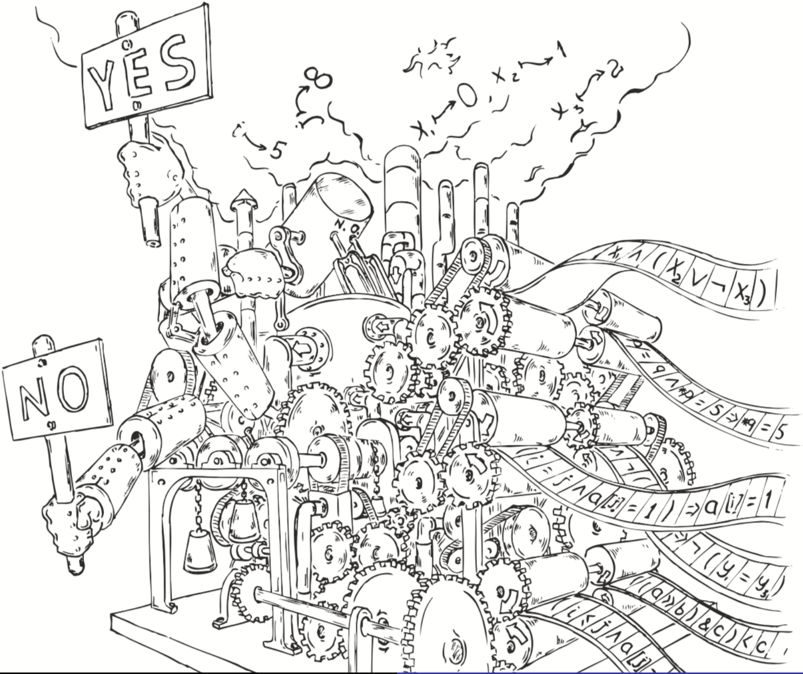

YES
NO

SAT/SMT solvers
8. Quantified Formulas

Roman Kholin

Lalambda

Tbilisi, 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 $Q[n]V[n]\dots Q[1]V[1].< quantifier - freeformula >$, 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

$$\mathcal{Q} := \neg\exists x.\ \neg(\exists y.\ ((y \implies x) \wedge (\neg x \vee y)) \wedge \neg\forall y.\ ((y \wedge x) \vee (\neg x \wedge \neg y)))$$

Projection

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

Example

$$\mathcal{Q} := \neg\exists x.\, \neg(\exists y.\, ((y \implies x) \wedge (\neg x \vee y)) \wedge \neg\forall y.\, ((y \wedge x) \vee (\neg x \wedge \neg y)))$$

Algorithm 9.2.2: QUANTIFIER-ELIMINATION

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

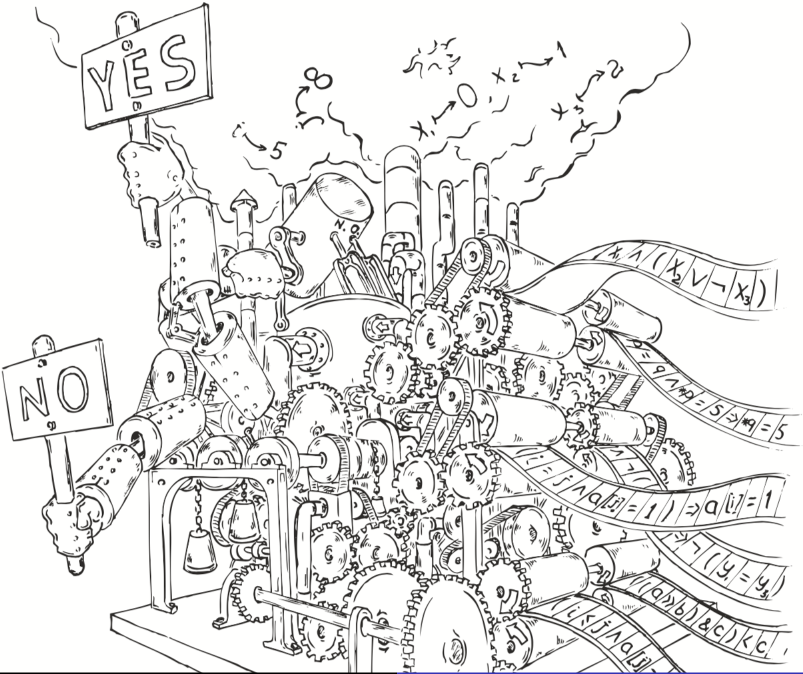
Output: A (quantifier-free) formula over constants ϕ', which is valid if and only if ϕ is valid

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

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

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


YES
NO