SAT-Solving und Anwendungen

QBF Solving

Prof. Dr. Wolfgang Küchlin
Dr. Eray Gençay
Rouven Walter, M.Sc. Informatik

Universität Tübingen

07. Dezember 2017

Quantified Boolean Formulas

Quantifiers

We extend Boolean formulas by allowing variables to be quantified

- $\exists x : (P)$ There exists (an assignment to) x, such that P holds.
- $\forall x : (P)$ For all (assignments to) x, formula P holds.

Quantified variables are called **bound**, unquantified variables are called **free**.

Definition (Quantifier Semantics)

Let Q be a Boolean formula in which x is free.

- $(\exists x : (Q)) \equiv Q|_{x=\top} \vee Q|_{x=\bot}$
- $(\forall x : (Q)) \equiv Q|_{x=\top} \wedge Q|_{x=\bot}$

Definition (QBF and PQBF)

QBF is the set of **fully quantified formulas** (where every variable is bound).
PQBF (partial QBF) is the set of **partially quantified formulas** (where some variables are bound and some are free).

Quantifier Elimination (QE) Problems

From the definition of quantifier semantics, it follows (by induction) that every QBF formula is equivalent to a Boolean constant ($\top$ or $\bot$), and that a PQBF formula is equivalent to a (quantifier-free) formula in the remaining free variables.

The Quantifier Elimination (QE) Problem

The **QE Problem** is: Given a quantified Boolean formula Q, compute an equivalent quantifier-free formula F.

The QBF Problem

Is a given QBF formula equivalent to $\top$ or to $\bot$? (Loosely: is it **true** or **false**?)

SAT and EQBF

SAT can be seen as an existential QBF problem (EQBF): Is a fully existentially quantified formula **true** or **false**?

The PQBF (partial QBF) Problem

Given a PQBF formula Q, compute an equivalent **quantifier-free** formula F.

Simplification Rules and Normal Forms

Example

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

By analogy with Predicate Logic, all quantifiers can be moved to the front of a formula, so that a **quantifier prefix Q** is followed by a quantifier-free **matrix M** in CNF. If Tseitin variables are introduced during CNF conversion of a matrix M into a matrix M', then only $M \equiv_{SAT} M'$, but if we quantify the Tseitin variables $z_1, \ldots, z_n$ existentially, then $Q : M \equiv Q \exists z_1, \ldots, z_n : M'$ holds.

Proposition

$$\exists x : \varphi \vee \exists x : \psi \equiv \exists x : (\varphi \vee \psi)$$

$$\exists x : \varphi \wedge \exists x : \psi \not\equiv \exists x : (\varphi \wedge \psi)$$

$$\forall x : \varphi \vee \forall x : \psi \not\equiv \forall x : (\varphi \vee \psi)$$

$$\forall x : \varphi \wedge \forall x : \psi \equiv \forall x : (\varphi \wedge \psi)$$

Examples

Example (QBF Problems)

$$\exists x \exists y((x \vee y) \wedge (\neg x \vee \neg y)) \equiv ? \quad \text{(answer is } \top)$$

$$\forall x \exists y((x \vee y) \wedge (\neg x \vee \neg y)) \equiv ? \quad \text{(answer is } \top)$$

$$\exists x \forall y((x \vee y) \wedge (\neg x \vee \neg y)) \equiv ? \quad \text{(answer is } \bot)$$

$$\forall x \forall y((x \vee y) \wedge (\neg x \vee \neg y)) \equiv ? \quad \text{(answer is } \bot)$$

Example (PQBF Problems)

$\varphi = \exists x \exists y \ (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)$ (QE result is $\varphi \equiv z$)
$\varphi = \exists x \forall y((x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w))$ (QE result is $\varphi \equiv \neg u \vee w$)

For the general QE Problem, the definition of quantifier semantics already suggests a procedure for eliminating quantified variables (QE Procedure). For the special case of QBF problems, where the result is effectively **true** or **false**, there are also solvers in analogy to SAT solving procedures. Note that $\varphi \equiv \psi$ *iff* $\forall \vec{x} : (\varphi \leftrightarrow \psi) =$ **true**.

PQBF and Quantifier Elimination

Quantifier Elimination (QE)

Given a PQBF formula Q, a *Quantifier Elimination (QE)* method computes an equivalent quantifier free formula F by eliminating from Q all quantified variables.

QE by Substitution

Let Q be a propositional formula in which x is free.

- $(\exists x : Q) \equiv Q|_{x=\top} \vee Q|_{x=\bot}$
- $(\forall x : Q) \equiv Q|_{x=\top} \wedge Q|_{x=\bot}$

Quantified variables can be eliminated from a PQBF formula P by replacing each occurrence of x by $\top$ in one copy of P and by $\bot$ in another copy of P.

QE Method: Substitute and Simplify (SuSi)

Clearly, QE by Substitution "explodes" the formula. However, the formula can be simplified after each substitution, because a variable is replaced by a constant. Sometimes this is enough to keep the explosion in check.

Quantifier Elimination by Substitution and Simplification

For QE by SuSi, the elimination of the quantifiers can be done in any order.

Example (Solving QBF Problems by SuSi QE)

Let $\varphi = \forall x \exists y : ((x \vee y) \wedge (\neg x \vee \neg y))$. We choose to eliminate x first. Then $Q = \exists y : ((x \vee y) \wedge (\neg x \vee \neg y))$ in the elimination schema. Hence

$$\begin{array}{rcl} \varphi & \equiv & Q|_{x=\top} \wedge Q|_{x=\bot} \\ \text{where } Q|_{x=\top} & = & \exists y : ((\top \vee y) \wedge (\neg\top \vee \neg y)) \equiv \exists y : (\top \wedge \neg y) \equiv \exists y : (\neg y) \\ \text{and } Q|_{x=\bot} & = & \exists y : ((\bot \vee y) \wedge (\neg\bot \vee \neg y) \equiv \exists y : (y \wedge \top) \equiv \exists y : (y) \\ \text{Hence } \varphi & \equiv & \exists y : (\neg y) \wedge \exists y : (y) \equiv \top \end{array}$$

