SAT-Solving und Anwendungen
QBF Solving

Prof. Dr. Wolfgang Küchlin
Dipl. Inform. Christoph Zengler

Universität Tübingen

12. Juni 2012

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Das QBF Problem

SAT

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

QBF

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

Quantoren:

- $\exists x(P)$ Es existiert ein (oder mehrere) x, so dass Aussage P gilt
- $\forall x(P)$ Für alle x gilt Aussage P

Beispiel (QBF Probleme)

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

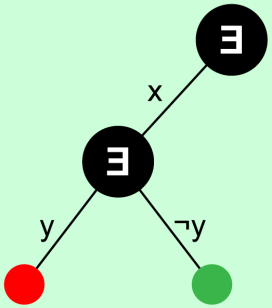
$$\forall x \forall y((x \vee y) \wedge (\neg x \vee \neg y)) = \mathbf{F}$$

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

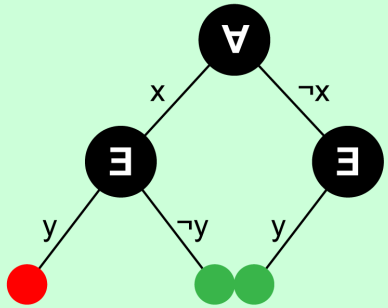
Visualisierung von QBF

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

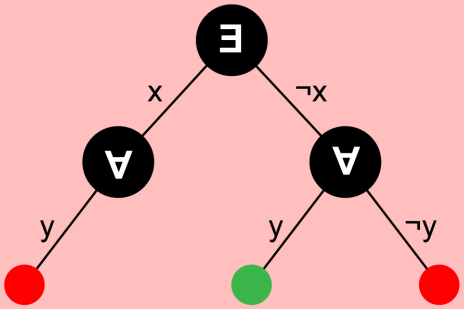
$\exists x\ \exists y\ ((x \vee y) \wedge (\neg x \vee \neg y))$



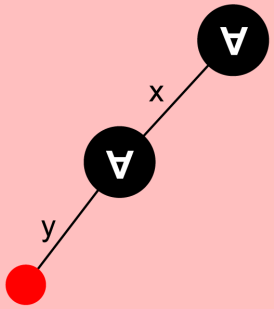
$\forall x\ \exists y\ ((x \vee y) \wedge (\neg x \vee \neg y))$



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



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



Komplexität von QBF - 1

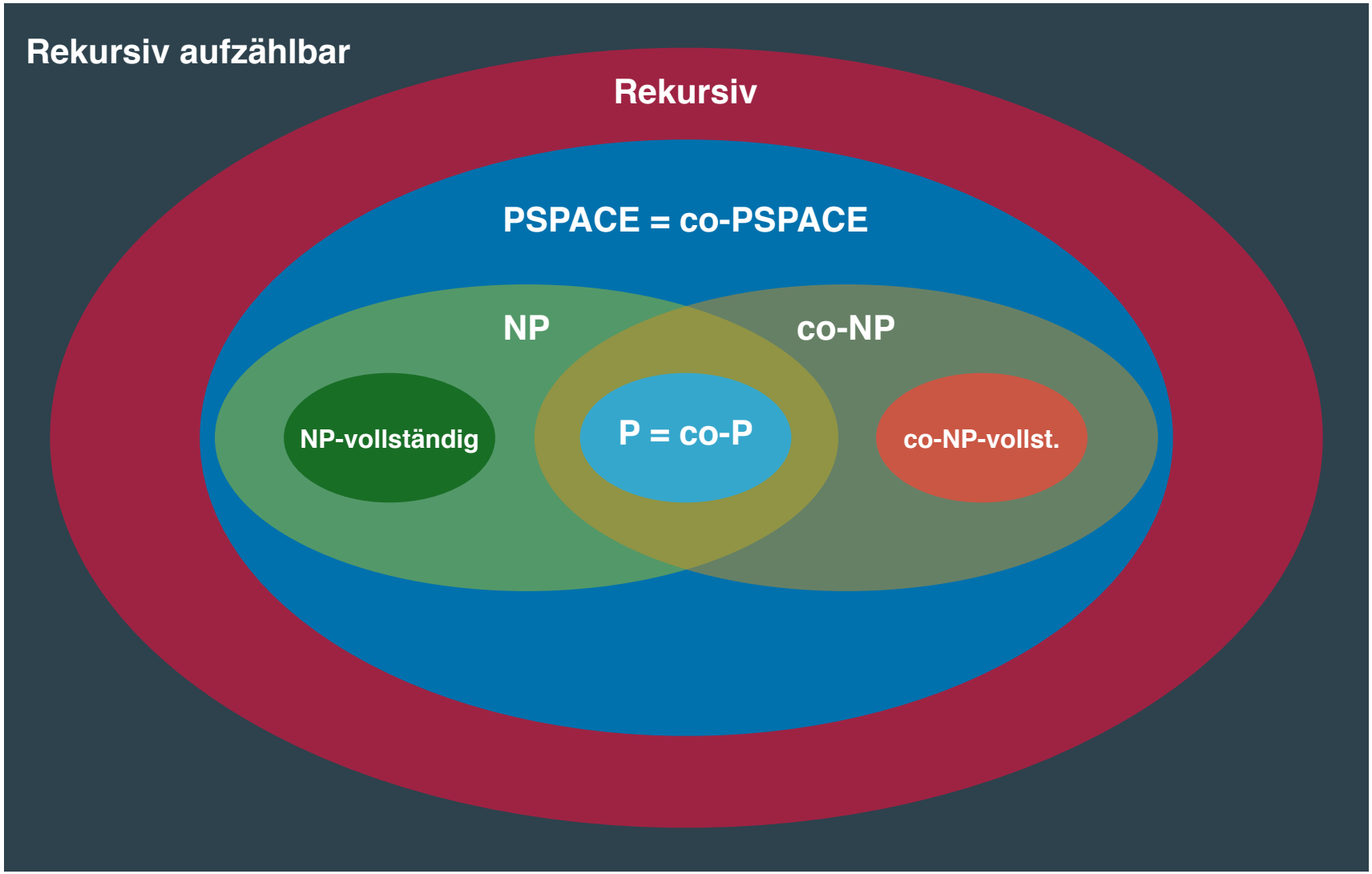
Im SAT Fall:

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

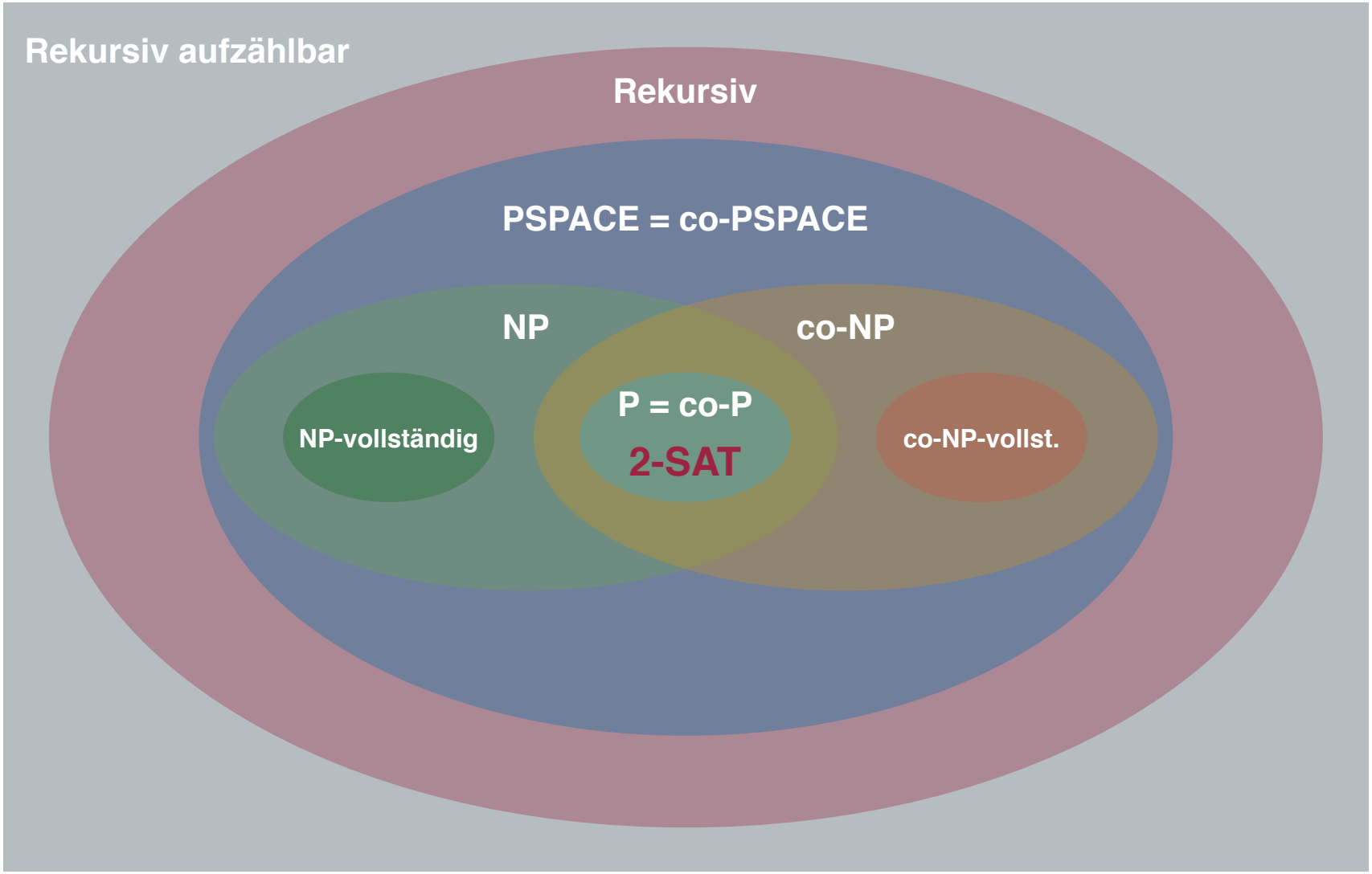
Im QBF Fall:

- 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

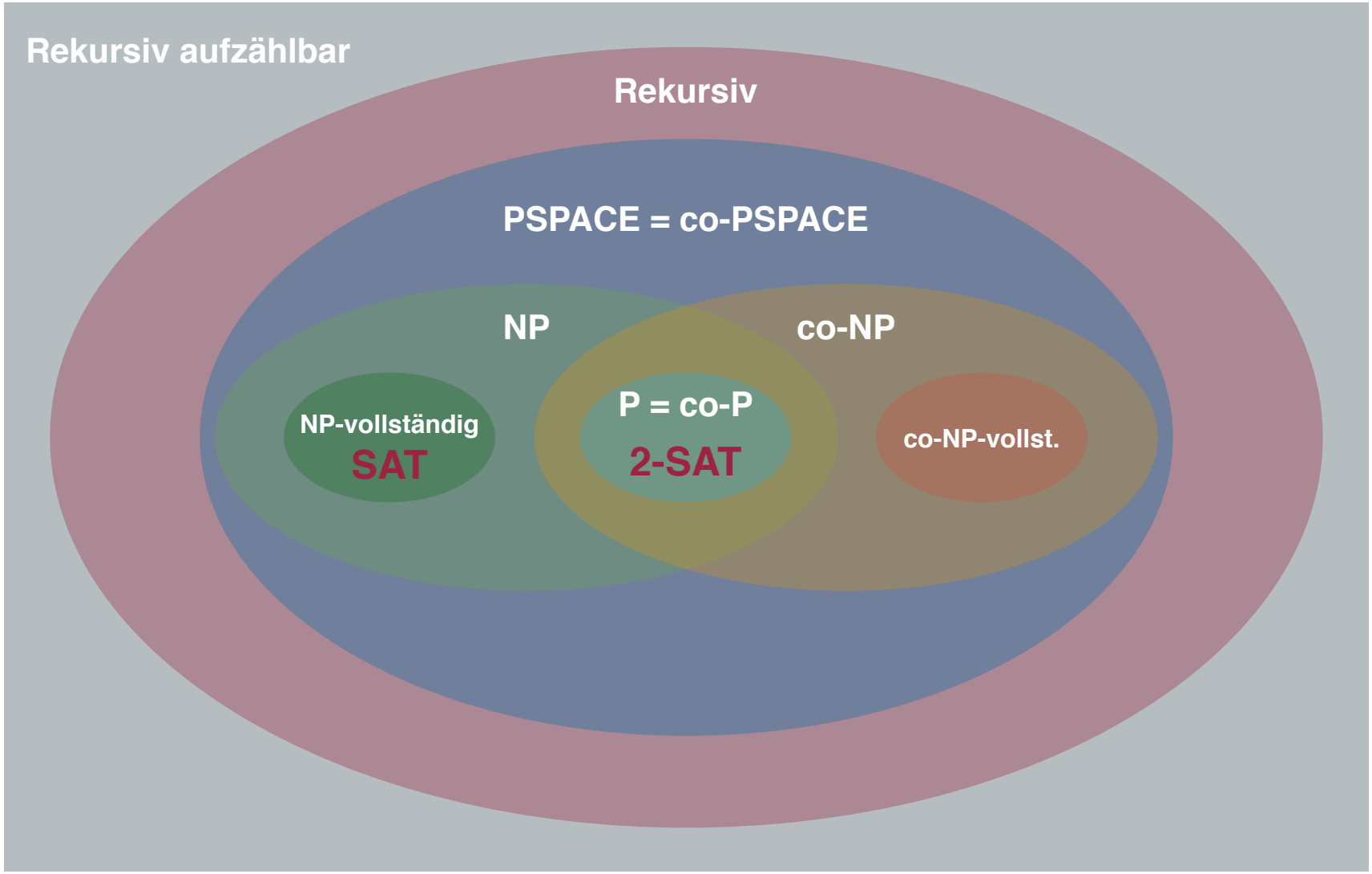
Komplexität von QBF - 2



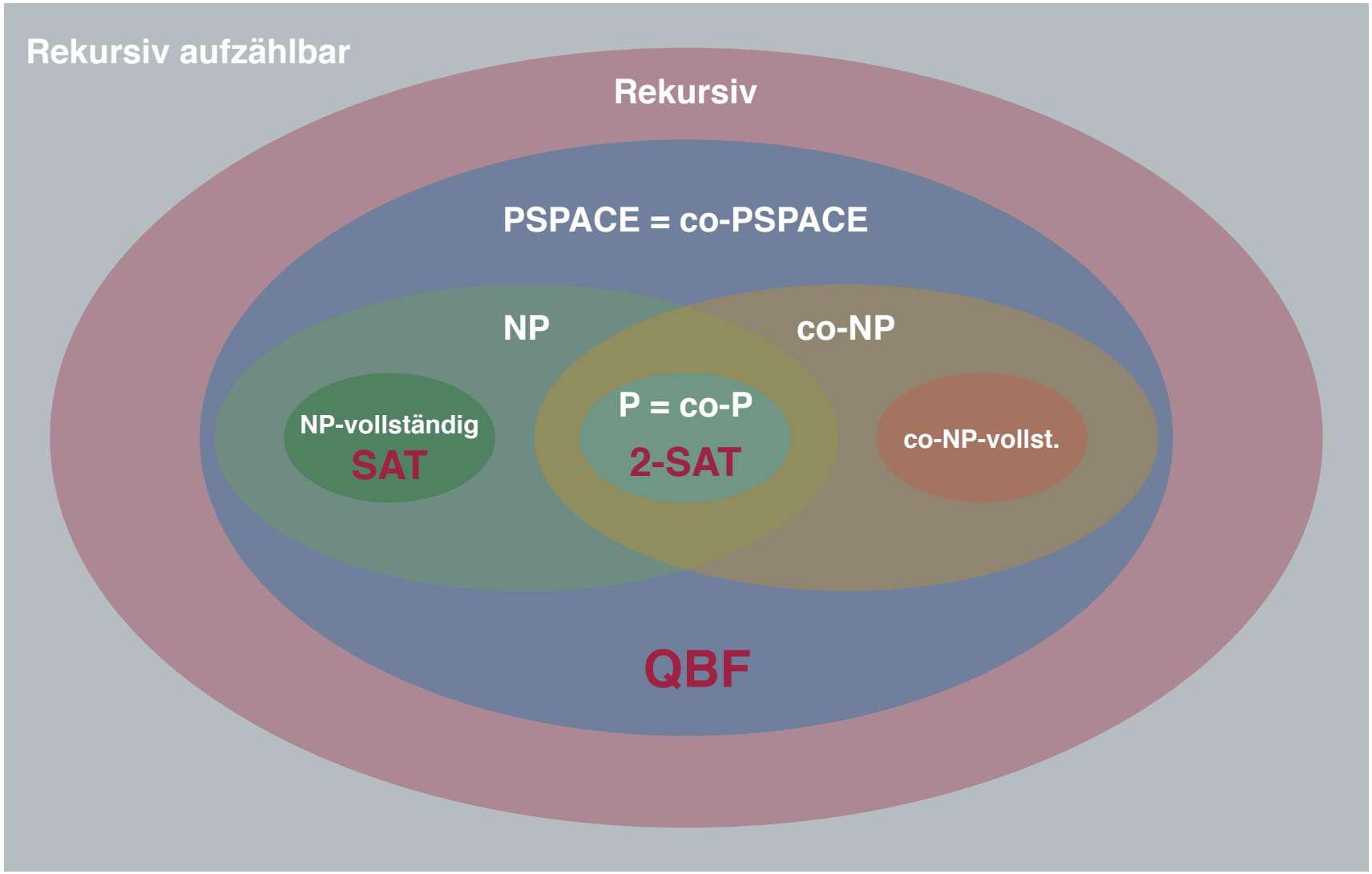
Komplexität von QBF - 2



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)

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

- 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 ist.

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 $e' \in E(C), e' \neq e$ gilt, dass $\nu(e') = \bot$
3. Für alle $u \in U(C)$ mit $\nu(u) \neq \top$ gilt, dass $\nu(u) = nil \Rightarrow ql(u) > ql(e)$

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).

- Nur existenzquantifizierte Variablen können durch UP impliziert werden

Backtracking für eine erfüllende Belegung

- Für allquantifizierte Variablen müssen beide Belegungen getestet werden
- Jede allquantifizierte Variable muss geflipped werden
- Jede allquantifizierte Variable bekommt ein flag "flipped"
