NEVER TRUST YOUR SOLVER: CERTIFICATION FOR SAT AND QBF

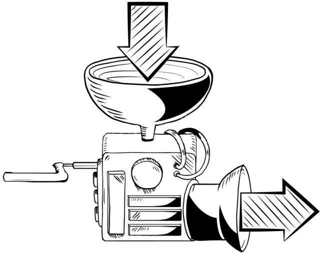
Martina Seidl

JOHANNES KEPLER
UNIVERSITY LINZ

Propositional Satisfiability Checking

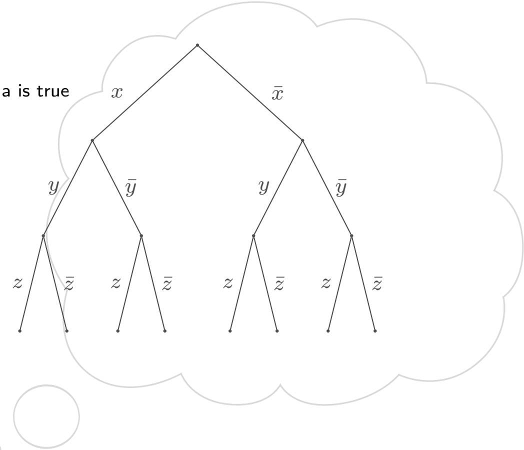
Is there a truth value for x, y, z such that the formula is true

$$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$$

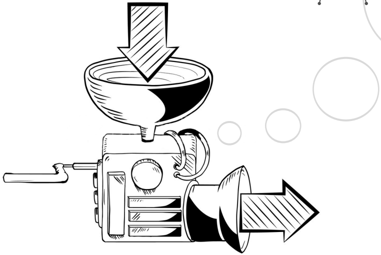


Propositional Satisfiability Checking

Is there a truth value for x, y, z such that the formula is true



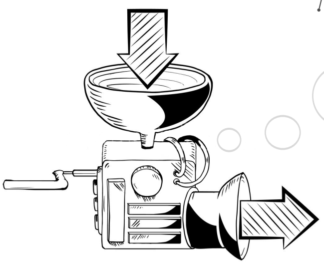
$$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$$

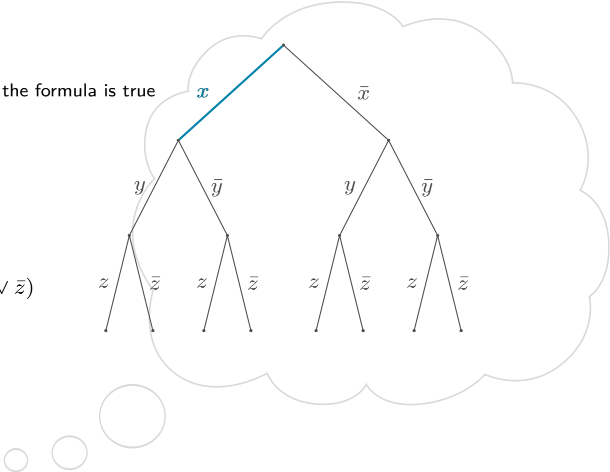


Propositional Satisfiability Checking

Is there a truth value for x, y, z such that the formula is true

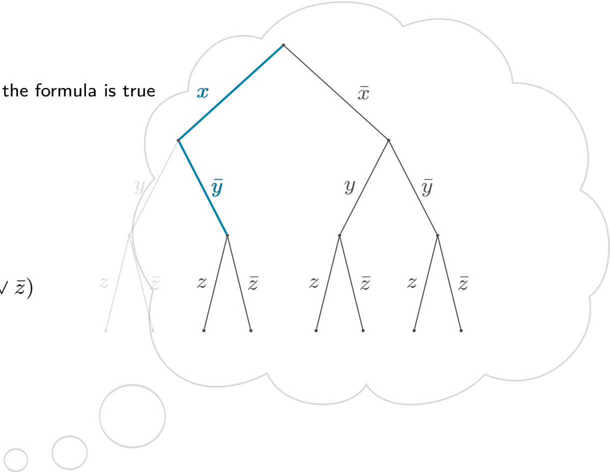
$$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$$



