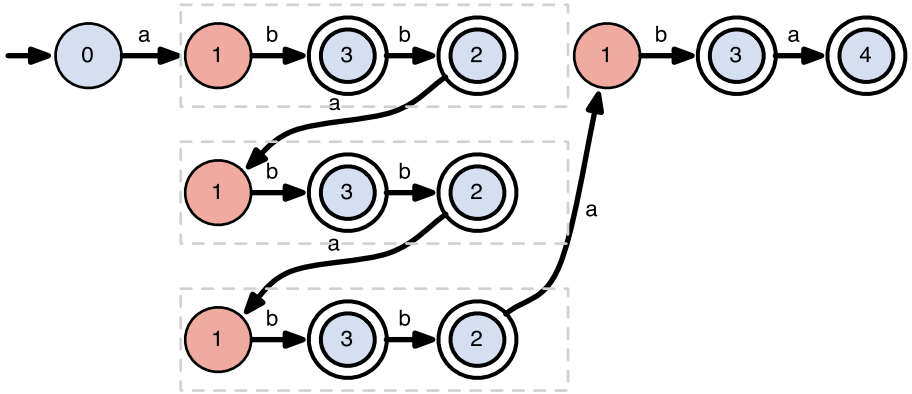
Berechenbarkeitstheorie

22. Vorlesung



Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

Satz von Cook-Levin

SAT ist NP-vollständig.

Satz von Cook-Levin

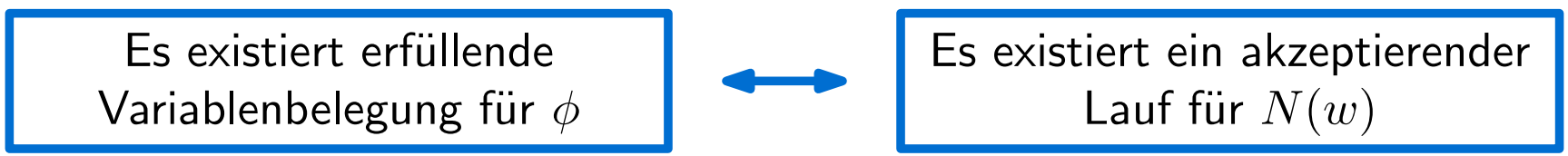
SAT ist NP-vollständig.

- Wir betrachten ein $L \in$ NP (fest für den Rest des Beweises) mit polyzeit NTM N, welche L erkennt

Satz von Cook-Levin

SAT ist NP-vollständig.

- Wir betrachten ein $L \in$ NP (fest für den Rest des Beweises) mit polyzeit NTM N, welche L erkennt



Reduktion: $f_N(w) = \phi$

Satz von Cook-Levin

SAT ist NP-vollständig.

- Wir betrachten ein $L \in$ NP (fest für den Rest des Beweises) mit polyzeit NTM N, welche L erkennt

Es existiert erfüllende Variablenbelegung für ϕ	$\longleftrightarrow$	Es existiert ein akzeptierender Lauf für $N(w)$

Reduktion: $f_N(w) = \phi$

Tableau:

#	q_0	a	b	b	b	□	□	□	□	□	#
#	b	q_3	b	b	b	□	□	□	□	□	#
#	a	a	b	b	b	q_A	a	□	□	□	#
#	a	a	b	b	b	q_A	a	□	□	□	#

Satz von Cook-Levin

SAT ist NP-vollständig.

- Wir betrachten ein $L \in$ NP (fest für den Rest des Beweises) mit polyzeit NTM N, welche L erkennt

Es existiert erfüllende Variablenbelegung für ϕ	$\longleftrightarrow$	Es existiert ein akzeptierender Lauf für $N(w)$

Reduktion: $f_N(w) = \phi$

Tableau:

#	q_0	a	b	b	b	□	□	□	□	□	#
#	b	q_3	b	b	b	□	□	□	□	□	#
#	a	a	b	b	b	q_A	a	□	□	□	#
#	a	a	b	b	b	q_A	a	□	□	□	#

- Tableau existiert mit $q_A \Leftrightarrow$ es gibt für $N(w)$ akz. Lauf $\Leftrightarrow w \in L$

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w
- Formel ϕ besteht aus 4 Teilen:

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w
- Formel ϕ besteht aus 4 Teilen:

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- ϕ_{zelle} sichert die korrekte Verwendung der Variablen x für das Tableau

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w
- Formel ϕ besteht aus 4 Teilen:

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- ϕ_{zelle} sichert die korrekte Verwendung der Variablen x für das Tableau
- ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w
- Formel ϕ besteht aus 4 Teilen:

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- ϕ_{zelle} sichert die korrekte Verwendung der Variablen x für das Tableau
- ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält
- ϕ_{akz} sichert dass q_A im Tableau auftritt

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w
- Formel ϕ besteht aus 4 Teilen:

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- ϕ_{zelle} sichert die korrekte Verwendung der Variablen x für das Tableau
- ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält
- ϕ_{akz} sichert dass q_A im Tableau auftritt
- ϕ_{trans} sichert Zeile i eine Nachfolgerkonfiguration der Zeile $(i-1)$ ist

- Variablen von ϕ kodieren die Zellenbelegung des Tableaus

$$x_{i,j,s} = 1 \iff \text{Tabellenzelle } (i,j) \text{ enthält Symbol } s$$

- $s \in C := Q \cup \Gamma \cup \{\#, \square\}$, C hängt von N ab, aber nicht von w
- Formel ϕ besteht aus 4 Teilen:

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- ϕ_{zelle} sichert die korrekte Verwendung der Variablen x für das Tableau
- ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält
- ϕ_{akz} sichert dass q_A im Tableau auftritt
- ϕ_{trans} sichert Zeile i eine Nachfolgerkonfiguration der Zeile $(i-1)$ ist

Alles zusammen erzwingt, dass die Variablen ein (korrektes) Tableau mit q_A beschreiben

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i, j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i, j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$$

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i, j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

Für jedes Paar (i, j)

mindestens ein $x_{i,j,.}$ auf 1

keine zwei verschieden $x_{i,j,.}$ auf 1

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i, j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

Für jedes Paar (i, j)

mindestens ein $x_{i,j,.}$ auf 1

keine zwei verschieden $x_{i,j,.}$ auf 1

② ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i, j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

Für jedes Paar (i, j)

mindestens ein $x_{i,j,.}$ auf 1

keine zwei verschieden $x_{i,j,.}$ auf 1

② ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält

- sei $w = a_1 a_2 \cdots a_n$

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,n+2,a_n} \wedge x_{1,n+3,\square} \wedge \cdots \wedge x_{1,n^k,\#}$$

erzwinge Zeichnen für Zeichen

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i,j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

Für jedes Paar (i,j)

mindestens ein $x_{i,j,.}$ auf 1

keine zwei verschieden $x_{i,j,.}$ auf 1

② ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält

- sei $w = a_1 a_2 \cdots a_n$

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,n+2,a_n} \wedge x_{1,n+3,\square} \wedge \cdots \wedge x_{1,n^k,\#}$$

erzwinge Zeichnen für Zeichen

③ ϕ_{akz} sichert dass q_A im Tableau auftritt

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i,j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$$

Für jedes Paar (i,j)

mindestens ein $x_{i,j,.}$ auf 1

keine zwei verschieden $x_{i,j,.}$ auf 1

② ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält

- sei $w = a_1 a_2 \cdots a_n$

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,n+2,a_n} \wedge x_{1,n+3,\square} \wedge \cdots \wedge x_{1,n^k,\#}$$

erzwinge Zeichnen für Zeichen

③ ϕ_{akz} sichert dass q_A im Tableau auftritt

$$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A}$$

① ϕ_{zelle} sichert die korrekte Verwendung der Variablen x

- wir müssen sicherstellen, dass für jede Zelle (i, j) genau ein $x_{i,j,.}$ auf 1 gesetzt wird

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

Für jedes Paar (i, j)

mindestens ein $x_{i,j,.}$ auf 1

keine zwei verschieden $x_{i,j,.}$ auf 1

② ϕ_{start} sichert dass erste Zeile Startkonfiguration enthält

- sei $w = a_1 a_2 \cdots a_n$

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,n+2,a_n} \wedge x_{1,n+3,\square} \wedge \cdots \wedge x_{1,n^k,\#}$$

erzwinge Zeichnen für Zeichen

③ ϕ_{akz} sichert dass q_A im Tableau auftritt

$$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A}$$

In mindestens einer Zelle steht q_A

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau
- Übergangsfunktion von N legt fest, welche Fenster erlaubt sind

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau
- Übergangsfunktion von N legt fest, welche Fenster erlaubt sind

Bsp.

a	q_4	b
q_1	a	a

erlaubt, falls $(q_1, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau
- Übergangsfunktion von N legt fest, welche Fenster erlaubt sind

Bsp.

a	q_4	b
q_1	a	a

erlaubt, falls $(q_1, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

q_2	a	b
b	q_3	b

erlaubt, falls $(q_3, \mathtt{b}, R) \in \delta(q_2, \mathtt{a})$

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau
- Übergangsfunktion von N legt fest, welche Fenster erlaubt sind

Bsp.