- Wird x 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.

Beispiel (Long Distance Resolution)

$$(a_{(1)} \vee b_{(3)} \vee x_{(4)} \vee y_{(4)} \vee c_{(5)}) \wedge (a_{(1)} \vee \neg b_{(3)} \vee \neg x_{(4)} \vee \neg y_{(4)} \vee d_{(5)})$$

Resolution über b ist eine Tautologie:

$$(a_{(1)} \vee x_{(4)} \vee \neg x_{(4)} \vee y_{(4)} \vee \neg y_{(4)} \vee c_{(5)} \vee d_{(5)})$$

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

Wegen geändertem Kriterium für Unit Clauses muss auch das Stopp-Kriterium für 1UIP angepasst werden

Stopp-Kriterium für 1UIP

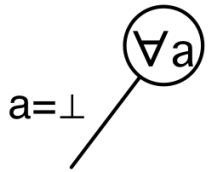
1. Nur eine existenzquantifizierte Variable e ist auf höchstem Level
2. e ist auf einem Decision Level, auf dem die Entscheidungsvariable existenzquantifiziert ist
3. Alle allquantifizierten Variablen u mit $ql(u) < ql(e)$ werden zu 0 evaluiert auf einem Decision Level $<$ dem von e

Wie bei SAT: Backtracking zu größtem Level, der $<$ dem Decision Level von e ist.