Example (Solving QBF Problems by SuSi QE)

Let $\varphi = \exists x \forall y : ((x \vee y) \wedge (\neg x \vee \neg y))$. We choose to eliminate y first. Then $Q = ((x \vee y) \wedge (\neg x \vee \neg y))$ and $\varphi \equiv \exists x : [Q|_{x=\top} \wedge Q|_{x=\bot}]$,

$$\begin{array}{rcl} \text{where } Q|_{y=\top} & = & ((x \vee \top) \wedge (\neg x \vee \neg\top)) \equiv (\top \wedge \neg x) \equiv (\neg x) \\ \text{and } Q|_{y=\bot} & = & ((x \vee \bot) \wedge (\neg x \vee \neg\bot) \equiv (x \wedge \top) \equiv (x) \\ \text{Hence } \varphi & \equiv & \exists x : (\neg x \wedge x) \equiv \bot \end{array}$$

Quantifier Elimination by Substitution and Simplification

Example (Solving a PQBF Problem by SuSi QE)

$$
\begin{aligned}
\varphi &= \exists x \forall y : ((x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w)) \\
QE(\varphi, x) = \varphi_x &= (Q|_{x=\top} \vee Q|_{x=\bot}) \\
\text{where } \ Q &= \forall y : ((x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w)) \\
\text{with } \ Q|_{x=\top} &\equiv \forall y : ((\top \vee y \vee \neg u) \wedge (\neg \top \vee \neg y \vee w)) \equiv \forall y : (\neg y \vee w) \\
\text{and } \ Q|_{x=\bot} &\equiv \forall y : ((\bot \vee y \vee \neg u) \wedge (\neg \bot \vee \neg y \vee w)) \equiv \forall y : (y \vee \neg u) \\
\text{Hence } \ \varphi_x &\equiv [\forall y : (\neg y \vee w)] \vee [\forall y : (y \vee \neg u)] =: \varphi_{x_1} \vee \varphi_{x_2} \\
QE(\varphi_x, y) = \varphi_{xy} &= QE(\varphi_{x_1}, y) \vee QE(\varphi_{x_2}, y) \\
\text{with } \ QE(\varphi_{x_1}, y) &= [(\neg \top \vee w)] \wedge [(\neg \bot \vee w)] \equiv w \\
\text{and } \ QE(\varphi_{x_2}, y) &= [(\top \vee \neg u)] \wedge [(\bot \vee \neg u)] \equiv \neg u \\
\text{Hence } \ QE(\varphi_x, y) &\equiv (w \vee \neg u) \equiv \varphi
\end{aligned}
$$

Quantifier Elimination by Substitution and Simplification

Example (Solving an existential PQBF Problem by SuSi QE)

$$
\begin{aligned}
\varphi &= \exists y \exists x : ((x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
QE(\varphi, x) = \varphi_x &= \exists y : (Q|_{x=\top} \vee Q|_{x=\bot}) \\
\text{where } \; Q &= ((x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
Q|_{x=\top} &\equiv ((\top \vee \neg w) \wedge (\neg \top \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
&\equiv ((z) \wedge (\neg z \vee y) \wedge (w \vee z)) \equiv z \wedge (\neg z \vee y) \equiv (z \wedge y) \\
Q|_{x=\bot} &\equiv ((\bot \vee \neg w) \wedge (\neg \bot \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
&\equiv ((\neg w) \wedge (\neg z \vee y) \wedge (w \vee z)) \equiv (\neg z \vee y) \wedge (\neg w \wedge z) \\
&\equiv (y \wedge z) \wedge \neg w. \\
\text{Hence } \; \varphi_x &\equiv \exists y : ([z \wedge y] \vee [(y \wedge z) \wedge \neg w]) \equiv \exists y : (z \wedge y). \\
QE(\varphi_x, y) = \varphi_{xy} &\equiv [(z \wedge \top)] \vee [(z \wedge \bot)] \\
&\equiv z
\end{aligned}
$$

The important case where all quantifiers are existential admits many other elimination procedures, such as Variable Elimination and Model Enumeration and Projection.

Quantifier Elimination for E(P)QBF

EPQBF formulas are PQBF formulas with only existential quantifiers. Already Davis and Putnam (1960) used a QE procedure called *variable elimination (VE)*.

Existential QE by Variable Elimination

Let $\exists x : Q$ be an EPQBF formula in normal form. Transform Q into Q' as follows:

- If x occurs only in one polarity in Q, then remove all clauses from Q which contain x.
- Otherwise let P be all clauses in which x occurs positive, let N be all clauses in which x occurs negative, and let R be all clauses in which x does not occur. Finally, let S be the set of all resolvents between clauses in P and clauses in N, and let $Q' = R \cup S$. Then $(\exists x : Q) \equiv Q'$.

Note that x no longer occurs in Q', because the "parent clauses" in P and N are eliminated.

In resolution theorem proving, the parent clauses of any resolvent r are retained, so that $Q \equiv Q \cup \{r\}$. In VE, *all* resolvents must be produced, but the parent clauses are then deleted. This maintains only satisfiability-equivalence between Q and Q', but $(\exists x : Q) \equiv Q'$.

EPQBF Quantifier Elimination by Variable Elimination

Example (Variable Elimination)

$$
\begin{aligned}
\varphi &= \exists y \exists x : ((x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
P_x &= \{(x \vee \neg w)\} \\
N_x &= \{(\neg x \vee z)\} \\
R_x &= \{(\neg z \vee y) \wedge (w \vee z)\} \\
S_x &= \{(\neg w \vee z)\} \\
VE(\varphi, x) = \varphi_x = S_x \cup R_x &= \exists y : ((\neg w \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
VE(\varphi_x, y) = \varphi_{xy} &= (\neg w \vee z) \wedge (w \vee z) \\
&\equiv z
\end{aligned}
$$

Example (Variable Elimination)

$$
\begin{aligned}
\varphi &= \exists w \exists x : (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z) \\
\text{Know from above } \varphi_x &= \exists w : ((\neg w \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)) \\
VE(\varphi_x, w) = \varphi_{xw} &= z \wedge (\neg z \vee y) \equiv (z \wedge y) \equiv \varphi
\end{aligned}
$$

EPQBF QE by Model Enumeration and Projection

Existential QE by Model Elimination and Projection

Let $E = \exists x : Q$ be an EPQBF formula. Let $D = t_1 \vee \cdots \vee t_n$ be any DNF of Q. Let $D' = t'_1 \vee \cdots \vee t'_n$ consist of the terms of D where all occurrences of x and $\neg x$ are removed. (This is a **projection** of D onto the variables of Q.) Then $D' \equiv E$.

