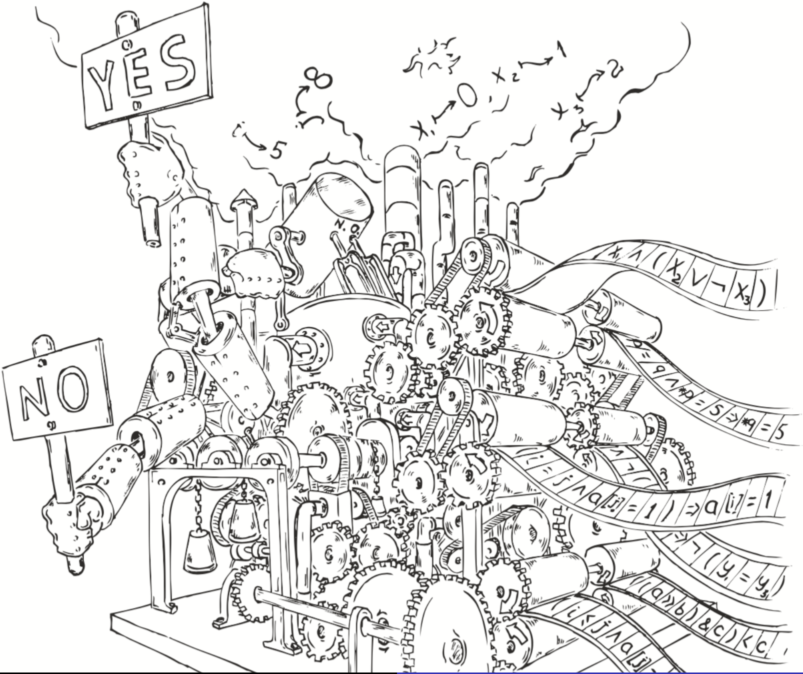

YES
NO

SAT/SMT solvers
8. Quantified Formulas

Roman Kholin

Lomonosov Moscow State University

Moscow, 2023

Definitions

$$\forall x.\ \varphi \iff \neg\exists x.\ \neg\varphi$$

$$\underbrace{\forall x.\ ((x<0) \land \exists y.\ \overbrace{(y > x \land (y \geq 0 \lor \exists x.\ \underbrace{(y = x+1)}_{\text{scope of } \exists x}))}^{\text{scope of } \exists y})}_{\text{scope of } \forall x}\ .$$

- A variable is called free in a given formula if at least one of its occurrences is not bound by any quantifier
- A formula Q is called a sentence (or closed) if none of its variables are free

Syntax

QBF:

$$\begin{aligned} formula &: formula \wedge formula \mid \neg formula \mid (formula) \mid \\ &\quad identifier \mid \exists\, identifier.\ formula \end{aligned}$$

Complexity - PSPACE

QDLA:

$$\begin{aligned} formula &: formula \wedge formula \mid \neg formula \mid (formula) \mid \\ &\quad predicate \mid \forall\, identifier.\ formula \\ predicate &: \Sigma_i a_i x_i \leq c \end{aligned}$$

Definitions

Prenex normal form

- A formula is said to be in prenex normal form (PNF) if it is in the form $Q[n]V[n] \ldots Q[1]V[1]. < quantifier - freeformula >$, where $Q[i]$ - quantor, $V[i]$ - variable
- For every quantified formula Q there exists a formula Q' in prenex normal form such that Q is valid if and only if Q' is valid

Algorithm 9.2.1: PRENEX

Input: A quantified formula
Output: A formula in prenex normal form

1. Eliminate Boolean connectives other than $\vee$, $\wedge$, and $\neg$.
2. Push negations to the right across all quantifiers, using De Morgan's rules (see Sect. 1.3) and (9.1).
3. If there are name conflicts across scopes, solve by renaming: give each variable in each scope a unique name.
4. Move quantifiers out by using equivalences such as

$$
\begin{aligned}
\phi_1 \wedge Qx.\ \phi_2(x) &\iff Qx.\ (\phi_1 \wedge \phi_2(x))\ , \\
\phi_1 \vee Qx.\ \phi_2(x) &\iff Qx.\ (\phi_1 \vee \phi_2(x))\ , \\
Q_1 y.\ \phi_1(y) \wedge Q_2 x.\ \phi_2(x) &\iff Q_1 y.\ Q_2 x.\ (\phi_1(y) \wedge \phi_2(x))\ , \\
Q_1 y.\ \phi_1(y) \vee Q_2 x.\ \phi_2(x) &\iff Q_1 y.\ Q_2 x.\ (\phi_1(y) \vee \phi_2(x))\ ,
\end{aligned}
$$

where $Q, Q_1, Q_2 \in \{\forall, \exists\}$ are quantifiers, $x \notin var(\phi_1)$, and $y \notin var(\phi_2)$.