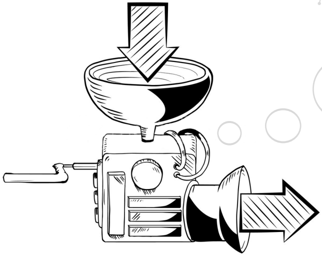


Propositional Satisfiability Checking

Is there a truth value for x, y, z such that the formula is true

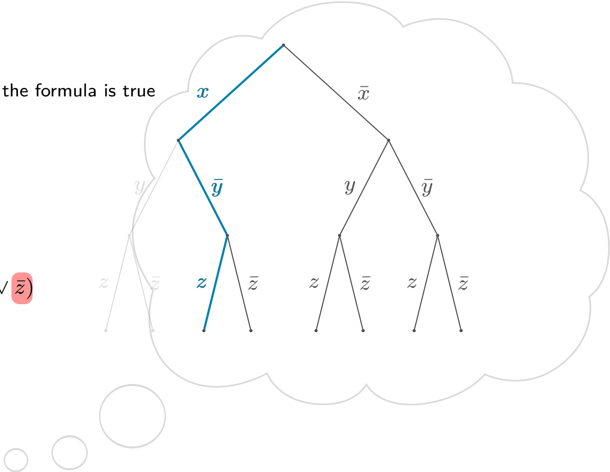


$$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$$

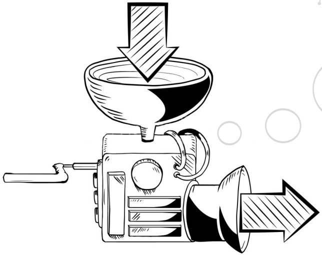


Propositional Satisfiability Checking

Is there a truth value for x, y, z such that the formula is true

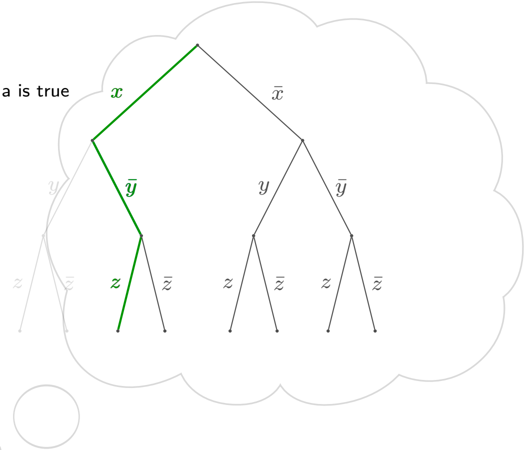


$(x \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (z \vee \bar{z})$

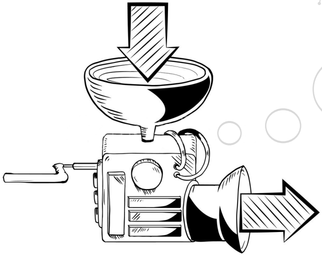


Propositional Satisfiability Checking

Is there a truth value for x, y, z such that the formula is true



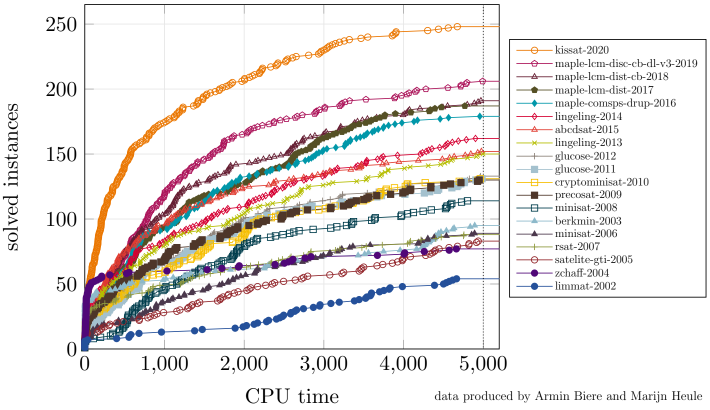
$$(x \lor y) \land (\bar{x} \lor \bar{y}) \land (z \lor \bar{z})$$



Satisfiable

Evolution of SAT Solver

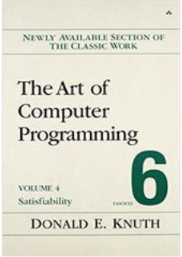
SAT Competition Winners on the SC2020 Benchmark Suite



data produced by Armin Biere and Marijn Heule

"SAT is a key technology of the 21st century."

-Edmund Clarke
Handbook of Satisfiability

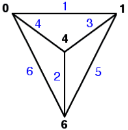


"The SAT problem is evidently a 'killer app,' because it is key to the solution of so many other problems."

-Donald Knuth
The Art of Computer Programming, vol. 4 on SAT

Practical Applications of SAT

formal verification

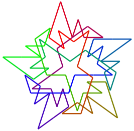


graph theory

bioinformatics

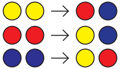
train safety

planning



combinatorics

cryptography



rewrite termination

encode → **SAT solver** → decode

from http://www.cs.utexas.edu/users/marijn/talks/Ptn-Linz.pdf

Propositional Logic

Elements of a formula:

- **literal**: variable or negated variable
- **clause**: disjunction of literals
- **formula in CNF (conjunctive normal form)**: conjunction of clauses

Example

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

Propositional Logic

Elements of a formula:

- **literal**: variable or negated variable
- **clause**: disjunction of literals
- **formula in CNF (conjunctive normal form)**: conjunction of clauses

Example

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

Semantics: A CNF formula is true under an assignment σ of the Boolean variables iff each clause contains at least one literal that is true under σ.

How to Ensure Correctness of SAT Solvers?

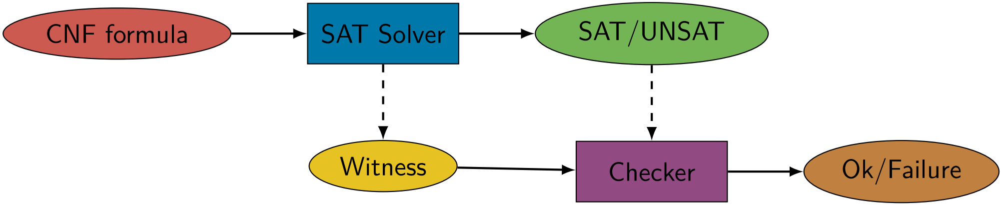
1. Carefull testing: incomplete

How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete
2. Verification of SAT Solver: not feasible in general

How to Ensure Correctness of SAT Solvers?

1. Carefull testing: incomplete
2. Verification of SAT Solver: not feasible in general
3. Check result by independent checker



Witnesses

- True formula: easy
  Check if the assignment returned by SAT solver is a satisfying assignment.

Witnesses

- True formula: easy
  Check if the assignment returned by SAT solver is a satisfying assignment.

- False formula: ??
  - unsatisfiability proof
  - ideally, checking is polynomial in the proof size

CERTIFICATION FOR SAT

Theoretical Background: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{C} \qquad \text{(cl-init)}$$

Theoretical Background: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{\mathsf{C}} \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \vee p \qquad C_2 \vee \bar{p}}{C_1 \vee C_2} \qquad (res)$$

Theoretical Background: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{\mathsf{C}} \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \vee p \qquad C_2 \vee \bar{p}}{C_1 \vee C_2} \qquad (res)$$

- in other words:
  $(\neg p \rightarrow C_1)$ AND $(p \rightarrow C_2)$ DERIVE $C_1 \vee C_2$

Theoretical Background: Resolution

Proof system with two rules:

Clause Axiom

$$\frac{}{\mathsf{C}} \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \lor p \qquad C_2 \lor \bar{p}}{C_1 \lor C_2} \qquad (res)$$

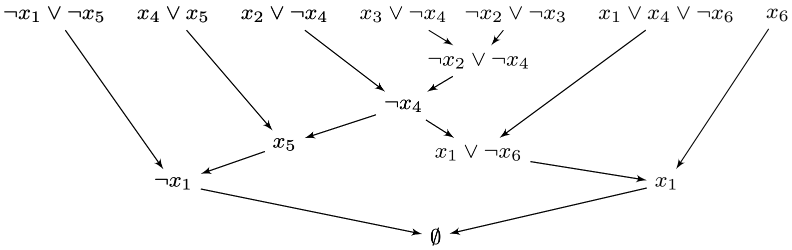
- in other words:
  $(\neg p \rightarrow C_1)$ AND $(p \rightarrow C_2)$ DERIVE $C_1 \lor C_2$
- resolution is **sound** and **complete**