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}

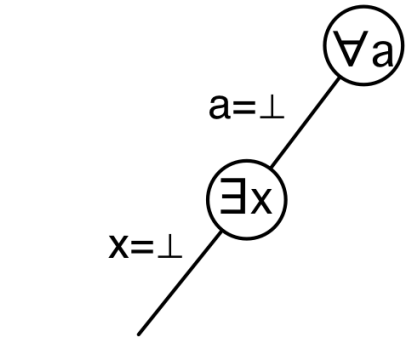


Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision

Ein Beispiel für QBF

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

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



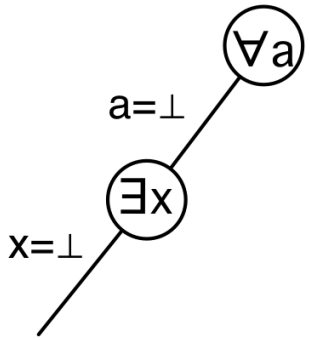
Konflikt: {x,¬y,z,b}

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

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

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



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

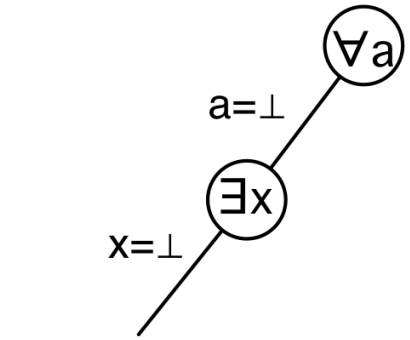
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
2	x	∃(2)	⊥	decision
	z	∃(4)	⊥	$\{x,\neg z,a\}$
	y	∃(2)	⊤	$\{x,y,z,\neg b\}$

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

Ein Beispiel für QBF

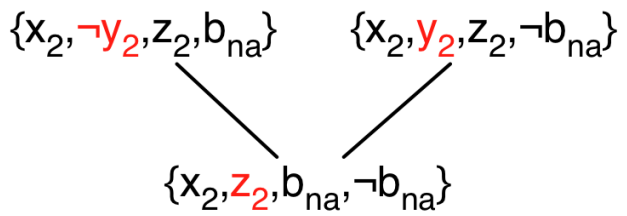
$\forall a \exists x \exists y \forall b \exists z$

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



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

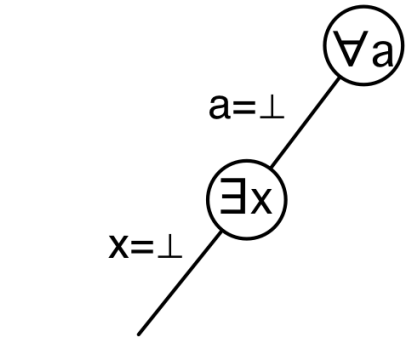
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
2	x	∃(2)	⊥	decision
	z	∃(4)	⊥	$\{x,\neg z,a\}$
	y	∃(2)	⊤	$\{x,y,z,\neg b\}$



Ein Beispiel für QBF

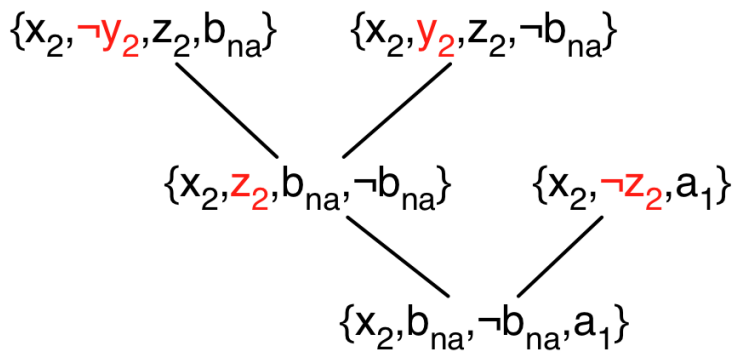
$\forall a \exists x \exists y \forall b \exists z$

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



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

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

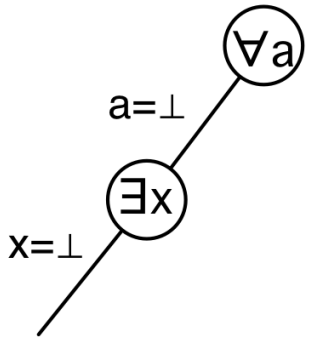


Ein Beispiel für QBF

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

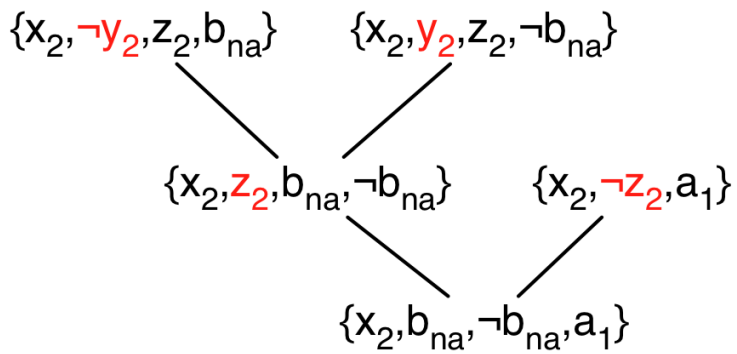
{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