Note that D could be the set of all models of Q or any implicant cover of Q. Now let $t = \{x\} \cup t'$ or $\bar{t} = \{\bar{x}\} \cup t'$ be any term in D, i.e. t and $\bar{t}$, respectively, are implicants of Q. Then t' is an implicant of $E = \exists x : Q$. Moreover, since $D = t_1 \vee \cdots \vee t_n$ is an implicant cover of Q, also $D' = t'_1 \vee \cdots \vee t'_n \equiv \exists x : Q$. For if e is any implicant of E, then either xe or $\bar{x}e$ is an implicant of Q. W.l.o.g., let this be xe. Since D is an implicant cover of Q, there exists a t_i which is contained in xe. Therefore, t'_i is contained in e. Hence any e is covered by D'.

Example (Model Elimination and Projection)

Let $\varphi = \exists w : ((\neg w \vee z) \wedge (\neg z \vee y) \wedge (w \vee z))$. We know from above that $\varphi \equiv z \wedge y$. Therefore, $M = \{\{z, y, w\}, \{z, y, \neg w\}\}$ is the set of all models of φ. Projection of M onto y and z (the free variables of φ) yields $M' = \{\{z, y\}\}$.

Das QBF Problem

QBF

Ist eine **vollständig quantifizierte** Formel in Aussagenlogik **wahr** oder **falsch**? (Spezialfall SAT: Eine **vollständig existenziell quantifizierte** Formel)

Beispiel (QBF Probleme)

$$\forall x \exists y ((x \vee y) \wedge (\neg x \vee \neg y)) \equiv \mathbf{T}$$

$$\forall x \forall y ((x \vee y) \wedge (\neg x \vee \neg y)) \equiv \mathbf{F}$$

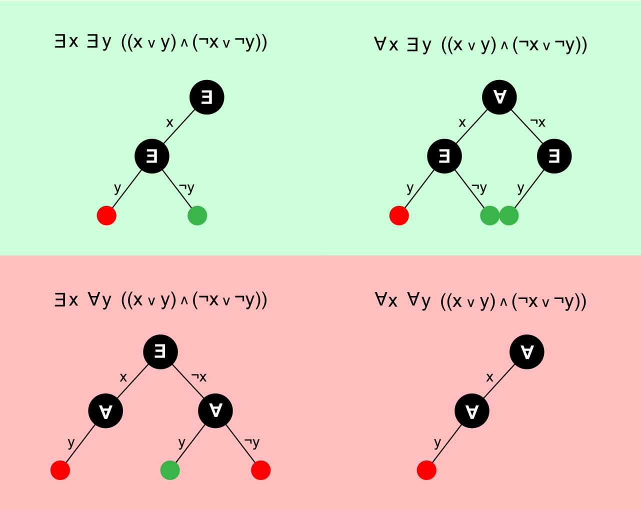
Spezialfall SAT

Ist eine gegebene **vollständig existenziell quantifizierte** Formel der Aussagenlogik **wahr** (**erfüllbar**) oder **falsch** (**nicht erfüllbar**)? Als QBF Formeln: $\exists x_1, \ldots, x_n : (P) \equiv \top$ bzw. $\exists x_1, \ldots, x_n : (P) \equiv \bot$?

Bemerkung: Heutzutage hat sich die Bezeichnung QBF durchgesetzt, man findet jedoch auch in vielen Publikationen noch die Bezeichnung QSAT

Visualisierung von QBF

- 2 verschiedene Knotentypen: Existenzknoten und Allknoten
- Existenzknoten benötigen 1 erfüllenden Ast, Allknoten benötigen 2 erfüllende Äste



Komplexität von QBF – 1

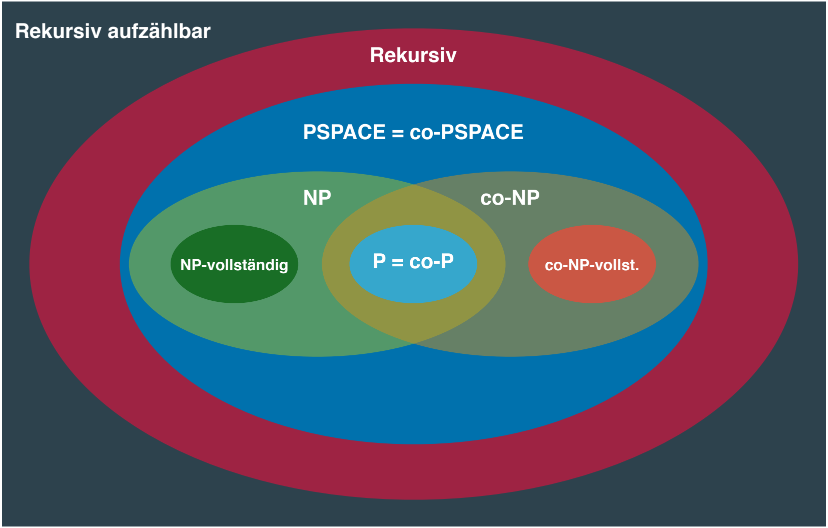
Im SAT Fall:

- NP-vollständig (Nichtdeterministische Turingmaschine, Polynomiale Zeit)
- Eine erfüllende Belegung kann geraten werden und in Polynomialzeit überprüft werden

