Practical Aspects of Dependency Schemes in QBF Solving

Florian Lonsing and Armin Biere

Institute for Formal Models and Verification (FMV)
Johannes Kepler University, Linz, Austria
http://fmv.jku.at

Alpine Verification Meeting (AVM)
October 18 - 19, 2010
Lugano, Switzerland

JOHANNES KEPLER UNIVERSITY LINZ | JKU

TNF

Questions to be Discussed:

- What is QBF? Propositional logic with quantification.
- How to solve a QBF? Our focus: backtracking search.
- What makes it difficult?
  Structural property of QBFs: dependent variables.
- Theoretical solution? Identifying variable independence.
- Practical solution?
  Combining backtracking search with dependency schemes.
- Observable effects? Experiments.
- Conclusions and open problems.

Propositional Logic (SAT):

- Our focus: formulae in conjunctive normal form (CNF).
- Boolean variables $V := \{x_1, \ldots, x_n\}$, literals $l := v$ or $l := \overline{v}$ for $v \in V$.
- Clauses $C_i := (l_1 \vee \ldots \vee l_{k_i})$, CNF $\phi := \bigwedge_{i:=1}^{n} C_i$.

Quantified Boolean Formulae (QBF):

- Prenex CNF: quantifier-free CNF over quantified Boolean variables.
- PCNF $F := Q_1 x_1 \ldots Q_n x_n.\ \phi$, where $Q_i \in \{\exists, \forall\}$ (i.e. no free variables).
- $Q_i x_i \leq Q_{i+1} x_{i+1}$: variables are linearly ordered.
- *Prefix order limits freedom in QBF solving (to be continued!).*

Example

A CNF: $(x \vee \overline{y}) \wedge (\overline{x} \vee y)$, and a PCNF: $\forall x \exists y.\ (x \vee \overline{y}) \wedge (\overline{x} \vee y)$.

QBF Applications:

- Compact encodings in verification e.g. bounded model checking (BMC).

Drawbacks of Prenex CNF

Solving QBF by Backtracking Search:

- QDPLL: based on DPLL algorithm for SAT.
- PCNF $Q_1 x_1 \ldots Q_n x_n.\ \phi$: must branch on variables in prefix order.
  - $\exists a \forall x, y \exists b.\ \phi$: branching on b possible by prefix order only if x, y assigned.

Respecting Prefix Order is Crucial:

Example

- $\forall x \exists y.\ (x = y)$ is satisfiable: value of y *depends* on value of x.
- $\exists y \forall x.\ (x = y)$ is unsatisfiable: value of y is fixed for all values of x.
- $\forall x \exists y.\ (x = y)$: branching on y before x was assigned is unsound!

Can Prefix Order be Relaxed to Increase Freedom?

- Set of branching variables depends on prefix order.
- Theoretically: can we go from linear to *partial order* on the variables?
- Partial order R: $(x, y) \notin R$ allows arbitrary assignment order of x, y.

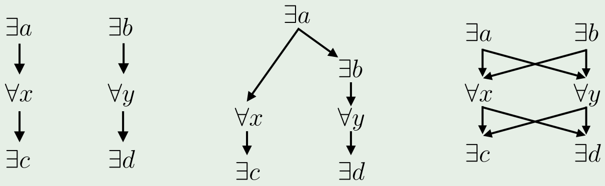
Independence of Variables: different assignment orders preserve result.

Dependency Schemes: $D \subseteq (V_\exists \times V_\forall) \cup (V_\forall \times V_\exists)$.

- PCNF-based binary relations based on QBF semantics.
- Conservative (i.e. sound) over-approximations of full independence.
  - $(x, y) \notin D$: y independent from x.
  - $(x, y) \in D$: *conservatively* regard y as depending on x.
- Trivial (D^{triv}), standard dependency scheme (D^{std}), quantifier trees (D^{tree}).

