSMA-tree for φ with $FV(\varphi) = \bar{z}$
$\mathcal{M}$: extension of $\mathcal{M}_0$ to $\bar{z}$
@post: rv iff $\mathcal{M} \models \mathcal{G}$ (rv is "returned value")

```
function QSMA(M, T)
    for all nodes n in T do
        n.U ← ⊥
        n.O ← ⊤
    return SubtreeIsSolved(root(T), M)
```

$\bot$: under-approximation of all formulas and identity for disjunction
$\top$: over-approximation of all formulas and identity for conjunction

`subtreeIsSolved`: core function of QSMA I

Take node n and model $\mathcal{M}$ extending $\mathcal{M}_0$ to $Rigid(n)$
Determine whether $\mathcal{M} \models \mathcal{G}_n$ (same as $\mathcal{M} \models n.\psi$)

- @pre: $\mathcal{M}$: extension of $\mathcal{M}_0$ to $Rigid(n)$
  $\forall b \in T.\ [\![b.U]\!] \subseteq [\![b.\psi]\!] \subseteq [\![b.O]\!]$
- @post: $\forall b \in T.\ [\![b.U]\!] \subseteq [\![b.\psi]\!] \subseteq [\![b.O]\!]$
  $\mathcal{M} \models (n.U \vee \neg n.O)$
  rv iff $\mathcal{M} \models n.U$ iff $\mathcal{M} \models \mathcal{G}_n$
  $\neg rv$ iff $\mathcal{M} \models \neg n.O$ iff $\mathcal{M} \not\models \mathcal{G}_n$

subtreeIsSolved: core function of QSMA II

function $\textsc{subtreeIsSolved}(n, \mathcal{M})$
    if $\mathcal{M} \models n.U$ **then**
        return *true*
    else if $\mathcal{M} \models \neg n.O$ **then**
        return *false*
    while true **do**
        Loop body

Let $b.p$ denote the Boolean proxy variable labeling arc (n, b)

The loop in `subtreeIsSolved` I

while true **do**

  $L \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$

  $\mathcal{M}' \leftarrow \mathrm{SMA}(L, \mathcal{M})$

  if $\mathcal{M}' = \mathit{nil}$ **then**

    $n.O \leftarrow n.O \wedge \mathrm{MBO}(L, FV(L) \setminus \mathit{Rigid}(n), \mathcal{M})$

    return *false*

  else

    if $\textsc{solutionForallChildren}(n, \mathcal{M}')$ **then**

      $L' \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$

      $n.U \leftarrow n.U \vee \mathrm{MBU}(L', FV(L') \setminus \mathit{Rigid}(n), \mathcal{M})$

      return *true*

Why formula L?

$$L \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$$

Necessary condition for success: $\mathcal{M} \models \mathcal{G}_n$ implies $\mathcal{M}' \models L$

$\mathcal{M} \models \mathcal{G}_n$ means there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $Var(n)$ s.t.:

- $\mathcal{M}' \models n.F$
- If $\mathcal{M}'(b.p) = \text{true}$, $\mathcal{M}' \models b.\psi$
  the colored formula reduces to $b.O$ and $\mathcal{M}' \models b.O$ because $[\![b.\psi]\!] \subseteq [\![b.O]\!]$
- If $\mathcal{M}'(b.p) = \text{false}$, $\mathcal{M}' \not\models b.\psi$
  the colored formula reduces to $\neg b.U$ and $\mathcal{M}' \models \neg b.U$ because $\mathcal{M}' \not\models b.U$ as $[\![b.U]\!] \subseteq [\![b.\psi]\!]$

The loop in `subtreeIsSolved` II

while true **do**
  $L \leftarrow n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$
  $\mathcal{M}' \leftarrow \mathrm{SMA}(L, \mathcal{M})$
  if $\mathcal{M}' = \mathit{nil}$ **then**
    $n.O \leftarrow n.O \wedge \mathrm{MBO}(L, FV(L) \setminus \mathit{Rigid}(n), \mathcal{M})$
    return *false*
  else
    if $\textsc{solutionForallChildren}(n, \mathcal{M}')$ **then**
      $L' \leftarrow n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$
      $n.U \leftarrow n.U \vee \mathrm{MBU}(L', FV(L') \setminus \mathit{Rigid}(n), \mathcal{M})$
      return *true*

solutionForallChildren handles the recursion

function $\textsc{solutionForallChildren}(n, \mathcal{M})$
    for all children b of n **do**
        if $\mathcal{M}(b.p) \neq undef$ **then**
            if $\mathcal{M}(b.p) \neq \textsc{subtreeIsSolved}(b, \mathcal{M})$ **then**
                return *false*
    return *true*