Example

$$\neg\exists x.\neg(\exists y.((y \implies x) \wedge (\neg x \vee y)) \wedge \neg\forall y.((y \wedge x) \vee (\neg x \wedge \neg y)))$$

Example

$$\neg\exists x.\neg(\exists y.((y \implies x) \wedge (\neg x \vee y)) \wedge \neg\forall y.((y \wedge x) \vee (\neg x \wedge \neg y)))$$
$$\forall x.(\exists y.((\neg y \vee x) \wedge (\neg x \wedge y)) \wedge \exists y.((\neg y \vee \neg x) \wedge (x \vee y)))$$

Example

$$\neg\exists x.\neg(\exists y.((y \implies x) \wedge (\neg x \vee y)) \wedge \neg\forall y.((y \wedge x) \vee (\neg x \wedge \neg y)))$$
$$\forall x.(\exists y.((\neg y \vee x) \wedge (\neg x \wedge y)) \wedge \exists y.((\neg y \vee \neg x) \wedge (x \vee y)))$$
$$\forall x.(\exists y_1.((\neg y_1 \vee x) \wedge (\neg x \wedge y_1)) \wedge \exists y_2.((\neg y_2 \vee \neg x) \wedge (x \vee y_2)))$$

Example

$$\neg\exists x.\neg(\exists y.((y \implies x) \wedge (\neg x \vee y)) \wedge \neg\forall y.((y \wedge x) \vee (\neg x \wedge \neg y)))$$
$$\forall x.(\exists y.((\neg y \vee x) \wedge (\neg x \wedge y)) \wedge \exists y.((\neg y \vee \neg x) \wedge (x \vee y)))$$
$$\forall x.(\exists y_1.((\neg y_1 \vee x) \wedge (\neg x \wedge y_1)) \wedge \exists y_2.((\neg y_2 \vee \neg x) \wedge (x \vee y_2)))$$
$$\forall x.\exists y_1.\exists y_2.(\neg y_1 \vee x) \wedge (\neg x \vee y_1) \wedge (\neg y_2 \vee \neg x) \wedge (x \vee y_2).$$

Projection

Projection of $Q[n]V[n]\ldots Q[2]V[2].\exists x.\phi$
is $Q[n]V[n]\ldots Q[2]V[2].\phi$

Algorithm 9.2.2: QUANTIFIER-ELIMINATION

Input: A sentence $Q[n]V[n]\dots Q[1]V[1].\ \phi$, where ϕ is quantifier-free

Output: A (quantifier-free) formula over constants ϕ', which is valid if and only if ϕ is valid

1. $\phi' := \phi$;
2. **for** $i := 1, \dots, n$ **do**
3.   **if** $Q[i] = \exists$ **then**
4. $\phi' := \ \text{PROJECT}(\ \phi', V[i])$;
5.   **else**
6. $\phi' := \neg\text{PROJECT}(\neg\phi', V[i])$;
7. Return ϕ';

Quantifier elimination for Quantified Boolean Formulas

$\exists y.\exists z.\forall x.(y \vee x) \wedge (z \vee \neg x) \wedge (y \vee \neg z \vee \neg x) \wedge (\neg y \vee z)$
$\exists y.\exists z.(y) \wedge (z) \wedge (y \vee \neg z \vee) \wedge (\neg y \vee z)$

Quantifier elimination for Quantified Boolean Formulas

$\exists y.\exists z.\forall x.(y \vee x) \wedge (z \vee \neg x) \wedge (y \vee \neg z \vee \neg x) \wedge (\neg y \vee z)$