Example

$$\exists a, b \forall x, y \exists c, d.\ (a \vee x \vee c) \wedge (a \vee b) \wedge (b \vee d) \wedge (y \vee d).$$



D^{std} D^{tree} D^{triv}

Improvements Over Prefix: $D^{\text{std}} \subseteq D^{\text{tree}} \subseteq D^{\text{triv}}$ (theoretically).

```
State qdpll ()
  while (true)
    State s = bcp ();
    if (s == UNDEF)
      // Make decision.
      v = select_branch_var ();
      assign_branch_var (v);
    else
      // Conflict or solution.
      // s == UNSAT or s == SAT.
      btlevel = analyze_leaf (s);
      if (btlevel == INVALID)
        return s;
      else
        backtrack (btlevel);
```

```
DecLevel analyze_leaf (State s)
  R = get_initial_reason (s);
  // s == UNSAT: 'R' is empty clause.
  // s == SAT: 'R' is sat. cube...
  // ..or new cube from assignment.
  while (!stop_res (R))
    p = get_pivot (R);
    A = get_antecedent (p);
    R = constraint_res (R, p, A);
  add_to_formula (R);
  assign_forced_lit (R);
  return get_asserting_level (R);
```

Figure: QDPLL with conflict-driven clause and solution-driven cube learning.

Backtracking Search with Constraint Learning:

- Classical QDPLL based on quantifier prefix, i.e. D^{triv}.
- `bcp`: propagate implied (i.e. necessary) assignments.
- `select_branch_var`: branching.
- `analyze_leaf`: add learned constraint produced by Q-resolution.

```
State qdpll ()
  while (true)
    State s = bcp ();
    if (s == UNDEF)
      // Make decision.
      v = select_branch_var ();
      assign_branch_var (v);
    else
      // Conflict or solution.
      // s == UNSAT or s == SAT.
      btlevel = analyze_leaf (s);
      if (btlevel == INVALID)
        return s;
      else
        backtrack (btlevel);
```

```
DecLevel analyze_leaf (State s)
  R = get_initial_reason (s);
  // s == UNSAT: 'R' is empty clause.
  // s == SAT: 'R' is sat. cube...
  // ..or new cube from assignment.
  while (!stop_res (R))
    p = get_pivot (R);
    A = get_antecedent (p);
    R = constraint_res (R, p, A);
  add_to_formula (R);
  assign_forced_lit (R);
  return get_asserting_level (R);
```

Figure: QDPLL with conflict-driven clause and solution-driven cube learning.

Replacing D^{triv} with Arbitrary Partial Order $D \subseteq D^{\text{triv}}$:

- Same basic framework: considering D as a parameter of QDPLL.
- Only change: representation of D for dependency checking.
- Expecting more implications, shorter learned constraints.
- Expecting more freedom for selecting branching variables.

D-Aware Unit Literal Detection

Given dependency scheme D for PCNF F. Write $x \prec y$ if $(x, y) \in D$.

Definition (Unit Clause Rule)

A clause $C \in F$ is *unit* under a partial assignment iff

- no literal $l \in C$ is assigned true,
- exactly one existential literal $l_e \in L_\exists(C)$ is unassigned,
- for all unassigned universal literals $l_u \in L_\forall(C)$: $l_u \not\prec l_e$.

If C is unit then assigning l_e to true is necessary for F-satisfiability.

Example

$\exists x \forall a \exists y, z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Assign $\overline{x}, \overline{y}$: $\exists x \forall a \exists y, z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Given D^{triv} from prefix: $(x \vee a \vee y \vee z)$ *not* unit since $a \prec z$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec z$: $(x \vee a \vee y \vee z)$ unit.

Practical Effect: expecting more units when using $D \subseteq D^{\text{triv}}$.

D-Aware Unit Literal Detection

Given dependency scheme D for PCNF F. Write $x \prec y$ if $(x, y) \in D$.

Definition (Unit Clause Rule)

A clause $C \in F$ is *unit* under a partial assignment iff

- no literal $l \in C$ is assigned true,
- exactly one existential literal $l_e \in L_\exists(C)$ is unassigned,
- for all unassigned universal literals $l_u \in L_\forall(C)$: $l_u \not\prec l_e$.