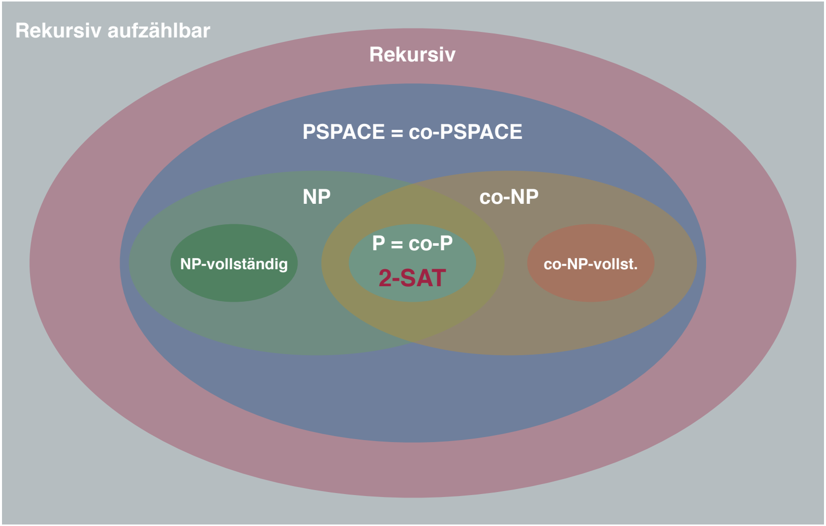
Im QBF Fall:

- PSPACE-vollständig (Deterministische Turingmaschine, Polynomialer Platz)
- Es kann nicht mehr einfach eine Belegung angegeben werden, sondern man muss für jede mögliche Belegung der allquantifizierten Variablen eine Belegung der existenzquantifizierten Variablen angeben

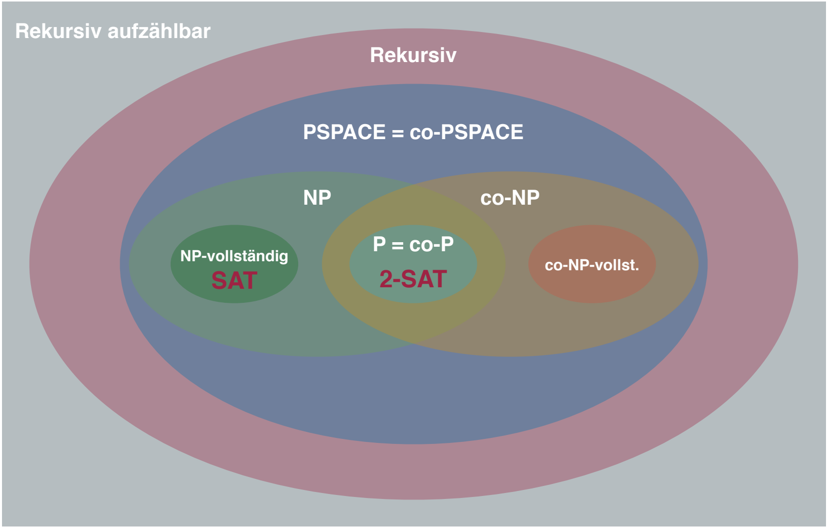
Komplexität von QBF – 2



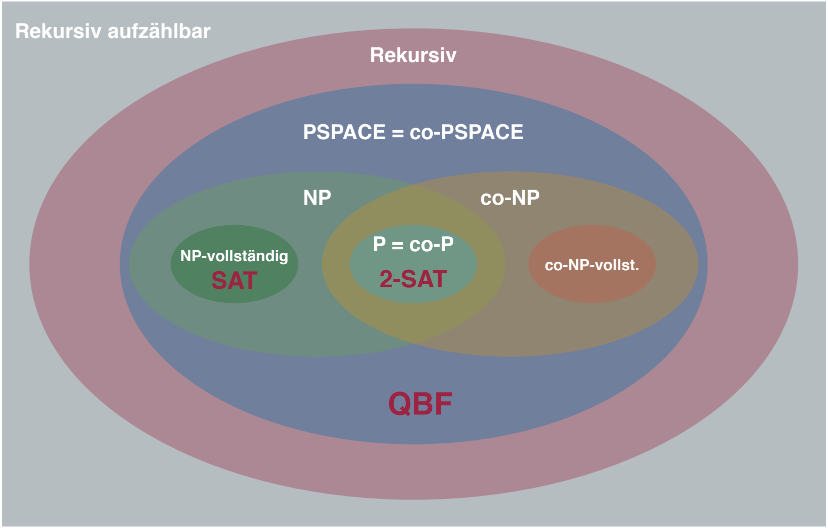
Komplexität von QBF – 2



Komplexität von QBF – 2



Komplexität von QBF – 2



Formales

Pränexe Normal Form (PNF)

Eine quantifizierte Boolesche Formel φ ist in PNF wenn sie von der Form

$$Q_1 x_1, ..., Q_n x_n \psi$$

mit $Q_i \in \{\exists, \forall\}$ ist und ψ quantorenfrei ist.

Jede Formel kann in PNF gebracht werden.

Freie / Gebundene Variablen

Eine Variable x kommt frei in einer Formel φ vor, wenn sie nicht quantifiziert ist. Ist sie quantifiziert, so kommt sie gebunden vor.

Im QBF Fall gibt es nur gebundene Variablen (Formel ist voll quantifiziert)

Vergleich der Algorithmen

SAT vs. QBF Algorithmus

Algorithm 1: SAT

```
level := 0;
while true do
    unitPropgation();
    if a conflict is reached then
        level := analyseConflict();
        if level = 0 then
            return false
        backtrack(level);
    else
        if formula is satisfied then
            return true
        level := level + 1;
        choose an unassigned x ∈ var(P);
        α := α ∪ [x ↦ 0];
```

Algorithm 2: QBF

```
level := 0;
while true do
    unitPropgation();
    if a conflict is reached then
        level := analyseConflict();
        if level = 0 then
            return false
        backtrack(level);
    else
        if formula is satisfied then
            level := analyseSAT();
            if level = 0 then
                return true
            backtrack(level)
        else
            level := level + 1;
            choose an unassigned x ∈ var(P)
              (wrt. the q-level);
            α := α ∪ [x ↦ 0];
```

