Dependencies in Quantified Boolean Formulae

Mgr. Tomáš Peitl

Supervisor: Prof. Stefan Szeider, PhD

January 4, 2016

Overview

QBF

- What is a Quantified Boolean Formula?

Overview

QBF

- What is a Quantified Boolean Formula?
- What can a Quantified Boolean Formula do?

Overview

QBF

- What is a Quantified Boolean Formula?
- What can a Quantified Boolean Formula do?
- How can we help a Quantified Boolean Formula do it?

What is a Quantified Boolean Formula?

$$(x \vee y) \wedge (\neg x \vee \neg y)$$

What is a Quantified Boolean Formula?

$$\exists x \exists y \, (x \vee y) \wedge (\neg x \vee \neg y)$$

What is a Quantified Boolean Formula?

$$\exists x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$$

$$\forall x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$$

What is a Quantified Boolean Formula?

$\exists x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\forall x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\exists x \forall y \; (x \vee y) \wedge (\neg x \vee \neg y)$

What is a Quantified Boolean Formula?

$\exists x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\forall x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\exists x \forall y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\forall x \exists y \; \{x, y\}, \{\overline{x}, \overline{y}\}$

What is a Quantified Boolean Formula?

$\exists x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\forall x \exists y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\exists x \forall y \; (x \vee y) \wedge (\neg x \vee \neg y)$

$\forall x \exists y \; \{x, y\}, \{\overline{x}, \overline{y}\}$

Conjunctive Normal Form (CNF) is a conjunction of **clauses** which are disjunctions of **literals** which are either **variables** or their **negations**.

QBFs and games

QBFs are good at 2-player games

$$\forall x \exists y \, \{x, y\}, \{\overline{x}, \overline{y}\}$$

$\exists$ vs. $\forall$

QBFs and complexity

- The problem of satisfiability of QBFs (QSAT) is PSPACE-complete.
- The problem of satisfiability of propositional formulae (SAT) is NP-complete.

Applications

- Model Checking
- Verification
- Planning
- Ontology reasoning
- Logic synthesis
- Games

Proofs, (counter)models and strategies

$\exists x \forall y \; \{x, y\}, \{\overline{x}, \overline{y}\}$

Proofs, (counter)models and strategies

$$\exists x \forall y \; \{x, y\}, \{\overline{x}, \overline{y}\}$$

$$x = 0 \Longrightarrow y = 0$$

Proofs, (counter)models and strategies

$$\exists x \forall y \; \{x, y\}, \{\overline{x}, \overline{y}\}$$

$$x = 0 \Longrightarrow y = 0$$

$$x = 1 \Longrightarrow y = 1$$

Proofs, (counter)models and strategies

$$\exists x \forall y \; \{x, y\}, \{\overline{x}, \overline{y}\}$$

$$x = 0 \Longrightarrow y = 0$$

$$x = 1 \Longrightarrow y = 1$$

$$y = x$$

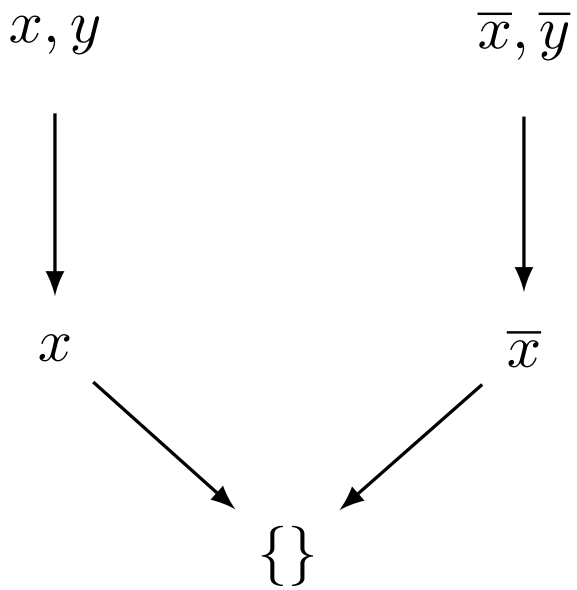
Proofs, (counter)models and strategies

$$\exists x \forall y \; \{x, y\}, \{\overline{x}, \overline{y}\}$$

$$x = 0 \Longrightarrow y = 0$$

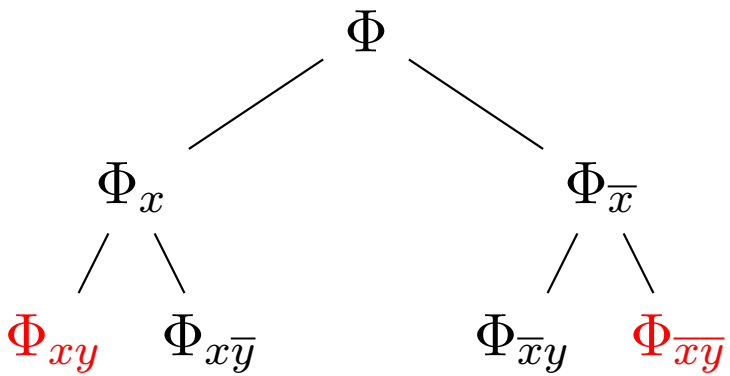
$$x = 1 \Longrightarrow y = 1$$

$$y = x$$

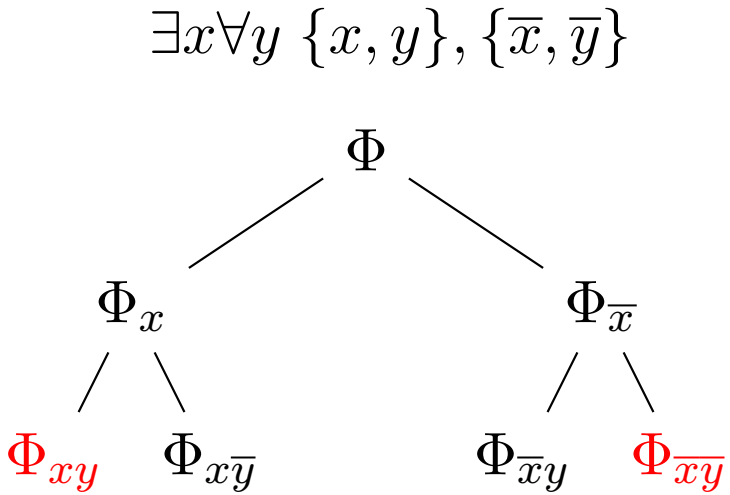


(Q)DPLL and Clause Learning

$$\exists x \forall y \; \{x, y\}, \{\overline{x}, \overline{y}\}$$



(Q)DPLL and Clause Learning



Simple inference can speed up the search. Moreover, inferred assignments don't cause conflicts, so conflicts can be used to learn new constraints.

Selection heuristics

- Much of the success of the current state-of-the-art SAT solvers is due to clever heuristics for selecting branching variables
- Using the same heuristics for QBF runs into the limits of the quantifier prefix - the dependencies between variables

$$\exists x \forall y \{x, y\}, \{\overline{x}, \overline{y}\}$$

$$y = x$$

Dependency schemes

- Detect independence between variables
- Allow a wider range of heuristics
- Produce generalized proofs which don't lend to strategy extraction
- Standard, Resolution-Path dependency scheme - based on special kinds of connectivity in graph representations of the formula matrix

$$\exists x \forall y \; \{x\}, \{y\}$$

$$\exists x \forall y \; \{x, y\}, \{\overline{x}, y\}$$

Thank you for your attention!