Resolution Example

We **prove unsatisfiability** of

$$\{(\neg x_1 \vee \neg x_5), (x_4 \vee x_5), (x_2 \vee \neg x_4), (x_3 \vee \neg x_4), (\neg x_2 \vee \neg x_3), (x_1 \vee x_4 \vee \neg x_6), (x_6)\}$$

as follows:



More Background: Boolean Constraint Propagation (BCP)

Let ϕ be a formula in CNF containing a unit clause C, i.e., ϕ has a clause $C = (l)$ which consists only of literal l. Then $BCP(\phi, l)$ is obtained from ϕ by

- removing all clauses with l
- removing all occurrences of $\bar{l}$

- BCP can trigger other applications of BCP
- if BCP results in empty clause, then formula is unsatisfiable
- if BCP produces the empty CNF, then formula satisfiable

Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

1. $\phi' = BCP(\phi, a) = \{(b \vee \neg c), (\neg b)\}$

Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

1. $\phi' = BCP(\phi, a) = \{(b \vee \neg c), (\neg b)\}$
2. $\phi'' = BCP(\phi', \neg b) = \{(\neg c)\}$

Example BCP

$$\phi = \{(\neg a \vee b \vee \neg c), (a \vee b), (\neg a \vee \neg b), (a)\}$$

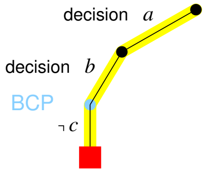
1. $\phi' = BCP(\phi, a) = \{(b \vee \neg c), (\neg b)\}$
2. $\phi'' = BCP(\phi', \neg b) = \{(\neg c)\}$
3. $\phi'' = BCP(\phi', c) = \{\} = \top$

Clause Learning by Example

$a = 1$

$b = 1$

$c = 0$



clauses

$\neg a \vee \neg b \vee \neg c$

$\neg a \vee \neg b \vee c$

$\neg a \vee b \vee \neg c$

$\neg a \vee b \vee c$

$a \vee \neg b \vee \neg c$

$a \vee \neg b \vee c$

$a \vee b \vee \neg c$

$a \vee b \vee c$

learn $\neg a \vee \neg b$

$$\bar{a} \vee \bar{b} \vee \bar{c} \qquad \bar{a} \vee \bar{b} \vee c$$

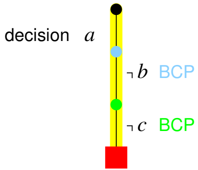
$$\bar{a} \vee \bar{b}$$

Clause Learning by Example

$a = 1$

$b = 0$

$c = 0$

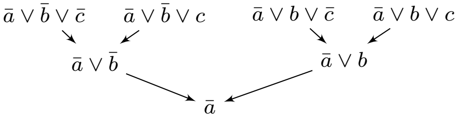


clauses

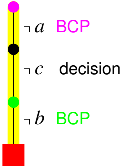
$\neg a \lor \neg b \lor \neg c$
$\neg a \lor \neg b \lor c$
$\neg a \lor b \lor \neg c$
$\neg a \lor b \lor c$
$a \lor \neg b \lor \neg c$
$a \lor \neg b \lor c$
$a \lor b \lor \neg c$
$a \lor b \lor c$

$\neg a \lor \neg b$

learn $\neg a$



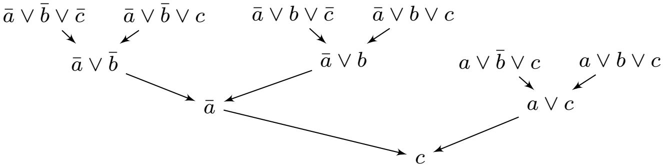
Clause Learning by Example



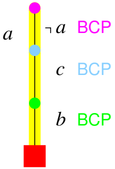
clauses

$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$
$\neg a \vee \neg b$
$\neg a$

learn c



Clause Learning by Example



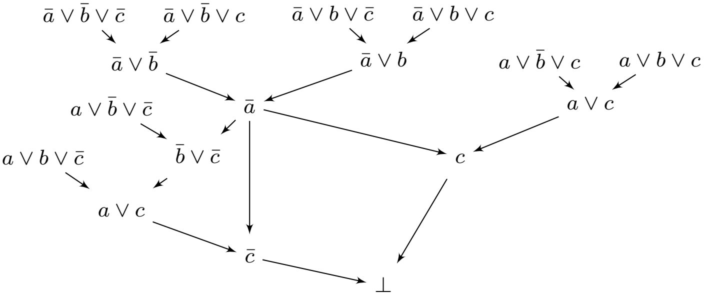
clauses

$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$

$\neg a \vee \neg b$
$\neg a$
c

learn $\bot$

empty clause



Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP$(\phi \wedge \bar{C}) = \bot$.

Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP$(\phi \wedge \bar{C}) = \bot$.

The RUP proof system allows the addition of RUP clauses.

Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP$(\phi \wedge \bar{C}) = \bot$.

The RUP proof system allows the addition of RUP clauses.

Example

input	learned
$\bar{a} \vee \bar{b} \vee c$	$\bar{a} \vee \bar{b}$
$\bar{a} \vee \bar{b} \vee \bar{c}$	$\bar{a}$
$a \vee \bar{b} \vee c$	c
$a \vee \bar{b} \vee \bar{c}$	$\bot$
$\bar{a} \vee b \vee c$	
$\bar{a} \vee b \vee \bar{c}$	
$a \vee b \vee c$	
$a \vee b \vee \bar{c}$	

Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP($\phi \wedge \bar{C}$) $= \bot$.

The RUP proof system allows the addition of RUP clauses.

Example

input	learned
$\bar{a} \vee \bar{b} \vee c$	$\bar{a} \vee \bar{b}$
$\bar{a} \vee \bar{b} \vee \bar{c}$	$\bar{a}$
$a \vee \bar{b} \vee c$	c
$a \vee \bar{b} \vee \bar{c}$	$\bot$
$\bar{a} \vee b \vee c$	
$\bar{a} \vee b \vee \bar{c}$	
$a \vee b \vee c$	
$a \vee b \vee \bar{c}$	

- BCP $(\phi \wedge a \wedge b) = \bot$

Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP$(\phi \wedge \bar{C}) = \bot$.

The RUP proof system allows the addition of RUP clauses.