Auswahlheuristik

- Prinzipiell die selben Heuristiken wie im SAT Fall
- Müssen Quantifikations Level beachten
- Quantifikations Level wird mit jedem Quantorenwechsel erhöht

Beispiel (Quantifikations Level)

$$\underbrace{\exists x\ \exists y}_{\text{Level 1}}\ \underbrace{\forall z\ \forall w}_{\text{Level 2}}\ \underbrace{\exists u}_{\text{Level 3}}\ (x \lor y \lor z \lor w \lor u)$$

- Heuristik muss von außen nach innen voranschreiten (d.h. von Level 1 aufwärts)
- Solange noch Variablen auf einem Level n nicht belegt sind, darf keine Variable auf einem Level $> n$ gewählt werden (gilt nicht für UP)
- Worst case: $\exists x_1 \forall x_2 \exists x_3 \forall x_4 ... \varphi$ (keine Wahlmöglichkeiten)

Neue Regel für Empty Clauses

Im SAT Fall: Eine Klausel ist unerfüllbar (empty clause) wenn sie noch nicht erfüllt ist und alle Variablen belegt sind.

Neue Regel im QBF Fall:

- $E(C)$ Literale mit existenzquantifizierten Variablen einer Klausel C
- $U(C)$ Literale mit allquantifizierten Variablen einer Klausel C
- $ql(x)$ Quantifikationslevel einer Variable x

Empty Clause

Eine Klausel C ist unerfüllbar (empty clause), wenn

1. für alle $e \in E(C)$ gilt $\nu(e) = \bot$
2. für alle $u \in U(C)$ gilt $\nu(u) \neq \top$

Beispiel (Empty Clause)

a, b, c sind existenzquantifiziert, x, y sind allquantifiziert,

$[a \mapsto \top, b \mapsto \bot, c \mapsto \top, x \mapsto \bot]$

- $(\neg a \vee b \vee \neg c \vee x \vee y)$ ist unerfüllbar (für $[x \mapsto \bot, y \mapsto \bot]$ nicht erfüllbar)

Neue Regel für Unit Clauses

Regel im SAT Fall: Eine Klausel ist unit, wenn sie noch nicht erfüllt ist und genau eine Variable nicht belegt (*nil*) ist. (Es gibt eine eindeutige letzte Belegung für die Erfüllung.)

Neue Regel im QBF Fall:

Unit Clause

Eine Klausel C ist unit, wenn

1. ein $e \in E(C)$ existiert, so dass gilt $\nu(e) = nil$.
2. Für jedes $e' \in E(C)$ mit $e' \neq e$ gilt, dass $\nu(e') = \bot$.
3. Für alle $u \in U(C)$ gilt $\nu(u) \neq \top$.
4. Für alle $u \in U(C)$ gilt: Falls $\nu(u) = nil$, dann $ql(u) > ql(e)$.

Beispiel (Unit Clause)

a, b, c sind existenzquantifiziert, x, y sind allquantifiziert

$[a \mapsto \bot, c \mapsto \top, x \mapsto \bot]$, für die nächste durch UP implizierte Variable b gilt $ql(b) = 5$

- $(a_{(2)} \vee b_{(5)} \vee \neg c_{(3)} \vee x_{(1)} \vee y_{(6)})$ ist unit (denn nur $[b_{(5)} \mapsto \top]$ rettet $[y_{(6)} \mapsto \bot]$).
- $(a_{(2)} \vee b_{(5)} \vee \neg c_{(3)} \vee x_{(4)} \vee y_{(1)})$ ist nicht unit (denn für $[y_{(1)} \mapsto \top]$ ist $[b_{(5)} \mapsto \top]$ nicht zwingend).

Verarbeitung allquantifizierter Variablen

- Für allquantifizierte Variablen müssen beide Belegungen getestet werden
- Jede allquantifizierte Variable muss geflipped werden
- Jede allquantifizierte Variable bekommt ein flag “flipped”
- Wird x zum ersten Mal belegt, so wird das flag auf `false` gesetzt
- Wird der Wert von x geflipped, wird das flag auf `true` gesetzt

Backtracking im erfüllenden Fall

Suche die letzte allquantifizierte Variable x, deren flag `false` ist, mache ein Backtracking zum level von x und flippe den Wert von x.

Lernen in QBF

Funktioniert im Prinzip wie bei SAT, aber mit 2 Besonderheiten:

- Es kann zu Long Distance Resolutions kommen
- Modifiziertes Stopp-Kriterium für 1UIP

Long Distance Resolution

- Resolution bei SAT: Nur über 1 Literal, das sich im Vorzeichen unterscheidet (Distance 1)
- Bei QBF: Mehrere Literale können sich im Vorzeichen unterscheiden

Grund: Allquantifizierte Variablen, die noch nicht belegt sind. Das Vorzeichen der betreffenden Literale ist im Grunde willkürlich, da beide Polaritäten nacheinander belegt werden müssen.

Beispiel (Long Distance Resolution)

$$\exists a, .., \exists b \forall x, y \exists c : (a_{(1)} \vee b_{(3)} \vee x_{(4)} \vee y_{(4)} \vee c_{(5)}) \wedge (a_{(1)} \vee \neg b_{(3)} \vee \neg x_{(4)} \vee \neg y_{(4)} \vee d_{(5)})$$

Resolution über b ist eine Tautologie:

$$\exists a, .., \exists b \forall x, y \exists c : (a_{(1)} \vee x_{(4)} \vee \neg x_{(4)} \vee y_{(4)} \vee \neg y_{(4)} \vee c_{(5)} \vee d_{(5)})$$

Aber gelernte Klauseln haben (für SAT und QBF) eigentlich nur zwei Zwecke:

- Erkennen von Konfliktsituationen
- Erkennen von Möglichkeiten zur Unit-Propagation

$\rightarrow$ Beides funktioniert mit obigen Tautologieklauseln

Stopp-Kriterium für 1UIP

Ziel von 1UIP:

- neu gelernte Klausel soll nach Backtracking unit sein

Erinnerung: geändertes Kriterium für Unit Clauses

Unit Clause

Eine Klausel C ist unit, wenn

1. ein $e \in E(C)$ existiert, so dass gilt $\nu(e) = nil$.
2. Für jedes $e' \in E(C)$ mit $e' \neq e$ gilt, dass $\nu(e') = \perp$.
3. Für alle $u \in U(C)$ gilt $\nu(u) \neq \top$.
4. Für alle $u \in U(C)$ gilt: Falls $\nu(u) = nil$, dann $ql(u) > ql(e)$.