$\exists y.\exists z.(y) \wedge (z) \wedge (y \vee \neg z \vee) \wedge (\neg y \vee z)$

$$\exists y.\ \exists x.\ x \wedge \neg x \wedge y \quad = \text{FALSE}\ ,$$
$$\exists y.\ \exists x.\ x \wedge y = \exists y.\ y = \text{TRUE}\ .$$

$$\exists x.\ \bigvee_i \bigwedge_j l_{ij} \iff \bigvee_i \exists x.\ \bigwedge_j l_{ij}$$

Quantifier elimination for Quantified Boolean Formulas

$\exists y.\exists z.\forall x.(y \vee x) \wedge (z \vee \neg x) \wedge (y \vee \neg z \vee \neg x) \wedge (\neg y \vee z)$

$\exists y.\exists z.(y) \wedge (z) \wedge (y \vee \neg z \vee) \wedge (\neg y \vee z)$

$$\exists y.\ \exists x.\ x \wedge \neg x \wedge y \quad = \text{FALSE}\ ,$$

$$\exists y.\ \exists x.\ x \wedge y = \exists y.\ y = \text{TRUE}\ .$$

$$\exists x.\ \bigvee_i \bigwedge_j l_{ij} \Longleftrightarrow \bigvee_i \exists x.\ \bigwedge_j l_{ij}$$

$\exists y.\exists z.\exists x.(y \vee x) \wedge (z \vee \neg x) \wedge (y \vee \neg z \vee \neg x) \wedge (\neg y \vee z)$

$\exists y.\exists z.(y \vee z) \wedge (y \vee \neg z) \wedge (\neg y \vee z)$

Quantifier elimination for Quantified Boolean Formulas

$\exists y.\exists z.\forall x.(y \vee x) \wedge (z \vee \neg x) \wedge (y \vee \neg z \vee \neg x) \wedge (\neg y \vee z)$

$\exists y.\exists z.(y) \wedge (z) \wedge (y \vee \neg z \vee) \wedge (\neg y \vee z)$

$$\exists y.\ \exists x.\ x \wedge \neg x \wedge y \quad = \text{FALSE}\ ,$$

$$\exists y.\ \exists x.\ x \wedge y = \exists y.\ y = \text{TRUE}\ .$$

$$\exists x.\ \bigvee_i \bigwedge_j l_{ij} \iff \bigvee_i \exists x.\ \bigwedge_j l_{ij}$$

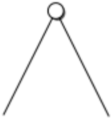
$\exists y.\exists z.\exists x.(y \vee x) \wedge (z \vee \neg x) \wedge (y \vee \neg z \vee \neg x) \wedge (\neg y \vee z)$

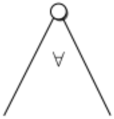
$\exists y.\exists z.(y \vee z) \wedge (y \vee \neg z) \wedge (\neg y \vee z)$

$$\exists x.\ \varphi = \varphi|_{x=0} \vee \varphi|_{x=1}$$

$$\forall x.\ \varphi = \varphi|_{x=0} \wedge \varphi|_{x=1}$$

Search-Based Algorithms for Quantified Boolean Formulas





Algorithm 9.3.1: SEARCH-BASED-DECISION-OF-QBF

Input: A QBF $\mathcal{Q}$ in PNF $Q[n]V[n] \dots\ Q[1]V[1].\ \phi$, where ϕ is in CNF

Output: "Valid" if $\mathcal{Q}$ is valid, and "Not valid" otherwise

function MAIN(QBF formula $\mathcal{Q}$)
    if QBF($\mathcal{Q}, \emptyset, n$) **then return** "Valid";
    else return "Not valid";

function QBF($\mathcal{Q}$, assignment set $\hat{v}$, *level* $\in \mathbb{N}_0$)
    if ($\phi|_{\hat{v}}$ simplifies to FALSE) **then return** FALSE;
    if ($level = 0$) **then return** TRUE;
    if ($Q[level] = \forall$) **then**
        return $\begin{pmatrix} \mathrm{QBF}(\mathcal{Q}, \hat{v} \cup \neg V[level], level - 1) \wedge \\ \mathrm{QBF}(\mathcal{Q}, \hat{v} \cup \ \ V[level], level - 1) \end{pmatrix}$;
    else
        return $\begin{pmatrix} \mathrm{QBF}(\mathcal{Q}, \hat{v} \cup \neg V[level], level - 1) \vee \\ \mathrm{QBF}(\mathcal{Q}, \hat{v} \cup \ \ V[level], level - 1) \end{pmatrix}$;

Skolemization

Skolem normal form

A formula is in Skolem normal form if it is in prenex normal form and has only universal quantifiers.