Example

input	learned
$\bar{a} \vee \bar{b} \vee c$	$\bar{a} \vee \bar{b}$
$\bar{a} \vee \bar{b} \vee \bar{c}$	$\bar{a}$
$a \vee \bar{b} \vee c$	c
$a \vee \bar{b} \vee \bar{c}$	$\bot$
$\bar{a} \vee b \vee c$	
$\bar{a} \vee b \vee \bar{c}$	
$a \vee b \vee c$	
$a \vee b \vee \bar{c}$	

- BCP $(\phi \wedge a \wedge b) = \bot$
- BCP $(\phi \wedge (\bar{a} \vee \bar{b}) \wedge a) = \bot$

Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP$(\phi \wedge \bar{C}) = \bot$.

The RUP proof system allows the addition of RUP clauses.

Example

input	learned
$\bar{a} \vee \bar{b} \vee c$	$\bar{a} \vee \bar{b}$
$\bar{a} \vee \bar{b} \vee \bar{c}$	$\bar{a}$
$a \vee \bar{b} \vee c$	c
$a \vee \bar{b} \vee \bar{c}$	$\bot$
$\bar{a} \vee b \vee c$	
$\bar{a} \vee b \vee \bar{c}$	
$a \vee b \vee c$	
$a \vee b \vee \bar{c}$	

- BCP $(\phi \wedge a \wedge b) = \bot$
- BCP $(\phi \wedge (\bar{a} \vee \bar{b}) \wedge a) = \bot$
- BCP $(\phi \wedge (\bar{a} \vee \bar{b}) \wedge \bar{a} \wedge \bar{c}) = \bot$

Certification By Reverse Unit Propagation (RUP)

Let C be a clause and ϕ be a propositional formula. Then C is called a RUP clause wrt ϕ iff BCP$(\phi \wedge \bar{C}) = \bot$.

The RUP proof system allows the addition of RUP clauses.

Example

input	learned
$\bar{a} \vee \bar{b} \vee c$	$\bar{a} \vee \bar{b}$
$\bar{a} \vee \bar{b} \vee \bar{c}$	$\bar{a}$
$a \vee \bar{b} \vee c$	c
$a \vee \bar{b} \vee \bar{c}$	$\bot$
$\bar{a} \vee b \vee c$	
$\bar{a} \vee b \vee \bar{c}$	
$a \vee b \vee c$	
$a \vee b \vee \bar{c}$	

- BCP $(\phi \wedge a \wedge b) = \bot$
- BCP $(\phi \wedge (\bar{a} \vee \bar{b}) \wedge a) = \bot$
- BCP $(\phi \wedge (\bar{a} \vee \bar{b}) \wedge \bar{a} \wedge \bar{c}) = \bot$
- BCP $(\phi \wedge (\bar{a} \vee \bar{b}) \wedge \bar{a} \wedge c \wedge \top) = \bot$

Blocked Clauses are Redundant

Definition:

A literal $l \in C$ is blocked in CNF ϕ iff forall $D \in \phi$ with $\bar{l} \in D$, there is a literal k such that $k \in C$ and $\bar{k} \in D$. A clause with a blocked literal is called **blocked clause**.

Blocked Clauses are Redundant

Definition:

A literal $l \in C$ is blocked in CNF ϕ iff forall $D \in \phi$ with $\bar{l} \in D$, there is a literal k such that $k \in C$ and $\bar{k} \in D$. A clause with a blocked literal is called **blocked clause**.

- blocked literal elimination: remove clause with blocked literal l
  - removal of blocked clauses preserves unsatisfiability
  - NOT model preserving
- powerful simplification technique
  - simulation of several circuit-level simplification techniques
- generalization of pure literal elimination

Blocked Clauses are Redundant

Definition:

A literal $l \in C$ is blocked in CNF ϕ iff forall $D \in \phi$ with $\bar{l} \in D$, there is a literal k such that $k \in C$ and $\bar{k} \in D$. A clause with a blocked literal is called **blocked clause**.

- blocked literal elimination: remove clause with blocked literal l
  - removal of blocked clauses preserves unsatisfiability
  - NOT model preserving
- powerful simplification technique
  - simulation of several circuit-level simplification techniques
- generalization of pure literal elimination

Example

the formula

$$(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$

is solvable by blocked clause elimination

Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

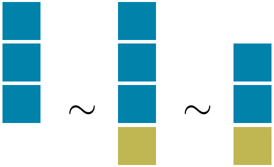
The RAT proof system allows the addition and deletion of RAT clauses.

Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF

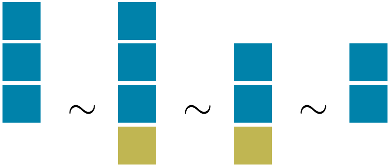


Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF

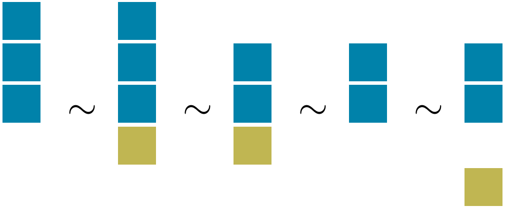


Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF

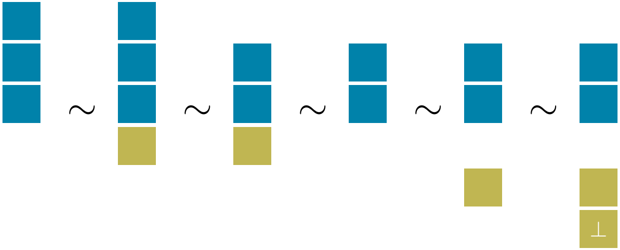


Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF

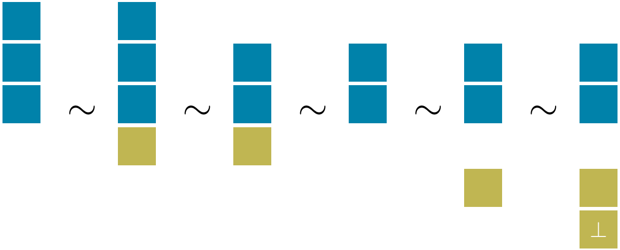


Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF



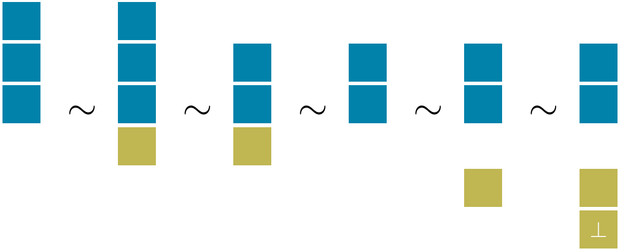
- very powerful proof system

Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF



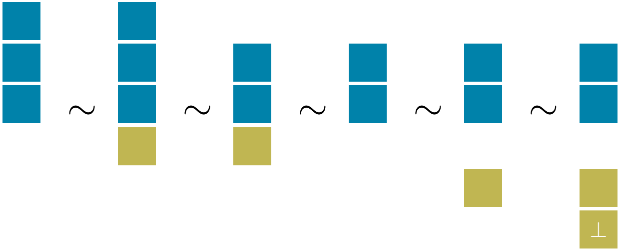
- very powerful proof system
- standard in state-of-the-art SAT solving

Certification By DRAT

Let C be a clause and ϕ be a propositional formula. Then C is called a RAT clause on literal l wrt ϕ iff for all $D \in \phi$ with $\bar{l} \in D$, the resolvent of C and D is a RUP wrt to ϕ.