- As soon as a child b (with assigned $b.p$) fails (expected truth value not met): return *false*
- Success for all children b (with assigned $b.p$): return *true*

The loop in `subtreeIsSolved` III

while true **do**
  $L \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$
  $\mathcal{M}' \leftarrow \mathrm{SMA}(L, \mathcal{M})$
  if $\mathcal{M}' = nil$ **then**
    $n.O \leftarrow n.O \wedge \mathrm{MBO}(L, FV(L) \setminus Rigid(n), \mathcal{M})$
    return *false*
  else
    if $\textsc{solutionForallChildren}(n, \mathcal{M}')$ **then**
      $L' \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$
      $n.U \leftarrow n.U \vee \mathrm{MBU}(L', FV(L') \setminus Rigid(n), \mathcal{M})$
      return *true*

Why formula L'?

$$L' \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$$

First: $\mathcal{M}' \models L'$

- $\mathcal{M}' \models n.F$ because $\mathcal{M}' \models L$
- If $\mathcal{M}'(b.p) = \mathrm{true}$: the colored formula reduces to $b.U$ and $\mathcal{M}' \models b.U$ since `subtreeIsSolved`$(b, \mathcal{M}')$ returned *true* (`solutionForallChildren` returned *true*)
- If $\mathcal{M}'(b.p) = \mathrm{false}$: the colored formula reduces to $\neg b.O$ and $\mathcal{M}' \models \neg b.O$ since `subtreeIsSolved`$(b, \mathcal{M}')$ returned *false* (`solutionForallChildren` returned *true*)

Why formula L'?

$L' \leftarrow n.F \wedge \bigwedge_{n \rightarrow b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$

Sufficient condition for success: $\mathcal{M}' \models L'$ implies $\mathcal{M} \models \mathcal{G}_n$

$\mathcal{M}' \models L'$ means that:

- $\mathcal{M}' \models n.F$
- If $\mathcal{M}'(b.p) = \text{true}$: the colored formula reduces to $b.U$ and $\mathcal{M}' \models b.U$ implies $\mathcal{M}' \models b.\psi$
- If $\mathcal{M}'(b.p) = \text{false}$: the colored formula reduces to $\neg b.O$ and $\mathcal{M}' \models \neg b.O$ implies $\mathcal{M}' \not\models b.\psi$

The loop in `subtreeIsSolved` IV

while true **do**
  $L \leftarrow n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow b.O) \wedge (\neg b.p \Rightarrow \neg b.U))$
  $\mathcal{M}' \leftarrow \mathrm{SMA}(L, \mathcal{M})$
  if $\mathcal{M}' = nil$ **then**
    $n.O \leftarrow n.O \wedge \mathrm{MBO}(L, FV(L) \setminus Rigid(n), \mathcal{M})$
    return *false*
  else
    if $\textsc{solutionForallChildren}(n, \mathcal{M}')$ **then**
      $L' \leftarrow n.F \wedge \bigwedge_{n \to b}((b.p \Rightarrow b.U) \wedge (\neg b.p \Rightarrow \neg b.O))$
      $n.U \leftarrow n.U \vee \mathrm{MBU}(L', FV(L') \setminus Rigid(n), \mathcal{M})$
      return *true*

When `solutionForallChildren` returns false

`solutionForallChildren` found a child b of n such that

- Either $\mathcal{M}'(b.p) =$ true but
  $\texttt{subtreeIsSolved}(b, \mathcal{M}')$ returned *false*:
  $\texttt{subtreeIsSolved}(b, \mathcal{M}')$ updated $b.O$
- Or $\mathcal{M}'(b.p) =$ false but
  $\texttt{subtreeIsSolved}(b, \mathcal{M}')$ returned *true*:
  $\texttt{subtreeIsSolved}(b, \mathcal{M}')$ updated $b.U$

Either way the state has changed: variable L will get a new formula and SMA will not produce the same assignment

QSMA is partially correct

Thm: `subtreeIsSolved` is partially correct: if the preconditions hold and it halts, the postconditions hold

And termination?