Daraus folgt das Stopp-Kriterium für 1UIP:

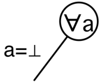
Stopp-Kriterium für 1UIP

1. Genau eine existenzquantifizierte Variable e ist auf höchstem Decision Level ℓ
2. Auf Decision Level ℓ ist die Entscheidungsvariable existenzquantifiziert
3. Alle allquantifizierten Variablen u mit $ql(u) < ql(e)$ werden zu 0 evaluiert auf einem Decision Level $<$ dem von e

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

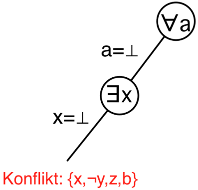


Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision

Ein Beispiel für QBF

∀a∃x∃y∀b∃z

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

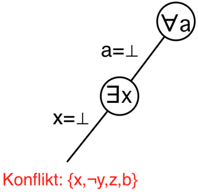


Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
2	x	∃(2)	⊥	decision
	z	∃(4)	⊥	{x,¬z,a}
	y	∃(2)	⊤	{x,y,z,¬b}

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

$\{x,y,z,\neg b\}$
$\{x,\neg z,a\}$
$\{\neg x,y,a\}$
$\{x,\neg y,z,b\}$



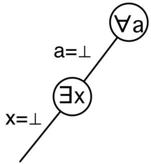
Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ F	$\perp$	decision
2	x	$\exists(2)$	$\perp$	decision
	z	$\exists(4)$	$\perp$	$\{x,\neg z,a\}$
	y	$\exists(2)$	$\top$	$\{x,y,z,\neg b\}$

$\{x_2,\neg y_2,z_2,b_{na}\}$ $\{x_2,y_2,z_2,\neg b_{na}\}$

Ein Beispiel für QBF

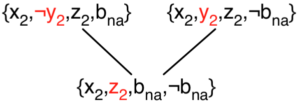
$\forall a \exists x \exists y \forall b \exists z$

$\{x, y, z, \neg b\}$
$\{x, \neg z, a\}$
$\{\neg x, y, a\}$
$\{x, \neg y, z, b\}$



Konflikt: $\{x, \neg y, z, b\}$

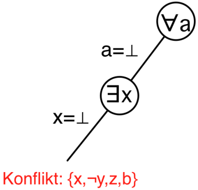
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
2	x	∃(2)	⊥	decision
	z	∃(4)	⊥	{x,¬z,a}
	y	∃(2)	⊤	{x,y,z,¬b}



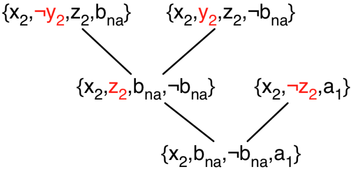
Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

$\{x,y,z,\neg b\}$
$\{x,\neg z,a\}$
$\{\neg x,y,a\}$
$\{x,\neg y,z,b\}$



Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ F	$\perp$	decision
2	x	$\exists(2)$	$\perp$	decision
	z	$\exists(4)$	$\perp$	$\{x,\neg z,a\}$
	y	$\exists(2)$	$\top$	$\{x,y,z,\neg b\}$

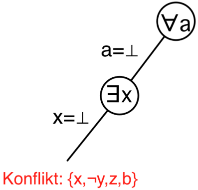


Ein Beispiel für QBF

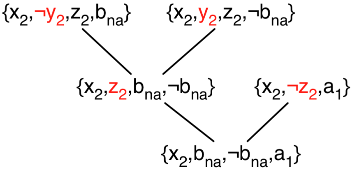
$\forall a \exists x \exists y \forall b \exists z$

$\{x,y,z,\neg b\}$
$\{x,\neg z,a\}$
$\{\neg x,y,a\}$
$\{x,\neg y,z,b\}$

$\{x,b,\neg b,a\}$



Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ F	$\perp$	decision
2	x	$\exists(2)$	$\perp$	decision
	z	$\exists(4)$	$\perp$	$\{x,\neg z,a\}$
	y	$\exists(2)$	$\top$	$\{x,y,z,\neg b\}$

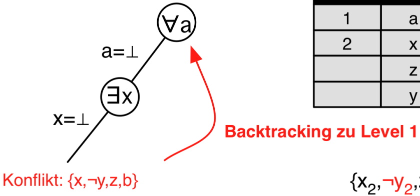


Ein Beispiel für QBF

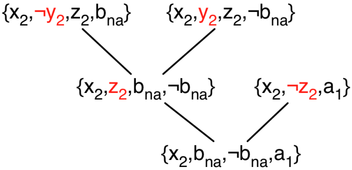
$\forall a \exists x \exists y \forall b \exists z$

$\{x,y,z,\neg b\}$
$\{x,\neg z,a\}$
$\{\neg x,y,a\}$
$\{x,\neg y,z,b\}$

$\{x,b,\neg b,a\}$



Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
2	x	∃(2)	⊥	decision
	z	∃(4)	⊥	{x,¬z,a}
	y	∃(2)	⊤	{x,y,z,¬b}

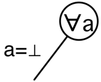


Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



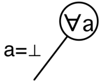
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



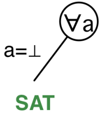
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
	x	∃(2)	⊤	{x,b,¬b,a}
	y	∃(2)	⊤	{¬x,y,a}

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



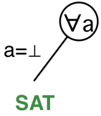
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
	x	∃(2)	T	{x,b,¬b,a}
	y	∃(2)	T	{¬x,y,a}

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ F	$\perp$	decision
	x	$\exists(2)$	$\top$	{x,b,¬b,a}
	y	$\exists(2)$	$\top$	{¬x,y,a}

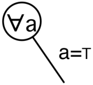
Backtracking zur letzten Variable mit Flag F