The RAT proof system allows the addition and deletion of RAT clauses.

Input CNF



- very powerful proof system
- standard in state-of-the-art SAT solving
- verified checkers available

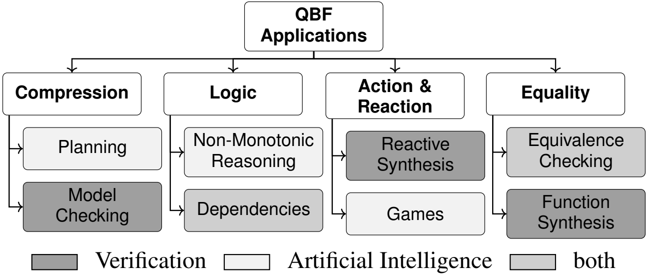
QUANTIFIED BOOLEAN FORMULAS

Quantified Boolean Formulas (QBF)

- Extension of propositional logic
  - explicit quantifiers ($\forall$, $\exists$) over the Boolean variables
- Canonical PSPACE-complete problem
  - more succinct encoding than SAT (NP-complete)
- Many application domains: synthesis, AI, verification, ...

Quantified Boolean Formulas (QBF)

- Extension of propositional logic
  - explicit quantifiers (∀, ∃) over the Boolean variables
- Canonical PSPACE-complete problem
  - more succinct encoding than SAT (NP-complete)
- Many application domains: synthesis, AI, verification, ...



QBF Syntax

- **QBFs in Prenex CNF (PCNF):**

$$\exists x \exists y \forall u \exists z . \underbrace{(\overset{\text{literals}}{\overset{\downarrow}{\neg u}} \vee \overset{\downarrow}{z}) \wedge \overbrace{(y \vee u \vee \neg z)}^{\text{clause}} \wedge (x \vee \neg u \vee \neg z)}_{\text{CNF}}$$

QBF Syntax

- **QBFs in Prenex CNF (PCNF):**

$$\exists x \exists y \forall u \exists z.\underbrace{(\overset{\text{literals}}{\overset{\downarrow}{\neg u}} \vee \overset{\downarrow}{z}) \wedge \overbrace{(y \vee u \vee \neg z)}^{\text{clause}} \wedge (x \vee \neg u \vee \neg z)}_{\text{CNF}}$$

- **QBFs in Prenex DNF (PDNF):**

$$\forall x \forall y \exists u \forall z.\underbrace{\overbrace{(u \wedge \neg z)}^{\text{cube}} \vee (\neg y \wedge \neg u \wedge z) \vee (\neg x \wedge u \wedge z)}_{\text{DNF}}$$

QBF Syntax

- **QBFs in Prenex CNF (PCNF):**

$$\exists x \exists y \forall u \exists z.\underbrace{(\overset{\text{literals}}{\neg u} \vee z) \wedge \overbrace{(y \vee u \vee \neg z)}^{\text{clause}} \wedge (x \vee \neg u \vee \neg z)}_{\text{CNF}}$$

- **QBFs in Prenex DNF (PDNF):**

$$\forall x \forall y \exists u \forall z.\underbrace{\overbrace{(u \wedge \neg z)}^{\text{cube}} \vee (\neg y \wedge \neg u \wedge z) \vee (\neg x \wedge u \wedge z)}_{\text{DNF}}$$

- **QBFs in Prenex Non-CNF**

QBF Syntax

- **QBFs in Prenex CNF (PCNF):**

$$\exists x \exists y \forall u \exists z.\underbrace{(\overset{\text{literals}}{\neg u} \vee z) \wedge \overbrace{(y \vee u \vee \neg z)}^{\text{clause}} \wedge (x \vee \neg u \vee \neg z)}_{\text{CNF}}$$

- **QBFs in Prenex DNF (PDNF):**

$$\forall x \forall y \exists u \forall z.\underbrace{\overbrace{(u \wedge \neg z)}^{\text{cube}} \vee (\neg y \wedge \neg u \wedge z) \vee (\neg x \wedge u \wedge z)}_{\text{DNF}}$$

- **QBFs in Prenex Non-CNF**

Note: $x, y < u < z$

QBF Semantics

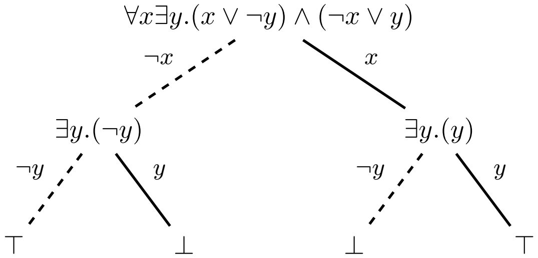
- $\forall x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **and** $\mathcal{Q}.\varphi[\neg x]$ true

QBF Semantics

- $\forall x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **and** $\mathcal{Q}.\varphi[\neg x]$ true
- $\exists x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **or** $\mathcal{Q}.\varphi[\neg x]$ true

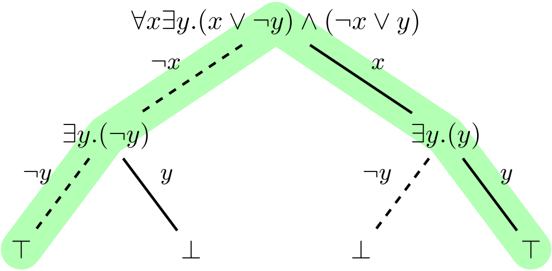
QBF Semantics

- $\forall x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **and** $\mathcal{Q}.\varphi[\neg x]$ true
- $\exists x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **or** $\mathcal{Q}.\varphi[\neg x]$ true
- Example:



QBF Semantics

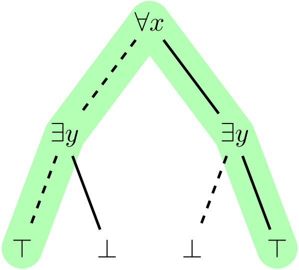
- $\forall x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **and** $\mathcal{Q}.\varphi[\neg x]$ true
- $\exists x \mathcal{Q}.\varphi$ true $\Leftrightarrow$ $\mathcal{Q}.\varphi[x]$ **or** $\mathcal{Q}.\varphi[\neg x]$ true
- Example:



Example: QBF (Counter-)Model

Tree model of **true** formula:

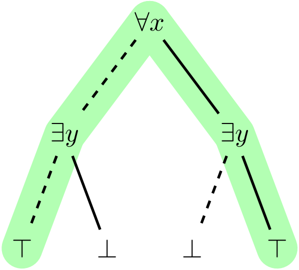
$$\forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$



Example: QBF (Counter-)Model

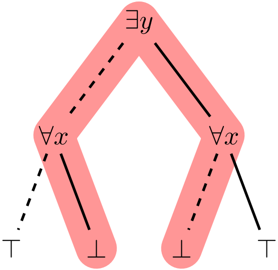
Tree model of **true** formula:

$$\forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$



Tree refutation of **false** formula:

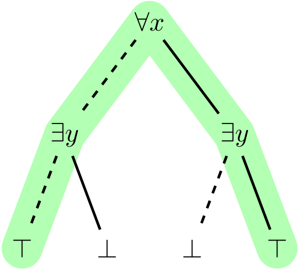
$$\exists y \forall x.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$



Example: QBF (Counter-)Model

Tree model of **true** formula:

$$\forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$

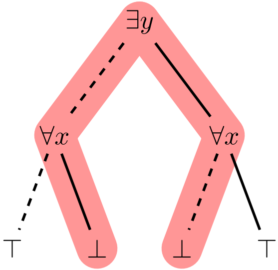


Skolem-functions of ∃-variables:

$$f_y(x) = x$$

Tree refutation of **false** formula:

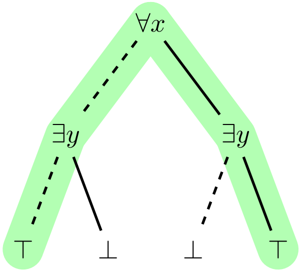
$$\exists y \forall x.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$



Example: QBF (Counter-)Model

Tree model of **true** formula:

$$\forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$

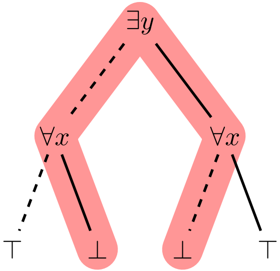


Skolem-functions of $\exists$-variables:

$$f_y(x) = x$$

Tree refutation of **false** formula:

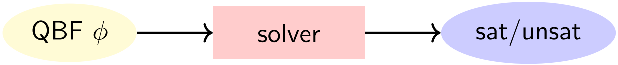
$$\exists y \forall x.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$$



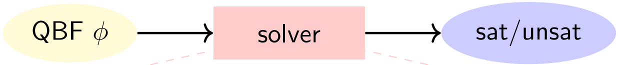
Herbrand-functions of $\forall$-variables:

$$f_x(y) = \bar{y}$$

Overview: Proof Systems for QBF



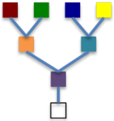
Overview: Proof Systems for QBF



different proof systems:

Q-Resolution

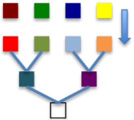
[Kleine Büning, Karpinski, Flögel, 95]



- QCDCL

Expansion

[Beyersdorff, Chew, Janota, 14]



- CEGAR

Interference

[Heule, Seidl, Biere, 14]

- QRAT

Solutions of QBFs

Definition (Dependency):

Let ϕ be a QBF in prenex form and v a variable of ϕ with $quant(v) \in \{\exists, \forall\}$. Then

$$D^{\phi}(v) = \{w \in vars(\phi) \mid w < v, quant(v) \neq quant(w)\}$$

Solutions of QBFs

Definition (Dependency):

Let ϕ be a QBF in prenex form and v a variable of ϕ with $quant(v) \in \{\exists, \forall\}$. Then

$$D^\phi(v) = \{w \in vars(\phi) \mid w < v, quant(v) \neq quant(w)\}$$

Solutions of false QBFs ϕ (counter-models)

- for all universal variables $x_1, \ldots, x_m$ of ϕ define Herbrand function $f_{x_i}(y_1, \ldots, y_n)$ with $D^\phi(x) = \{y_1, \ldots, y_n\}$
- $\phi[x_1/f_{x_1}, \ldots x_m/f_{x_m}]$ is unsat

Solutions of QBFs

Definition (Dependency):

Let ϕ be a QBF in prenex form and v a variable of ϕ with $quant(v) \in \{\exists, \forall\}$. Then

$$D^{\phi}(v) = \{w \in vars(\phi) \mid w < v, quant(v) \neq quant(w)\}$$

Solutions of false QBFs ϕ (counter-models)

- for all universal variables $x_1, \dots, x_m$ of ϕ define Herbrand function $f_{x_i}(y_1, \dots, y_n)$ with $D^{\phi}(x) = \{y_1, \dots, y_n\}$
- $\phi[x_1/f_{x_1}, \dots x_m/f_{x_m}]$ is unsat

Solutions of true QBF ϕ (models)

- for all existential variables $x_1, \dots, x_m$ of ϕ define Skolem function $f_{x_i}(y_1, \dots, y_n)$ with $D^{\phi}(x) = \{y_1, \dots, y_n\}$
- $\phi[x_1/f_{x_1}, \dots x_m/f_{x_m}]$ is valid

How To Get Solutions?

Special case: only values of variables in outermost quantifier are of interest

- false QBF is of form $\forall X \exists Y \Pi.\psi$
- true QBF is of form $\exists X \forall Y \Pi.\psi$

How To Get Solutions?

Special case: only values of variables in outermost quantifier are of interest

- false QBF is of form $\forall X \exists Y \Pi.\psi$
- true QBF is of form $\exists X \forall Y \Pi.\psi$

$\Rightarrow$ solutions are propositional assignments

How To Get Solutions?