If C is unit then assigning l_e to true is necessary for F-satisfiability.

Example

$\exists x \forall a \exists y,z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Assign $\overline{x}, \overline{y}$: $\exists x \forall a \exists y,z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Given D^{triv} from prefix: $(x \vee a \vee y \vee z)$ *not* unit since $a \prec z$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec z$: $(x \vee a \vee y \vee z)$ unit.

Practical Effect: expecting more units when using $D \subseteq D^{\text{triv}}$.

D-Aware Unit Literal Detection

Given dependency scheme D for PCNF F. Write $x \prec y$ if $(x, y) \in D$.

Definition (Unit Clause Rule)

A clause $C \in F$ is *unit* under a partial assignment iff

- no literal $l \in C$ is assigned true,
- exactly one existential literal $l_e \in L_\exists(C)$ is unassigned,
- for all unassigned universal literals $l_u \in L_\forall(C)$: $l_u \not\prec l_e$.

If C is unit then assigning l_e to true is necessary for F-satisfiability.

Example

$\exists x \forall a \exists y, z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Assign $\overline{x}, \overline{y}$: $\exists x \forall a \exists y, z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Given D^{triv} from prefix: $(x \vee a \vee y \vee z)$ *not* unit since $a \prec z$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec z$: $(x \vee a \vee y \vee z)$ unit.

Practical Effect: expecting more units when using $D \subseteq D^{\text{triv}}$.

D-Aware Unit Literal Detection

Given dependency scheme D for PCNF F. Write $x \prec y$ if $(x, y) \in D$.

Definition (Unit Clause Rule)

A clause $C \in F$ is *unit* under a partial assignment iff

- no literal $l \in C$ is assigned true,
- exactly one existential literal $l_e \in L_\exists(C)$ is unassigned,
- for all unassigned universal literals $l_u \in L_\forall(C)$: $l_u \not\prec l_e$.

If C is unit then assigning l_e to true is necessary for F-satisfiability.

Example

$\exists x \forall a \exists y, z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Assign $\overline{x}, \overline{y}$: $\exists x \forall a \exists y, z.\ \phi' \wedge (x \vee a \vee y \vee z)$.

Given D^{triv} from prefix: $(x \vee a \vee y \vee z)$ *not* unit since $a \prec z$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec z$: $(x \vee a \vee y \vee z)$ unit.

Practical Effect: expecting more units when using $D \subseteq D^{\text{triv}}$.

D-Aware Constraint Reduction

Constraint Reduction (CR): universal reduction of clauses.

- Delete literals of universally quantified variables from clauses.

Definition (Universal Reduction of Clauses)

A universal literal $l_u \in L_\forall(C)$ can be deleted from a clause $C \in F$ iff

- there is no existential $l_e \in L_\exists(C)$ with $l_u \prec l_e$.

The result of saturated universal reduction is denoted by $CR(C)$.

Example

$\exists x \forall a \exists y.\ \phi' \wedge (x \vee a \vee y)$.

Given D^{triv} from prefix: a is irreducible in $(x \vee a \vee y)$ since $a \prec y$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec y$: a is reducible in $(x \vee a \vee y)$, yielding $(x \vee y)$.

CR and Unit Literals: unit literal rule applies CR implicitly.

Practical Effect: expecting shorter learnt constraints when using $D \subseteq D^{\text{triv}}$.

Constraint Reduction (CR): universal reduction of clauses.

- Delete literals of universally quantified variables from clauses.

Definition (Universal Reduction of Clauses)

A universal literal $l_u \in L_\forall(C)$ can be deleted from a clause $C \in F$ iff

- there is no existential $l_e \in L_\exists(C)$ with $l_u \prec l_e$.

The result of saturated universal reduction is denoted by $CR(C)$.

Example

$\exists x \forall a \exists y.\ \phi' \wedge (x \vee a \vee y)$.