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



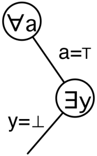
Level	Var	Quant	Wert	Grund
1	a	∀(1) T	T	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



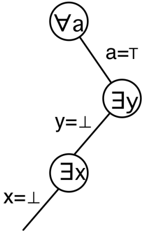
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



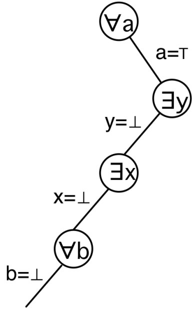
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

$\{x,y,z,\neg b\}$
$\{x,\neg z,a\}$
$\{\neg x,y,a\}$
$\{x,\neg y,z,b\}$

$\{x,b,\neg b,a\}$



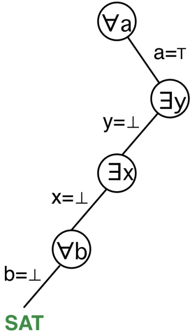
Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ T	$\top$	decision
2	y	$\exists(2)$	$\bot$	decision
3	x	$\exists(2)$	$\bot$	decision
4	b	$\forall(3)$ F	$\bot$	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

$\{x,y,z,\neg b\}$
$\{x,\neg z,a\}$
$\{\neg x,y,a\}$
$\{x,\neg y,z,b\}$

$\{x,b,\neg b,a\}$



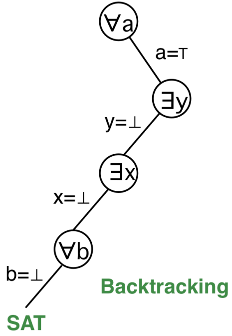
Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ T	$\top$	decision
2	y	$\exists(2)$	$\bot$	decision
3	x	$\exists(2)$	$\bot$	decision
4	b	$\forall(3)$ F	$\bot$	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



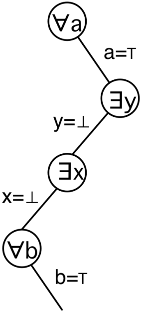
Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ T	$\top$	decision
2	y	$\exists(2)$	$\bot$	decision
3	x	$\exists(2)$	$\bot$	decision
4	b	$\forall(3)$ F	$\bot$	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



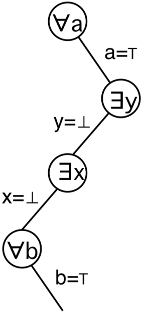
Level	Var	Quant	Wert	Grund
1	a	$\forall(1)$ T	⊤	decision
2	y	$\exists(2)$	⊥	decision
3	x	$\exists(2)$	⊥	decision
4	b	$\forall(3)$ T	⊤	decision

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



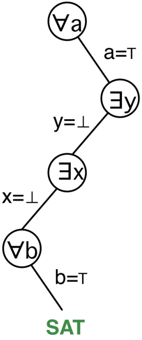
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) ⊤	⊤	decision
	z	∃(4)	⊤	{x,y,z,¬b}

Ein Beispiel für QBF

∀a∃x∃y∀b∃z

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



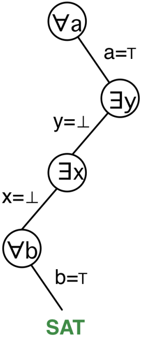
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) ⊤	⊤	decision
	z	∃(4)	⊤	{x,y,z,¬b}

Ein Beispiel für QBF

$\forall a \exists x \exists y \forall b \exists z$

{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) ⊤	⊤	decision
	z	∃(4)	⊤	{x,y,z,¬b}

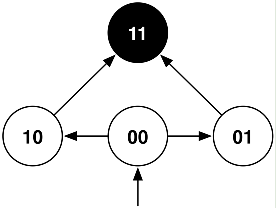
Keine weiteren Flags mit F

→ Result: TRUE

Codierung mit QBF - 1

Beispiel (Endlicher Automat)

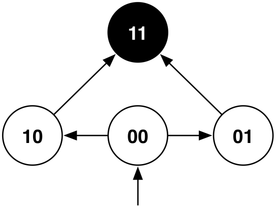
- 4 Zustände: $00, 01, 10, 11$
- Startzustand: 00
- Zustandsübergänge: $00 \rightarrow 01, 00 \rightarrow 10, 10 \rightarrow 11, 01 \rightarrow 11$
- Fehlerzustand: 11



Codierung:

- $s[0]$ und $s[1]$ codieren die beiden Bits des Zustands
- tiefergestellte Zahlen codieren den aktuellen timestep $s[0]_0$

Codierung mit QBF - 2

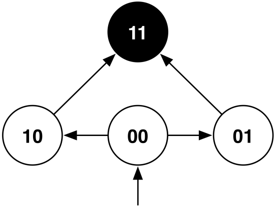


Fragestellung: Erreicht man einen Fehlerzustand in einem Schritt?

- Initialer Zustand: $I(0) = \neg s[0]_0 \wedge \neg s[1]_0$
- Übergangsfunktion: $T(0,1) = (\neg s[0]_0 \wedge \neg s[1]_0 \wedge \neg s[0]_1 \wedge s[1]_1) \vee (\neg s[0]_0 \wedge \neg s[1]_0 \wedge s[0]_1 \wedge \neg s[1]_1) \vee (\neg s[0]_0 \wedge s[1]_0 \wedge s[0]_1 \wedge s[1]_1) \vee (s[0]_0 \wedge \neg s[1]_0 \wedge s[0]_1 \wedge s[1]_1)$
- Fehlerzustand nach einem Schritt: $B(1) = s[0]_1 \wedge s[1]_1$

Checke $I(0) \wedge T(0,1) \wedge B(1)$, wenn erfüllbar, dann ist die Belegung ein Pfad von 00 zu 11, ansonsten gibt es keinen Pfad der Länge 1 zum Fehler

Codierung mit QBF - 3

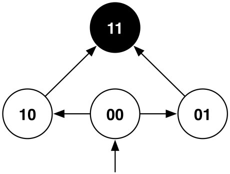


Fragestellung: Erreicht man einen Fehlerzustand in zwei Schritten?