Skolemization

Let $\psi = \exists x.\ P$ be a subformula in φ. Let $y_1, \ldots, y_n$ be universally quantified variables such that ψ is in their scope.

1. Remove the quantifier $\exists x$ from ψ
2. Replace occurrences of x in P with $f_x(y_1, \ldots, y_n)$, where f_x is a new function symbol. It is sufficient to include those y variables that are actually used in P. If $n = 0$, then replace x with a new constant c_x

$$\forall y_1.\forall y_2.f(y_1, y_2) \wedge \exists x.(f(x, y_2) \wedge x < 0)$$

Skolemization

Skolem normal form

A formula is in Skolem normal form if it is in prenex normal form and has only universal quantifiers.

Skolemization

Let $\psi = \exists x.\ P$ be a subformula in φ. Let $y_1, \ldots, y_n$ be universally quantified variables such that ψ is in their scope.

1. Remove the quantifier $\exists x$ from ψ
2. Replace occurrences of x in P with $f_x(y_1, \ldots, y_n)$, where f_x is a new function symbol. It is sufficient to include those y variables that are actually used in P. If $n = 0$, then replace x with a new constant c_x

$\forall y_1.\forall y_2.f(y_1, y_2) \wedge \exists x.(f(x, y_2) \wedge x < 0)$
$\forall y_1.\forall y_2.f(y_1, y_2) \wedge (f(f_x(y_1, y_2), y_2) \wedge f_x(y_1, y_2) < 0)$

Instantiation

A typical scenario: to prove the validity of a ground formula G based on sentences that represent axioms.
To prove G we can try to *instantiate* the universally quantified variables in order to reach a contradiction with $\neg G$.

Instantiation

A typical scenario: to prove the validity of a ground formula G based on sentences that represent axioms.

To prove G we can try to *instantiate* the universally quantified variables in order to reach a contradiction with $\neg G$.

$$\overbrace{f(h(a), b) = f(b, h(a))}^{G}$$

$$\forall x.\ \forall y.\ f(x, y) = f(y, x)$$

Instantiation

A typical scenario: to prove the validity of a ground formula G based on sentences that represent axioms.
To prove G we can try to *instantiate* the universally quantified variables in order to reach a contradiction with $\neg G$.

$$\overbrace{f(h(a), b) = f(b, h(a))}^{G}$$

$$\forall x.\ \forall y.\ f(x, y) = f(y, x)$$

$$(\forall x.\ \forall y.\ f(x, y) = f(y, x)) \land f(h(a), b) \neq f(b, h(a))$$

Instantiation

A typical scenario: to prove the validity of a ground formula G based on sentences that represent axioms.
To prove G we can try to *instantiate* the universally quantified variables in order to reach a contradiction with $\neg G$.

$$\overbrace{f(h(a), b) = f(b, h(a))}^{G}$$

$$\forall x.\ \forall y.\ f(x, y) = f(y, x)$$

$$(\forall x.\ \forall y.\ f(x, y) = f(y, x)) \wedge f(h(a), b) \neq f(b, h(a))$$

$$f(h(a), b) = f(b, h(a)) \wedge f(h(a), b) \neq f(b, h(a))$$

Simple strategy

Let $(\forall \overline{x}.\psi) \wedge G$ - formula that we attempt to prove to be unsatisfiable
Triggers - subterms in ψ that contain references to all the variables in $\overline{x}$

- For each quantified formula of the form $(\forall \overline{x}.\psi)$, identify all triggers
- Try to match each trigger tr to an existing ground term *gr* in G.
- Given a substitution $\overline{s}$, assign $G := G \wedge \psi[\overline{x} \leftarrow \overline{s}]$ and check the satisfiability of G