Given D^{triv} from prefix: a is irreducible in $(x \vee a \vee y)$ since $a \prec y$.

Given $D \subseteq D^{\mathrm{triv}}$ where $a \not\prec y$: a is reducible in $(x \vee a \vee y)$, yielding $(x \vee y)$.

CR and Unit Literals: unit literal rule applies CR implicitly.

Practical Effect: expecting shorter learnt constraints when using $D \subseteq D^{\mathrm{triv}}$.

Constraint Reduction (CR): universal reduction of clauses.

- Delete literals of universally quantified variables from clauses.

Definition (Universal Reduction of Clauses)

A universal literal $l_u \in L_\forall(C)$ can be deleted from a clause $C \in F$ iff

- there is no existential $l_e \in L_\exists(C)$ with $l_u \prec l_e$.

The result of saturated universal reduction is denoted by $CR(C)$.

Example

$\exists x \forall a \exists y.\ \phi' \wedge (x \vee a \vee y)$.

Given D^{triv} from prefix: a is irreducible in $(x \vee a \vee y)$ since $a \prec y$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec y$: a is reducible in $(x \vee a \vee y)$, yielding $(x \vee y)$.

CR and Unit Literals: unit literal rule applies CR implicitly.

Practical Effect: expecting shorter learnt constraints when using $D \subseteq D^{\text{triv}}$.

D-Aware Constraint Resolution

Constraint Resolution: Q-resolution for clauses.

- Combining propositional resolution with constraint reduction (CR).
- Learning: heuristically add Q-resolvents to cut off parts of search space.

Definition (Q-resolution for Clauses)

Let C_1, C_2 be clauses with $v \in L_\exists(C_1), \overline{v} \in L_\exists(C_2)$.

1. $C' := (CR(C_1) \cup CR(C_2)) \setminus \{v, \overline{v}\}$.
2. If $\{x, \overline{x}\} \subseteq C'$ for some variable x then no Q-resolvent exists.
3. Otherwise, Q-resolvent $C := CR(C')$ of C_1 and C_2 on v: $\{C_1, C_2\} \vdash_v C$.

Example

$$\exists x \forall a \exists y,z.\ \phi' \wedge \overbrace{(x \vee a \vee y \vee z)}^{C_1} \wedge \overbrace{(x \vee a \vee y \vee \overline{z})}^{C_2} \wedge \overbrace{(x \vee \overline{a} \vee \overline{y} \vee z)}^{C_3}.$$

Given D^{triv} from prefix: $\{C_1, C_2\} \vdash_z (x \vee a \vee y)$, but $\{(x \vee a \vee y), C_3\} \not\vdash_y$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec y$:

$\{C_1, C_2\} \vdash_z (x \vee y)$, and $\{(x \vee y), C_3\} \vdash_y (x \vee \overline{a} \vee z)$.

Practical Effect: enabling "blocked" resolution steps when using $D \subseteq D^{\text{triv}}$.

Constraint Resolution: Q-resolution for clauses.

- Combining propositional resolution with constraint reduction (CR).
- Learning: heuristically add Q-resolvents to cut off parts of search space.

Definition (Q-resolution for Clauses)

Let C_1, C_2 be clauses with $v \in L_\exists(C_1), \overline{v} \in L_\exists(C_2)$.

1. $C' := (CR(C_1) \cup CR(C_2)) \setminus \{v, \overline{v}\}$.
2. If $\{x, \overline{x}\} \subseteq C'$ for some variable x then no Q-resolvent exists.
3. Otherwise, Q-resolvent $C := CR(C') $ of C_1 and C_2 on v: $\{C_1, C_2\} \vdash_v C$.

Example

$\exists x \forall a \exists y,z.\ \phi' \wedge \overbrace{(x \vee a \vee y \vee z)}^{C_1} \wedge \overbrace{(x \vee a \vee y \vee \overline{z})}^{C_2} \wedge \overbrace{(x \vee \overline{a} \vee \overline{y} \vee z)}^{C_3}$.