- Initialer Zustand: $I(0) = \neg s[0]_0 \wedge \neg s[1]_0$
- Übergangsfunktion: $T(0,1) = (\neg s[0]_0 \wedge \neg s[1]_0 \wedge \neg s[0]_1 \wedge s[1]_1) \vee (\neg s[0]_0 \wedge \neg s[1]_0 \wedge s[0]_1 \wedge \neg s[1]_1) \vee (\neg s[0]_0 \wedge s[1]_0 \wedge s[0]_1 \wedge s[1]_1) \vee (s[0]_0 \wedge \neg s[1]_0 \wedge s[0]_1 \wedge s[1]_1)$

  $T(1,2) = (\neg s[0]_1 \wedge \neg s[1]_1 \wedge \neg s[0]_2 \wedge s[1]_2) \vee (\neg s[0]_1 \wedge \neg s[1]_1 \wedge s[0]_2 \wedge \neg s[1]_2) \vee (\neg s[0]_1 \wedge s[1]_1 \wedge s[0]_2 \wedge s[1]_2) \vee (s[0]_1 \wedge \neg s[1]_1 \wedge s[0]_2 \wedge s[1]_2)$
- Fehlerzustand nach zwei Schritten: $B(2) = s[0]_2 \wedge s[1]_2$
- $I(0) \wedge T(0,1) \wedge T(1,2) \wedge B(2)$ erfüllbar ($00 - 01 - 11$ oder $00 - 10 - 11$ als Pfad zum Fehler)
- **Aber:** zwei Kopien der Übergangsfunktion

Codierung mit QBF - 4



Codierung des Problems mit QBF:

- $T = (\neg u[0] \wedge \neg u[1] \wedge \neg v[0] \wedge v[1]) \vee (\neg u[0] \wedge \neg u[1] \wedge v[0] \wedge \neg v[1]) \vee (\neg u[0] \wedge u[1] \wedge v[0] \wedge v[1]) \vee (u[0] \wedge \neg u[1] \wedge v[0] \wedge v[1])$

Formel für k Schritte

$\exists s[0]_0 \ldots \exists s[0]_k \exists s[1]_0 \ldots \exists s[1]_k \forall u[0] \forall u[1] \forall v[0] \forall v[1]$
$I(0) \wedge B(k) \wedge$
$\left(\bigvee_{i=0}^{k-1} \left(u[0] \leftrightarrow s[0]_i \wedge u[1] \leftrightarrow s[1]_i \wedge v[0] \leftrightarrow s[0]_{i+1} \wedge v[1] \leftrightarrow s[1]_{i+1} \right) \rightarrow T \right)$

Immer wenn $u[0], u[1]$ und $v[0], v[1]$ äquivalent der Zustände $s[0]_i, s[1]_i$ und $s[0]_{i+1}, s[1]_{i+1}$ sind, muss die Übergangsformel T gelten. Jetzt nur noch eine Kopie der Übergangsfunktion T

Technisches

QDIMACS Format

- Klauseln wie im CNF Format
- Präambel mit Quantifikation der Variablen:

```
p cnf 10 5
a 2 3 4 0
e 1 5 6 0
...
```

- Ressourcen (Benchmarks, Solver, Literatur): `http://www.qbflib.org/`

Bekannte Solver:

- **Quaffle** (Zhang, Malik), *(Techniken dieser Vorlesung)*
- **Quantor, DepQBF** (Biere)
- **QuBE** (Giunchiglia)
- **SKizzo** (Benedetti)

Erweiterung: PSAT & PQSAT

SAT

Ist eine gegebene Formel in Aussagenlogik **erfüllbar** oder **nicht erfüllbar**?

QBF

Ist eine gegebene **vollständig quantifizierte** Formel in Aussagenlogik **wahr** oder **falsch**?

PSAT (parametric SAT)

Unter welchen **Bedingungen** ist eine Formel in Aussagenlogik erfüllbar, die nur **existentiell quantifizierte oder freie Variablen** besitzt.

PQSAT (parametric QSAT)

Unter welchen **Bedingungen** ist eine Formel in Aussagenlogik mit **beliebiger Quantifizierung** erfüllbar.

PSAT & PQSAT

Beispiel (PSAT)

$\varphi = \exists x \exists y \; (x \vee \neg w) \wedge (\neg x \vee z) \wedge (\neg z \vee y) \wedge (w \vee z)$

$$\mathrm{PSAT}(\varphi) = (\neg w \wedge z) \vee (w \wedge z) \equiv z$$

Beispiel (PQSAT)

$\varphi = \exists x \forall y ((x \vee y \vee \neg u) \wedge (\neg x \vee \neg y \vee w))$

$$\mathrm{PQSAT}(\varphi) = (\neg u \wedge \neg w) \vee (\neg u \wedge w) \vee (u \wedge w) \equiv \neg u \vee w$$

Anwendungen:

- Reachability Analysis im Symbolic Model Checking
- Berechnung von Craig Interpolanten
- Berechnen aller Fehlerpfade im Model Checking
- Counterexample Generalization
- Reparatur von Baubarkeitsaufträgen (Rekonfiguration)
- ...

Lösungsansätze

Ansätze zum Lösen von **PQSAT**:

- Top-Level DPLL für die freien Variablen, der an jedem Blatt wiederum SAT/QBF aufruft (1)

Ansätze zum Lösen von **PSAT**:

- Resolutionsbasierter Ansatz (Clause Distribution) (2)
- SAT-basierter Ansatz (Model Enumeration) (2)
- Knowledge Compilation Ansatz (DNNF) (2)

Forschung am Arbeitsbereich

(1) *Thomas Sturm, Christoph Zengler*: **Parametric Quantified SAT Solving**, ISSAC 2010

(2) *Andreas Kübler, Wolfgang Küchlin, Christoph Zengler*: **New Approaches to Boolean Quantifier Elimination**, (Poster) ISSAC 2011

(3) *Christoph Zengler, Wolfgang Küchlin*: **Boolean Quantifier Elimination for Automotive Configuration – A Case Study**, Formal Methods for Industrial Critical Systems (FMICS), 2013