- LRA: given $\exists x.F[\bar{z}, x]$
  under-approximation: $F[\bar{z}, \tilde{q}]$
  $\tilde{q}$: constant symbol for rational number q
- Consider an MBU such that
  $\mathrm{MBU}(F[\bar{z}, x], x, \mathcal{M}) = F[\bar{z}, \tilde{q}]$ and $\mathcal{M} \models F[\bar{z}, \tilde{q}]$
- Infinite enumeration of rational constants and infinite series of under-approximations $(\bigvee_{i=1}^{n} F[\bar{z}, x]\{x \leftarrow \tilde{q}_i\})_{n \in \mathbb{N}}$

MBU and MBO have finite basis: QSMA is totally correct

For all QF formulas $F[\bar{z}, \bar{x}, \bar{p}]$ and tuples $\bar{x}$ the sets

$$\{\mathrm{MBU}(F[\bar{z}, \bar{x}, \bar{p}], \bar{x}, \mathcal{M}) \mid \mathcal{M} : \text{extension of } \mathcal{M}_0 \text{ to } \bar{z} \text{ such that } \mathcal{M} \models \exists \bar{x}. F[\bar{z}, \bar{x}, \bar{p}]\}$$

$$\{\mathrm{MBO}(F[\bar{z}, \bar{x}, \bar{p}], \bar{x}, \mathcal{M}) \mid \mathcal{M} : \text{extension of } \mathcal{M}_0 \text{ to } \bar{z} \text{ such that } \mathcal{M} \not\models \exists \bar{x}. F[\bar{z}, \bar{x}, \bar{p}]\}$$

are finite

Thm: If MBU and MBO have finite basis, whenever the preconditions are satisfied `subtreeIsSolved` halts

Example I

- $\forall x.((\exists y.(x \simeq 2{\cdot}y)) \Rightarrow (\exists z.(3{\cdot}x \simeq 2{\cdot}z)))$
- $\neg(\exists x.((\exists y.(x \simeq 2{\cdot}y)) \wedge (\forall z.(3{\cdot}x \not\simeq 2{\cdot}z))))$
- $\neg(\exists x.((\exists y.(x \simeq 2{\cdot}y)) \wedge (\neg(\exists z.(3{\cdot}x \simeq 2{\cdot}z)))))$
- $\varphi = \exists x.((\exists y.(x \simeq 2{\cdot}y)) \wedge (\neg(\exists z.(3{\cdot}x \simeq 2{\cdot}z))))$
- The original formula is true in LRA iff φ is false in LRA
- In this example the original formula is true in LRA
- $\varphi = \exists x.(p_1 \wedge \neg p_2)$ where
  $p_1 = \exists y.(x \simeq 2{\cdot}y) \qquad p_2 = \exists z.(3{\cdot}x \simeq 2{\cdot}z)$

Example II

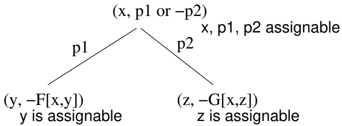
$$\varphi = \exists x.(p_1 \wedge \neg p_2) \qquad p_1 = \exists y.(x \simeq 2{\cdot}y) \qquad p_2 = \exists z.(3{\cdot}x \simeq 2{\cdot}z)$$

- Apply `subtreeIsSolved` to the root: $L \leftarrow p_1 \wedge \neg p_2$ as $b_1.U = b_2.U = \bot$ and $b_1.O = b_2.O = \top$
- Say SMA produces $x{\leftarrow}1$, $p_1{\leftarrow}$true, $p_2{\leftarrow}$false
- Recurse on b_1: $L \leftarrow x \simeq 2{\cdot}y$ (no children)
- SMA produces $y{\leftarrow}\frac{1}{2}$: return *true*
- Recurse on b_2: $L \leftarrow 3{\cdot}x \simeq 2{\cdot}z$ (no children)
- SMA produces $z{\leftarrow}\frac{3}{2}$: return *true*
- But $p_2{\leftarrow}$false, hence return *false*

OptiQSMA: Reconsider an earlier example

$$\varphi = \exists x.(p_1 \vee \neg p_2) \qquad p_1 = \exists y.\neg F[x, y] \qquad p_2 = \exists z.\neg G[x, z]$$