Given D^{triv} from prefix: $\{C_1, C_2\} \vdash_z (x \vee a \vee y)$, but $\{(x \vee a \vee y), C_3\} \not\vdash_y$.

Given $D \subseteq D^{triv}$ where $a \not\prec y$:
$\{C_1, C_2\} \vdash_z (x \vee y)$, and $\{(x \vee y), C_3\} \vdash_y (x \vee \overline{a} \vee z)$.

Practical Effect: enabling “blocked” resolution steps when using $D \subseteq D^{triv}$.

D-Aware Constraint Resolution

Constraint Resolution: Q-resolution for clauses.

- Combining propositional resolution with constraint reduction (CR).
- Learning: heuristically add Q-resolvents to cut off parts of search space.

Definition (Q-resolution for Clauses)

Let C_1, C_2 be clauses with $v \in L_\exists(C_1), \overline{v} \in L_\exists(C_2)$.

1. $C' := (CR(C_1) \cup CR(C_2)) \setminus \{v, \overline{v}\}$.
2. If $\{x, \overline{x}\} \subseteq C'$ for some variable x then no Q-resolvent exists.
3. Otherwise, Q-resolvent $C := CR(C')$ of C_1 and C_2 on v: $\{C_1, C_2\} \vdash_v C$.

Example

$$\exists x \forall a \exists y,z.\ \phi' \wedge \overbrace{(x \vee a \vee y \vee z)}^{C_1} \wedge \overbrace{(x \vee a \vee y \vee \overline{z})}^{C_2} \wedge \overbrace{(x \vee \overline{a} \vee \overline{y} \vee z)}^{C_3}.$$

Given D^{triv} from prefix: $\{C_1, C_2\} \vdash_z (x \vee a \vee y)$, but $\{(x \vee a \vee y), C_3\} \not\vdash_y$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec y$:
$\{C_1, C_2\} \vdash_z (x \vee y)$, and $\{(x \vee y), C_3\} \vdash_y (x \vee \overline{a} \vee z)$.

Practical Effect: enabling "blocked" resolution steps when using $D \subseteq D^{\text{triv}}$.

D-Aware Constraint Resolution

Constraint Resolution: Q-resolution for clauses.

- Combining propositional resolution with constraint reduction (CR).
- Learning: heuristically add Q-resolvents to cut off parts of search space.

Definition (Q-resolution for Clauses)

Let C_1, C_2 be clauses with $v \in L_\exists(C_1), \overline{v} \in L_\exists(C_2)$.

1. $C' := (CR(C_1) \cup CR(C_2)) \setminus \{v, \overline{v}\}$.
2. If $\{x, \overline{x}\} \subseteq C'$ for some variable x then no Q-resolvent exists.
3. Otherwise, Q-resolvent $C := CR(C')$ of C_1 and C_2 on v: $\{C_1, C_2\} \vdash_v C$.

Example

$\exists x \forall a \exists y, z.\ \phi' \wedge \overbrace{(x \vee a \vee y \vee z)}^{C_1} \wedge \overbrace{(x \vee a \vee y \vee \overline{z})}^{C_2} \wedge \overbrace{(x \vee \overline{a} \vee \overline{y} \vee z)}^{C_3}$.

Given D^{triv} from prefix: $\{C_1, C_2\} \vdash_z (x \vee a \vee y)$, but $\{(x \vee a \vee y), C_3\} \not\vdash_y$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec y$:

$\{C_1, C_2\} \vdash_z (x \vee y)$, and $\{(x \vee y), C_3\} \vdash_y (x \vee \overline{a} \vee z)$.

Practical Effect: enabling "blocked" resolution steps when using $D \subseteq D^{\text{triv}}$.

D-Aware Constraint Resolution

Constraint Resolution: Q-resolution for clauses.

- Combining propositional resolution with constraint reduction (CR).
- Learning: heuristically add Q-resolvents to cut off parts of search space.