Special case: only values of variables in outermost quantifier are of interest

- false QBF is of form $\forall X \exists Y \Pi.\psi$
- true QBF is of form $\exists X \forall Y \Pi.\psi$

$\Rightarrow$ solutions are propositional assignments

$\Rightarrow$ many solvers are able to produce such assignments

How To Get Solutions?

Special case: only values of variables in outermost quantifier are of interest

- false QBF is of form $\forall X \exists Y \Pi.\psi$
- true QBF is of form $\exists X \forall Y \Pi.\psi$

$\Rightarrow$ solutions are propositional assignments

$\Rightarrow$ many solvers are able to produce such assignments

General case: ???

How To Get Solutions?

Special case: only values of variables in outermost quantifier are of interest

- false QBF is of form $\forall X \exists Y \Pi.\psi$
- true QBF is of form $\exists X \forall Y \Pi.\psi$

$\Rightarrow$ solutions are propositional assignments

$\Rightarrow$ many solvers are able to produce such assignments

General case: ???

$\Rightarrow$ extraction of solutions from proofs

Q-Resolution for False Formulas

Clause Axiom

$$\frac{}{C} \quad \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq C,\ C \text{ is a clause and } C \in \psi \qquad \text{(cl-init)}$$

Q-Resolution for False Formulas

Clause Axiom

$$\overline{\text{C}} \quad \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq C,\ C \text{ is a clause and } C \in \psi \qquad \text{(cl-init)}$$

Resolution Rule

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \begin{array}{l} \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2), \\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and} \\ C_1, C_2 \text{ are clauses, quant}(\mathcal{Q}, p) = \exists \end{array} \qquad (res)$$

Q-Resolution for False Formulas

Clause Axiom

$$\frac{}{\mathsf{C}} \quad \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq C,\ C \text{ is a clause and } C \in \psi \qquad (\text{cl-init})$$

Resolution Rule

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2),\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and } C_1, C_2 \text{ are clauses, } \mathrm{quant}(\mathcal{Q}, p) = \exists \qquad (res)$$

Universal Reduction

$$\frac{C \cup \{l\}}{\mathsf{C}} \quad \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C \cup \{l\}) \text{ and either } C \text{ is a clause, } \mathrm{quant}(\mathcal{Q}, l) = \forall,\ l' <_{\mathcal{Q}} l \text{ for all } l' \in C \text{ with } \mathrm{quant}(\mathcal{Q}, l') = \exists \text{ or} \qquad (red)$$

Q-Resolution Example

Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

Q-Resolution Example

Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

Truth Table

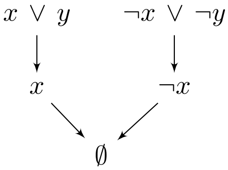
x	y	ψ
0	0	0
0	1	1
1	0	1
1	1	0

false

Q-Resolution Example

Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

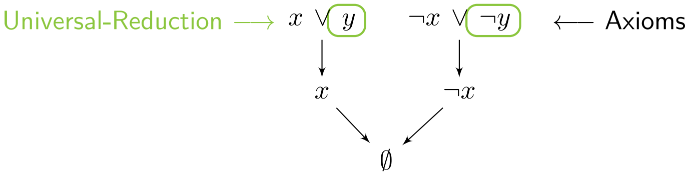
Q-Resolution Proof



Q-Resolution Example

Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

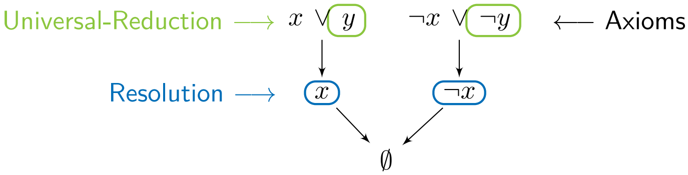
Q-Resolution Proof



Q-Resolution Example

Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

Q-Resolution Proof

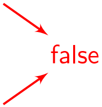


Q-Resolution Example

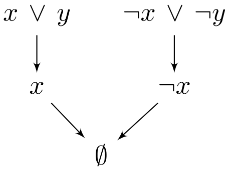
Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

Truth Table

x	y	ψ
0	0	0
0	1	1
1	0	1
1	1	0



Q-Resolution Proof



$$\longrightarrow \quad y = x \quad \Rightarrow \quad \psi = 0$$

Q-Resolution Example

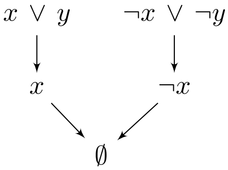
Exclusive OR (XOR): QBF $\psi = \exists x \forall y (x \vee y) \wedge (\neg x \vee \neg y)$

Truth Table

x	y	ψ
0	0	0
0	1	1
1	0	1
1	1	0

false

Q-Resolution Proof



$\longrightarrow \quad y = x \;\Rightarrow\; \psi = 0$

$\longrightarrow \quad f_y(x) = x$ (counter model)

Q-Resolution for True Formulas

Cube Axiom

$$\frac{}{C} \quad \begin{array}{l} A \text{ is an assignment,} \\ \phi[A] = \top, \\ \text{and } C = (\bigwedge_{l \in A} l) \text{ is a cube} \end{array} \qquad \text{(cu-init)}$$

Q-Resolution for True Formulas

Cube Axiom

$$\frac{}{\text{C}} \qquad \begin{array}{l} A \text{ is an assignment,} \\ \phi[A] = \top, \\ \text{and } C = (\bigwedge_{l \in A} l) \text{ is a cube} \end{array} \qquad \text{(cu-init)}$$

Resolution Rule

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \qquad \begin{array}{l} \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2), \\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and} \\ C_1, C_2 \text{ are cubes, quant}(\mathcal{Q}, p) = \forall \end{array} \qquad (res)$$

Q-Resolution for True Formulas

Cube Axiom

$$\frac{}{\text{C}} \qquad \begin{array}{l} A \text{ is an assignment,} \\ \phi[A] = \top, \\ \text{and } C = (\bigwedge_{l \in A} l) \text{ is a cube} \end{array} \qquad (\text{cu-init})$$

Resolution Rule

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \qquad \begin{array}{l} \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2), \\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and} \\ C_1, C_2 \text{ are cubes, } \text{quant}(\mathcal{Q}, p) = \forall \end{array} \qquad (res)$$

Existential Reduction

$$\frac{C \cup \{l\}}{\text{C}} \qquad \begin{array}{l} \text{if for all } x \in \mathcal{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C \cup \{l\}) \text{ and either} \\ C \text{ is a cube, } \text{quant}(\mathcal{Q}, l) = \exists, \\ \quad l' <_{\mathcal{Q}} l \text{ for all } l' \in C \text{ with } \text{quant}(\mathcal{Q}, l') = \forall \end{array} \qquad (red)$$

Function Extraction from Q-Resolution Proofs

- QCDCL solver produce Q-resolution proofs of empty clause/cube

Function Extraction from Q-Resolution Proofs

- QCDCL solver produce Q-resolution proofs of empty clause/cube
- From proof P Skolem/Herbrand function can be obtained

Function Extraction from Q-Resolution Proofs

- QCDCL solver produce Q-resolution proofs of empty clause/cube
- From proof P Skolem/Herbrand function can be obtained
- Runtime of extraction is linear in the size of P (but size of P can be exponential!!!)