- Apply `subtreeIsSolved` to the root r: $L \leftarrow p_1 \vee \neg p_2$
- If SMA yields $p_1 \leftarrow$true:
- Recurse on b_1: $L \leftarrow \neg F[x, y]$
- If SMA yields value for y s.t. $\neg F[x, y]$, return *true*
- If SMA yields $p_2 \leftarrow$false:
- Recurse on b_2: $L \leftarrow \neg G[x, z]$
- If SMA returns nil, return *false*



OptiQSMA: from the example to the general idea

- Pass $(p_1 \vee \neg p_2) \wedge (p_1 \Rightarrow \neg F[x, y])$ to SMA
  (in place of $p_1 \vee \neg p_2$)
- If SMA assigns true to p_1, it also assigns to x and y values that satisfy $\neg F[x, y]$
  $\exists y. \neg F[x, y]$ is found true without recursion
- If SMA assigns false to p_2, still need to recurse to check that $\exists z. \neg G[x, z]$ is false
- Fewer recursive calls to *subtreeIsSolved* by letting the underlying solver SMA look ahead

OptiQSMA: the look-ahead formula

- QSMA-tree $\mathcal{G} = (\bar{z}, T)$
- For all nodes n of T the look-ahead formula of n is

$$LF(n) = n.F \wedge \bigwedge_{n \rightarrow b}(b.p \Rightarrow LF(b))$$

- If $b.p$ is true look ahead at $b.F$ (the child's formula)

OptiQSMA: FAN and NAN nodes

- No alternation nodes:
  - NAN$(n, \mathcal{M})$: descendants b of n via a path where all proxies and $b.p$ are assigned true by $\mathcal{M}$
  - Handled together in one shot without recursion
- First alternation nodes:
  - FAN$(n, \mathcal{M})$: descendants b of n via a path where all proxies are assigned true but $b.p$ is assigned false by $\mathcal{M}$
  - Recursion needed

OptiQSMA: satisfaction with look-ahead

QSMA-tree $\mathcal{G} = (\bar{z}, T)$ with root r

- $\mathcal{M}$: extension of $\mathcal{M}_0$ to $Rigid(r) = \bar{z}$
- $\mathcal{M} \models_{la} \mathcal{G}$ if there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ to $FV(LF(r))$ such that
  1. $\mathcal{M}' \models LF(r)$
  2. For all nodes $b \in \text{FAN}(r, \mathcal{M}')$: $\mathcal{M}' \not\models_{la} \mathcal{G}_b$

For node $b \in \text{FAN}(r, \mathcal{M}')$: $\mathcal{M}'(b.p) = \text{false}$:
try to show $b.\psi$ false

Thm: $\mathcal{G}$ is the QSMA-tree for formula φ: $\mathcal{M} \models \mathcal{G}$ iff $\mathcal{M} \models_{la} \mathcal{G}$

Implementation and experimental results

- OptiQSMA is implemented in YicesQS (S. Graham-Lengrand) built on top of Yices 2 (B. Dutertre, D. Jovanović)
- YicesQS entered the Single Query Track (Main Track) of SMT-COMP in 2022 and 2023
- 2022: YicesQS won SAT performance and 24s performance columns for Arith (LRA, LIA, NRA, NIA);
  only solver to solve all LRA benchmarks;
  ranked 2nd for Largest Contribution Award
- 2023: YicesQS won SAT performance and 24s performance columns for Arith and was among the first three solvers in all columns for Arith

Current and future work

- Integration of QSMA in the CDSAT framework for conflict-driven reasoning in unions of theories:
  1. SMA as a CDSAT solver
  2. QSMA as a CDSAT module
  3. Formalize QSMA as transition system and unwrap it into CDSAT
- Improvements to YicesQS, e.g.:
  integer reasoning, bitvector reasoning

Thanks

- MPB, Stéphane Graham-Lengrand, and Christophe Vauthier. QSMA: a new algorithm for quantified satisfiability modulo theory and assignment. Proc. 29th Int. Conf. on Automated Deduction (CADE), LNAI 14132, 78-95, Springer, Aug. 2023.
- MPB. Reasoning about quantifiers in SMT: the QSMA algorithm (Abstract). Proc. 23rd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD), 1–1, TU Wien Academic Press, Oct. 2023.

Thank you!