Simple strategy

Let $(\forall \overline{x}.\psi) \wedge G$ - formula that we attempt to prove to be unsatisfiable
Triggers - subterms in ψ that contain references to all the variables in $\overline{x}$

- For each quantified formula of the form $(\forall \overline{x}.\psi)$, identify all triggers
- Try to match each trigger tr to an existing ground term *gr* in G.
- Given a substitution $\overline{s}$, assign $G := G \wedge \psi[\overline{x} \leftarrow \overline{s}]$ and check the satisfiability of G

$G := (b = c \implies f(h(a), g(c)) = f(g(b), h(a)))$
$\forall x.\forall y.f(x, y) = f(y, x)$

Let $(\forall \overline{x}.\psi) \wedge G$ - formula that we attempt to prove to be unsatisfiable

Triggers - subterms in ψ that contain references to all the variables in $\overline{x}$

- For each quantified formula of the form $(\forall \overline{x}.\psi)$, identify all triggers
- Try to match each trigger tr to an existing ground term *gr* in G.
- Given a substitution $\overline{s}$, assign $G := G \wedge \psi[\overline{x} \leftarrow \overline{s}]$ and check the satisfiability of G

$G := (b = c \implies f(h(a), g(c)) = f(g(b), h(a)))$
$\forall x.\forall y.f(x, y) = f(y, x)$
$(\forall x.\forall y.f(x, y) = f(y, x)) \wedge b = c \wedge f(h(a), g(c)) \neq f(g(b), h(a))$

Simple strategy

Let $(\forall \overline{x}.\psi) \wedge G$ - formula that we attempt to prove to be unsatisfiable

Triggers - subterms in ψ that contain references to all the variables in $\overline{x}$

- For each quantified formula of the form $(\forall \overline{x}.\psi)$, identify all triggers
- Try to match each trigger tr to an existing ground term *gr* in G.
- Given a substitution $\overline{s}$, assign $G := G \wedge \psi[\overline{x} \leftarrow \overline{s}]$ and check the satisfiability of G

$G := (b = c \implies f(h(a), g(c)) = f(g(b), h(a)))$
$\forall x. \forall y. f(x, y) = f(y, x)$
$(\forall x. \forall y. f(x, y) = f(y, x)) \wedge b = c \wedge f(h(a), g(c)) \neq f(g(b), h(a))$
$(b = c \wedge f(h(a), g(c)) \neq f(g(b), h(a))) \wedge f(h(a), g(c)) = f(g(c), h(a)) \wedge f(g(b), h(a)) = f(h(a), g(b))$

Input: Trigger tr, term gr, current substitution set sub

Output: Substitution set sub such that for each $\alpha \in sub$, $E \models \alpha(tr) = gr$.

function MATCH(tr, gr, sub)
  if tr is a variable x **then**
    return
      $\{\alpha \cup \{x \mapsto gr\} \mid \alpha \in sub, x \notin dom(\alpha)\} \cup$
      $\{\alpha \mid \alpha \in sub, find(\alpha(x)) = find(gr)\}$
  if tr is a constant c **then**
    if $c \in class(gr)$ **then return** sub
    else return $\emptyset$
  if tr is of the form $f(p_1, \ldots, p_n)$ **then return**

$$\bigcup_{f(gr_1,\ldots,gr_n)\in class(gr)} \begin{array}{l} \text{MATCH}(p_n, gr_n, \\ \quad \text{MATCH}(p_{n-1}, gr_{n-1}, \\ \qquad \ddots \\ \qquad\quad \text{MATCH}(p_1, gr_1, sub)\ldots)) \end{array}$$

E-matching

$$(\forall x.f(x) = x) \wedge (\forall y_1.\forall y_2.g(g(y1, y2), y2) = y2) \wedge g(f(g(a, b)), b) \neq b$$

E-matching

$(\forall x.f(x) = x) \wedge (\forall y_1.\forall y_2.g(g(y1, y2), y2) = y2) \wedge g(f(g(a, b)), b) \neq b$

$\text{match}(f(x), f(g(a, b)), \emptyset)$

E-matching

$(\forall x.f(x) = x) \wedge (\forall y_1.\forall y_2.g(g(y1, y2), y2) = y2) \wedge g(f(g(a, b)), b) \neq b$
$\text{match}(f(x), f(g(a, b)), \emptyset) = \text{match}(x, g(a, b), \emptyset)$