Function Extraction from Q-Resolution Proofs

- QCDCL solver produce Q-resolution proofs of empty clause/cube
- From proof P Skolem/Herbrand function can be obtained
- Runtime of extraction is linear in the size of P (but size of P can be exponential!!!)
- Approach by Jiang and Balabanov (CAV 2011):
  - Visit clauses of P in topological ordering
  - Inspect universal (existential) reduction steps
  - Update functions of reduced variables

Certification by Example

Input Formula

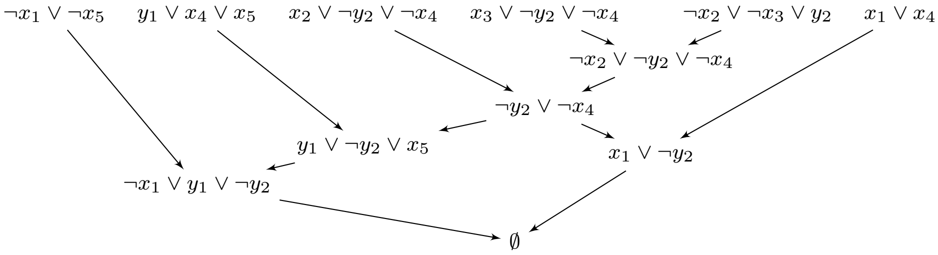
$$\exists x_1 \forall y_1 \exists x_2 x_3 \forall y_2 \exists x_4 x_5 . (\neg x_1 \vee \neg x_5) \wedge (y_1 \vee x_4 \vee x_5) \wedge (x_2 \vee \neg y_2 \vee \neg x_4) \wedge (x_3 \vee \neg y_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee y_2) \wedge (x_1 \vee x_4)$$

Certification by Example

Input Formula

$$\exists x_1 \forall y_1 \exists x_2 x_3 \forall y_2 \exists x_4 x_5 . (\neg x_1 \lor \neg x_5) \land (y_1 \lor x_4 \lor x_5) \land (x_2 \lor \neg y_2 \lor \neg x_4) \land (x_3 \lor \neg y_2 \lor \neg x_4) \land (\neg x_2 \lor \neg x_3 \lor y_2) \land (x_1 \lor x_4)$$

Q-Resolution Proof DAG

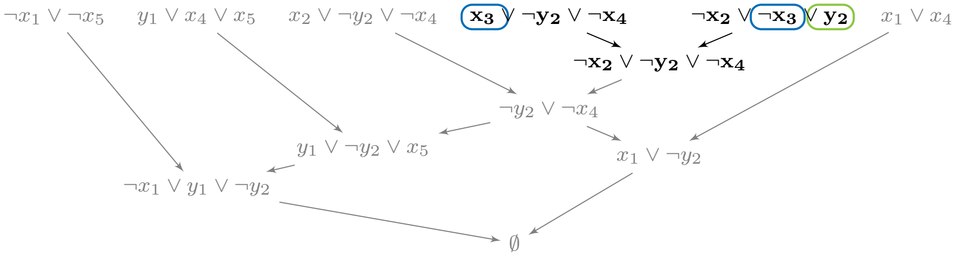


Certification by Example

Input Formula

$$\exists x_1 \forall y_1 \exists x_2 x_3 \forall y_2 \exists x_4 x_5 . (\neg x_1 \vee \neg x_5) \wedge (y_1 \vee x_4 \vee x_5) \wedge (x_2 \vee \neg y_2 \vee \neg x_4) \wedge (x_3 \vee \neg y_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee y_2) \wedge (x_1 \vee x_4)$$

Q-Resolution Proof DAG

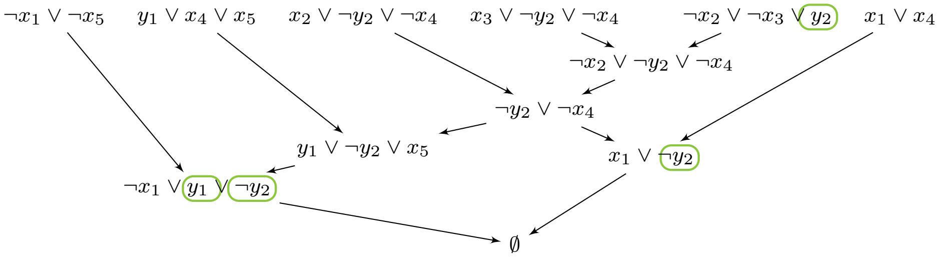


Certification by Example

Input Formula

$$\exists x_1 \forall y_1 \exists x_2 x_3 \forall y_2 \exists x_4 x_5 . (\neg x_1 \vee \neg x_5) \wedge (y_1 \vee x_4 \vee x_5) \wedge (x_2 \vee \neg y_2 \vee \neg x_4) \wedge (x_3 \vee \neg y_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee y_2) \wedge (x_1 \vee x_4)$$

Q-Resolution Proof DAG



Extracted Herbrand Functions

$$\left.\begin{array}{l} f_{\mathbf{y_1}}(x_1) = \neg x_1 \\ f_{\mathbf{y_2}}(x_2, x_3) = \neg x_2 \vee \neg x_3 \end{array}\right\} \text{Certificate}$$

SUMMARY

Summary:

- proof theory is important for practical solving
  - it explains what solvers do
  - it gives a tool for checking the solving result
-
- improvement of the quality of solvers
- standard in SAT solving, in progress for QBF

Summary:

- proof theory is important for practical solving
  - it explains what solvers do
  - it gives a tool for checking the solving result
-
- improvement of the quality of solvers
- standard in SAT solving, in progress for QBF

Challenges:

- proof size
- parallel solving
- heterogeneity in QBF solving approaches