Konflikt: {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}

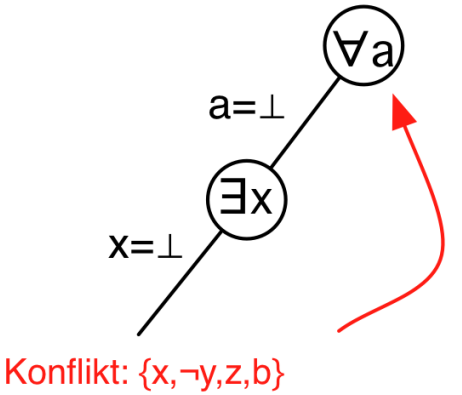


Ein Beispiel für QBF

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

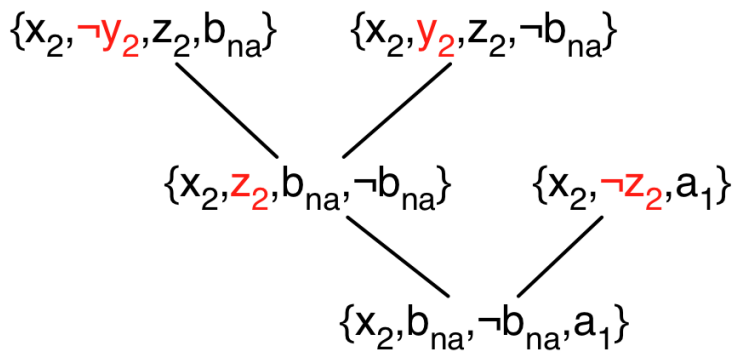
{x,y,z,¬b}
{x,¬z,a}
{¬x,y,a}
{x,¬y,z,b}

{x,b,¬b,a}



Backtracking zu Level 1

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}

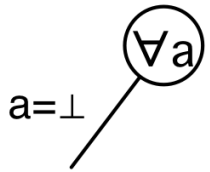


Ein Beispiel für QBF

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

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

{x,b,¬b,a}



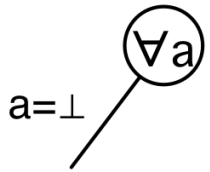
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision

Ein Beispiel für QBF

∀a∃x∃y∀b∃z

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

{x,b,¬b,a}



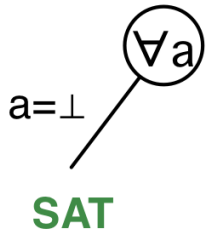
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
	x	∃(2)	T	{x,b,¬b,a}
	y	∃(2)	T	{¬x,y,a}

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}



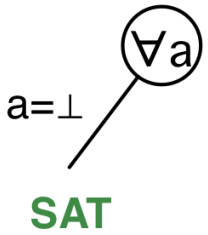
Level	Var	Quant	Wert	Grund
1	a	∀(1) F	⊥	decision
	x	∃(2)	T	{x,b,¬b,a}
	y	∃(2)	T	{¬x,y,a}

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}



SAT

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

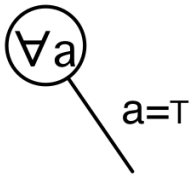
Backtracking zur letzten Variable mit Flag F

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}



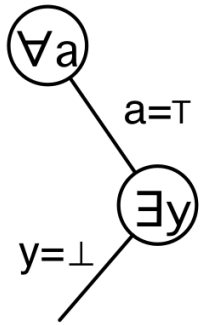
Level	Var	Quant	Wert	Grund
1	a	∀(1) T	T	decision

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}



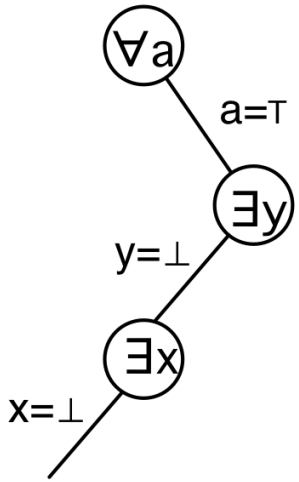
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision

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}



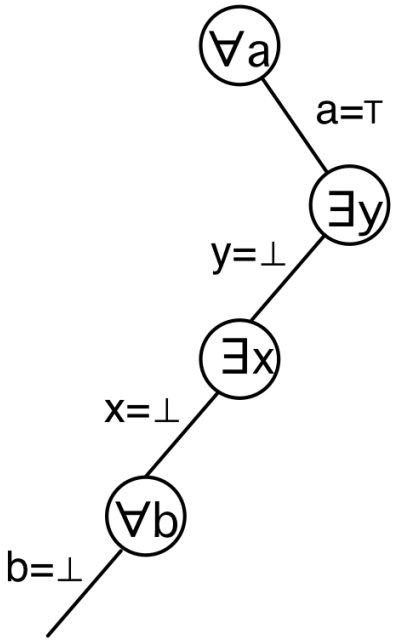
Level	Var	Quant	Wert	Grund
1	a	∀(1) T	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision

Ein Beispiel für QBF

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

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

{x,b,¬b,a}



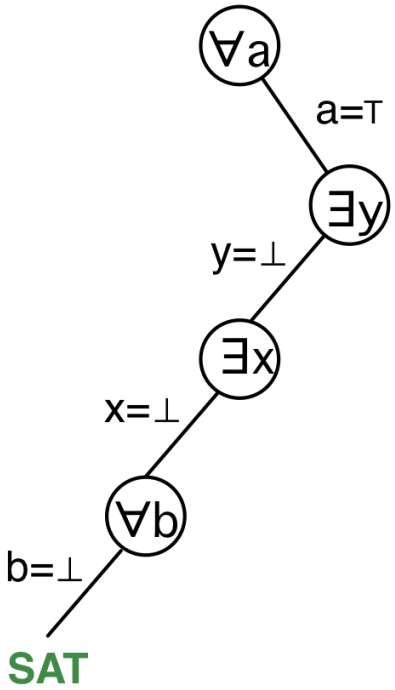
Level	Var	Quant	Wert	Grund
1	a	∀(1) T	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) F	⊥	decision

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}



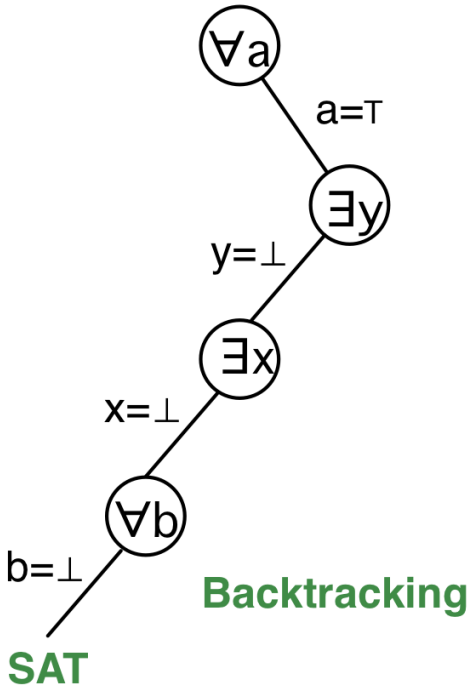
Level	Var	Quant	Wert	Grund
1	a	∀(1) T	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) F	⊥	decision

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}



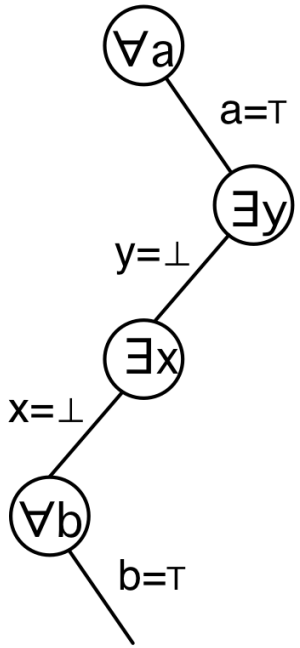
Level	Var	Quant	Wert	Grund
1	a	∀(1) T	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) F	⊥	decision

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}



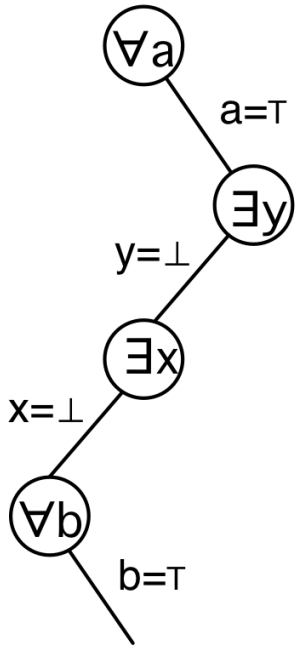
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) ⊤	⊤	decision

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}



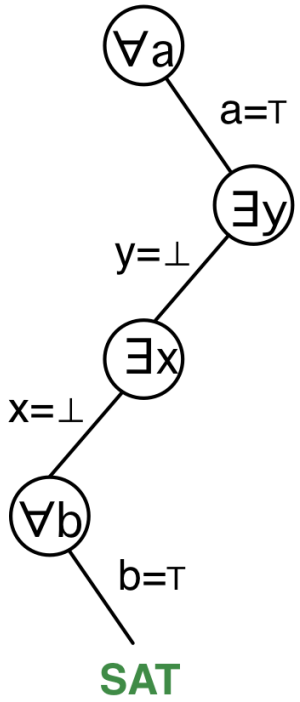
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) ⊤	⊤	decision
	z	∃(4)	⊤	{x,y,z,¬b}