E-matching

$(\forall x. f(x) = x) \wedge (\forall y_1.\forall y_2. g(g(y1, y2), y2) = y2) \wedge g(f(g(a, b)), b) \neq b$

$\text{match}(f(x), f(g(a, b)), \emptyset) =$

$\text{match}(x, g(a, b), \emptyset) =$

$\{x \to g(a, b)\}$

E-matching

$(\forall x.f(x) = x) \wedge (\forall y_1.\forall y_2.g(g(y1, y2), y2) =$
$y2) \wedge g(f(g(a, b)), b) \neq b$
$\text{match}(f(x), f(g(a, b), b), \emptyset) =$
$\text{match}(x, g(a, b), \emptyset) =$
$\{x \rightarrow g(a, b)\}$
$f(g(a, b)) = g(a, b)$

E-matching

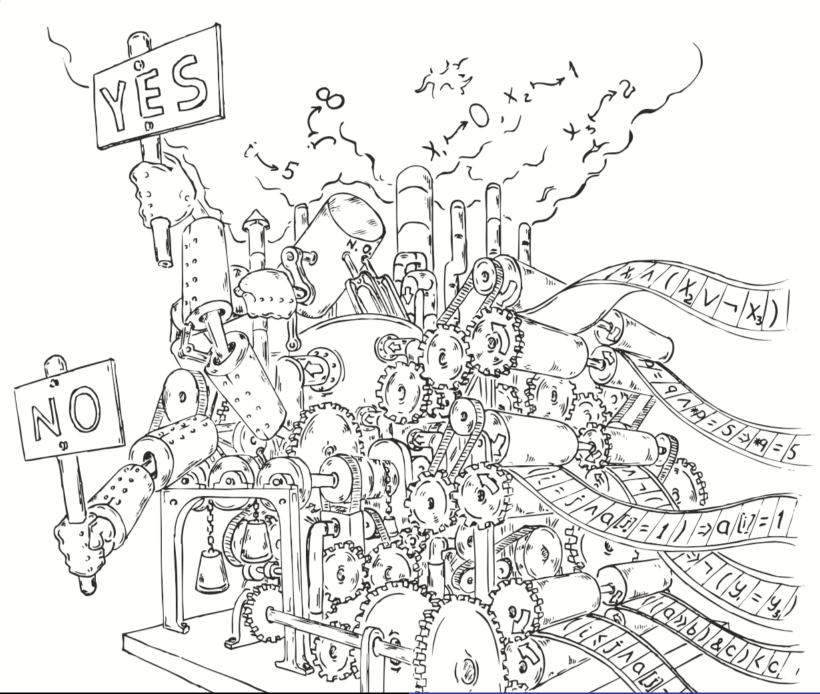
$(\forall x.f(x) = x) \wedge (\forall y_1.\forall y_2.g(g(y1, y2), y2) =$
$y2) \wedge g(f(g(a, b)), b) \neq b$
$\text{match}(g(g(y1, y2), y2), g(f(g(a, b)), b), \emptyset) =$
$\text{match}(y_2, b, match(g(y_1, y_2), f(g(a, b)), \emptyset)) =$
$\text{match}(y_2, b, match(g(y_1, y_2), g(a, b), \emptyset)) =$
$\text{match}(y_2, b, match(y_2, b, match(y_1, a, \emptyset))) =$
$\text{match}(y_2, b, match(y_2, b, \{y_1 \to a\})) =$
$\text{match}(y_2, b, \{y_1 \to a, y_2 \to b\}) = \{y_1 \to a, y_2 \to b\}$
$g(g(a, b), b) = b$

E-matching

$(\forall x.f(x) = x) \wedge (\forall y_1.\forall y_2.g(g(y1, y2), y2) =$
$y2) \wedge g(f(g(a, b)), b) \neq b$
$(f(g(a, b)) = g(a, b)) \wedge (g(g(a, b), b) = b) \wedge (g(f(g(a, b)), b) \neq b)$

E-matching

$$(\forall x. f(2x - x) < x) \wedge (f(a) \geq a)$$


YES
NO