Definition (Q-resolution for Clauses)

Let C_1, C_2 be clauses with $v \in L_\exists(C_1), \overline{v} \in L_\exists(C_2)$.

1. $C' := (CR(C_1) \cup CR(C_2)) \setminus \{v, \overline{v}\}$.
2. If $\{x, \overline{x}\} \subseteq C'$ for some variable x then no Q-resolvent exists.
3. Otherwise, Q-resolvent $C := CR(C')$ of C_1 and C_2 on v: $\{C_1, C_2\} \vdash_v C$.

Example

$$\exists x \forall a \exists y,z.\ \phi' \wedge \overbrace{(x \vee a \vee y \vee z)}^{C_1} \wedge \overbrace{(x \vee a \vee y \vee \overline{z})}^{C_2} \wedge \overbrace{(x \vee \overline{a} \vee \overline{y} \vee z)}^{C_3}.$$

Given D^{triv} from prefix: $\{C_1, C_2\} \vdash_z (x \vee a \vee y)$, but $\{(x \vee a \vee y), C_3\} \not\vdash_y$.

Given $D \subseteq D^{\text{triv}}$ where $a \not\prec y$:

$\{C_1, C_2\} \vdash_z (x \vee y)$, and $\{(x \vee y), C_3\} \vdash_y (x \vee \overline{a} \vee z)$.

Practical Effect: enabling “blocked” resolution steps when using $D \subseteq D^{\text{triv}}$.

DepQBF: Implementing QDPLL with D^{std}

- Top-ranked solver in QBFEVAL'10.
- Compact representation of D^{std} as dependency-DAG.
- Strategies from SAT solving: restarts, assignment caching,...

	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
QuBE7.0-pre⇒DepQBF	424	254.23	197	48.17	227	23.42
QuBE7.0	414	310.29	187	130.52	227	58.33

QBFEVAL'10 main track (568 formulae). DepQBF uses preprocessor integrated in QuBE7.0.

QBFEVAL'08 (solved only)						
	$D^{triv} \cap D^{tree}$		$D^{triv} \cap D^{std}$		$D^{tree} \cap D^{std}$	
solved	1172		1196		1206	
time	**23.15**	26.68	**23.73**	25.93	25.63	**22.37**
implied/assigned	90.4%	**90.7%**	88.6%	**90.5%**	90.9%	**92.1%**
backtracks	32431	**27938**	34323	**31085**	**25106**	26136
learnt constr. size	157	**99**	150	**96**	102	**95**

Observed effects of $D^{std} \subseteq D^{tree} \subseteq D^{triv}$ in DepQBF. Comparing intersections of solved formulae.

Conclusion

Drawbacks of Prenex CNF:

- Linear quantifier order limits freedom of QBF decision procedures.

Dependency Schemes:

- Expressing variable independence based on QBF semantics.
- From linear to partial orders on variables: increased freedom.

Practical Effects:

- Independence allows (implicit) deletion of literals from clauses.
- Shorter Q-resolvents: more unit clauses.
- (Skipped: similar effects for cube learning).
- Combining QDPLL with D^{std} in DepQBF: efficient despite of overhead.

Open Problems and Future Work:

- Theoretical results related to QDPLL with $D \subseteq D^{\text{triv}}$.
- Applying more powerful dependency schemes than D^{std}.

DepQBF 0.1 is open source: `http://fmv.jku.at/depqbf/`

U. Bubeck and H. Kleine Büning.
Bounded Universal Expansion for Preprocessing QBF.
In J. Marques-Silva and K. A. Sakallah, editors, *SAT*, volume 4501 of *LNCS*, pages 244–257. Springer, 2007.

M. Benedetti.
Quantifier Trees for QBFs.
In F. Bacchus and T. Walsh, editors, *SAT*, volume 3569 of *LNCS*, pages 378–385. Springer, 2005.

A. Biere.
Resolve and Expand.
In H. H. Hoos and D. G. Mitchell, editors, *SAT (Selected Papers)*, volume 3542 of *LNCS*, pages 59–70. Springer, 2004.