Ein Beispiel für QBF

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

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

{x,b,¬b,a}



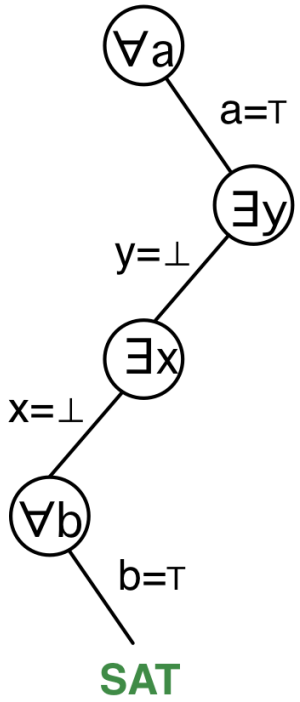
Level	Var	Quant	Wert	Grund
1	a	∀(1) ⊤	⊤	decision
2	y	∃(2)	⊥	decision
3	x	∃(2)	⊥	decision
4	b	∀(3) ⊤	⊤	decision
	z	∃(4)	⊤	{x,y,z,¬b}

Ein Beispiel für QBF

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

{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}

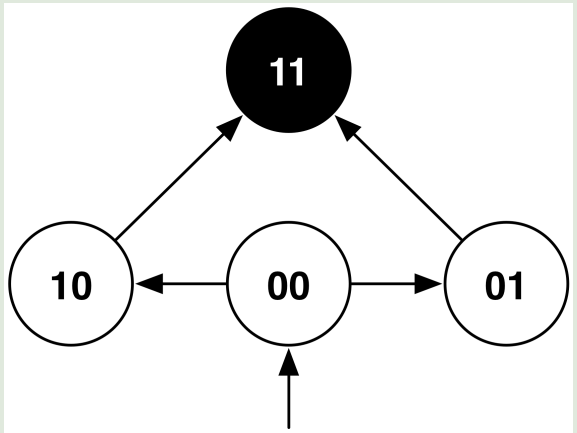
Keine weiteren Flags mit F

→ Result: TRUE

Codierung mit QBF - 1

Beispiel (Endlicher Automat)

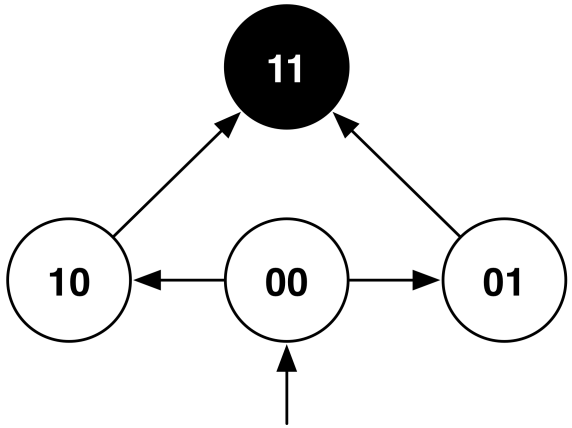
- 4 Zustände: $00, 01, 10, 11$
- Startzustand: 00
- Zustandsübergänge: $00 \rightarrow 01, 00 \rightarrow 10, 10 \rightarrow 11, 01 \rightarrow 11$
- Fehlerzustand: 11



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

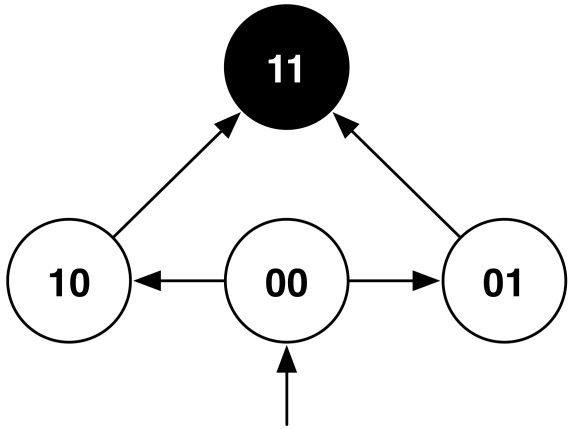


Fragestellung: Erreicht man einen Fehlerzustand in einem Schritt?

- 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

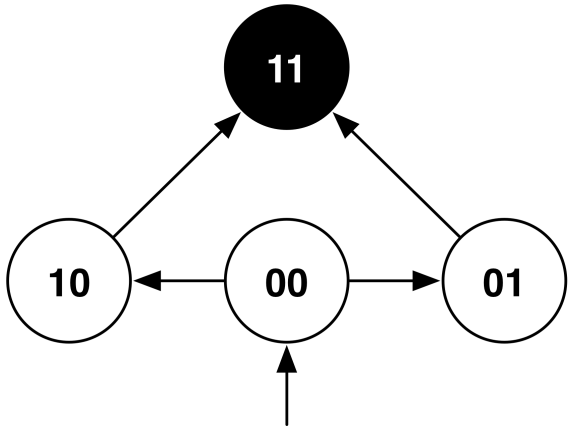


Fragestellung: Erreicht man einen Fehlerzustand in zwei Schritten?

- 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

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

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)

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

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

Beispiel (PQSAT)

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

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

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 Counting) (2)
- Knowledge Compilation Ansatz (DNNF) (2)

Aktuelle 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