H. Kleine Büning, M. Karpinski, and A. Flögel.
Resolution for Quantified Boolean Formulas.
Inf. Comput., 117(1):12–18, 1995.

M. Cadoli, A. Giovanardi, and M. Schaerf.
An Algorithm to Evaluate Quantified Boolean Formulae.
In *AAAI/IAAI*, pages 262–267, 1998.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Learning for Quantified Boolean Logic Satisfiability.

In *AAAI/IAAI*, pages 649–654, 2002.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas.
J. Artif. Intell. Res. (JAIR), 26:371–416, 2006.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Quantifier Structure in Search-Based Procedures for QBFs.
TCAD, 26(3):497–507, 2007.

F. Lonsing and A. Biere.
Integrating Dependency Schemes in Search-Based QBF Solvers.
In Ofer Strichman and Stefan Szeider, editors, *SAT*, volume 6175 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2010.

R. Letz.
Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas.
In U. Egly and C. G. Fermüller, editors, *TABLEAUX*, volume 2381 of *LNCS*, pages 160–175. Springer, 2002.

M. Samer and S. Szeider.
Backdoor Sets of Quantified Boolean Formulas.
Journal of Automated Reasoning (JAR), 42(1):77–97, 2009.

L. Zhang and S. Malik.
Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation.
In P. Van Hentenryck, editor, *CP*, volume 2470 of *LNCS*, pages 200–215. Springer, 2002.

Decision Problems: NP-complete for SAT vs. PSPACE-complete for QBF.

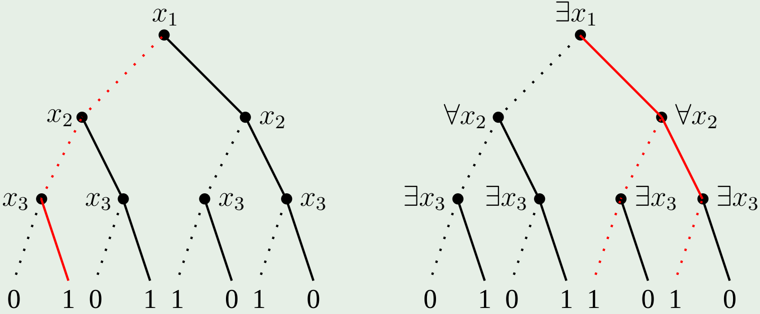
Example

SAT

$\phi(x_1, x_2, x_3)$

QBF

$\exists x_1 \forall x_2 \exists x_3.\ \phi(x_1, x_2, x_3)$

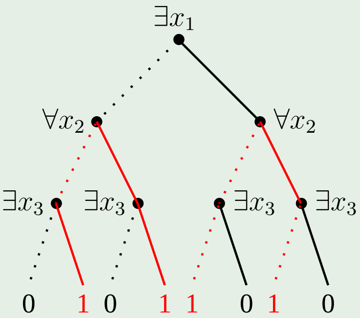


Is there a satisfying assignment? Is there a satisfying assignment **tree**?

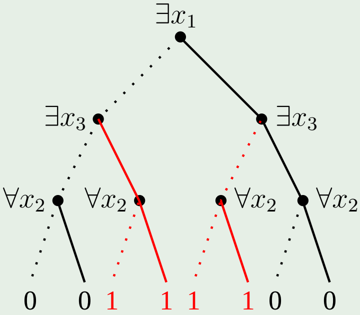
Semantical Definition:

- Given $\forall x \leq \exists y$: y independent from x if value change of x does **not** force value change of y.

Example ($\exists x_1 \forall x_2 \exists x_3.\ \phi(x_1, x_2, x_3)$)



Value of $\exists x_3$ independent from $\forall x_2$.



Can assign $\exists x_3$ **before** $\forall x_2$, although $\forall x_2 \leq \exists x_3$ in prefix order.

- Problem: how to detect independence **efficiently**? (PSPACE-complete!)