a	q_4	b
q_1	a	a

erlaubt, falls $(q_1, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

q_2	a	b
b	q_3	b

erlaubt, falls $(q_3, \mathtt{b}, R) \in \delta(q_2, \mathtt{a})$

q_4	b	a
a	a	a

erlaubt, falls $(\cdot, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau
- Übergangsfunktion von N legt fest, welche Fenster erlaubt sind

Bsp.

a	q_4	b
q_1	a	a

erlaubt, falls $(q_1, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

q_2	a	b
b	q_3	b

erlaubt, falls $(q_3, \mathtt{b}, R) \in \delta(q_2, \mathtt{a})$

q_4	b	a
a	a	a

erlaubt, falls $(\cdot, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

b	a	□
b	a	□

immer erlaubt

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

- **Fenster**=3x2 Ausschnitt aus Tableau
- Übergangsfunktion von N legt fest, welche Fenster erlaubt sind

Bsp.

a	q_4	b
q_1	a	a

erlaubt, falls $(q_1, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

q_2	a	b
b	q_3	b

erlaubt, falls $(q_3, \mathtt{b}, R) \in \delta(q_2, \mathtt{a})$

q_4	b	a
a	a	a

erlaubt, falls $(\cdot, \mathtt{a}, L) \in \delta(q_4, \mathtt{b})$

b	a	$\square$
b	a	$\square$

immer erlaubt

- ... und viele mehr (aber endlich viele, bei festem N)

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

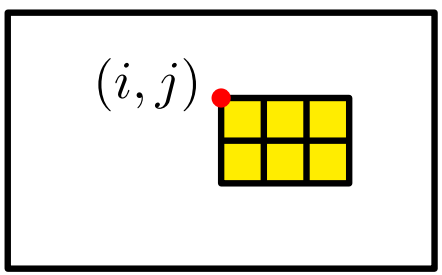
④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

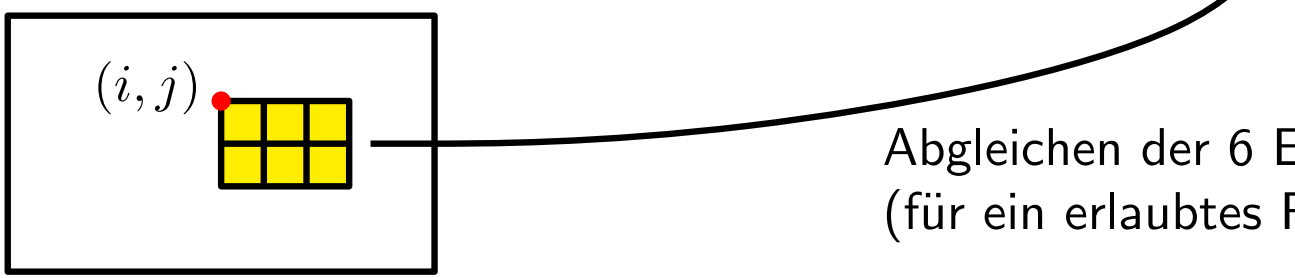
Für jedes Paar (i,j) mit $i \leq n^k - 3$ und $i \leq n^k - 2$



④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

Für jedes Paar (i,j) mit $i \leq n^k - 3$ und $i \leq n^k - 2$

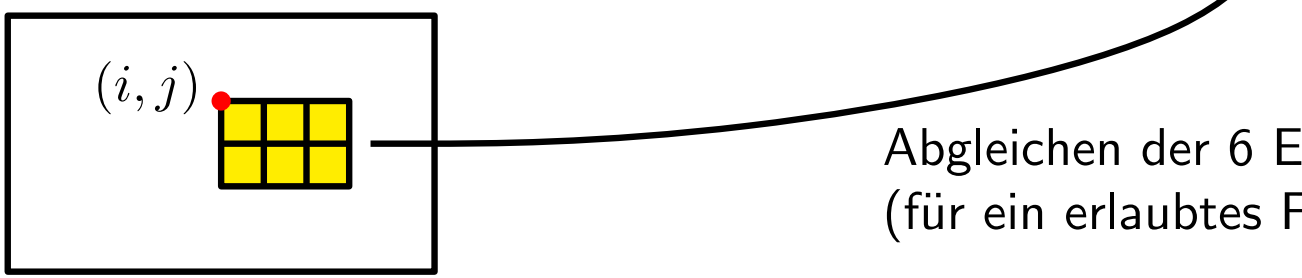


Abgleichen der 6 Einträge des Fensters
(für ein erlaubtes Fenster)

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

Für jedes Paar (i, j) mit $i \leq n^k - 3$ und $i \leq n^k - 2$



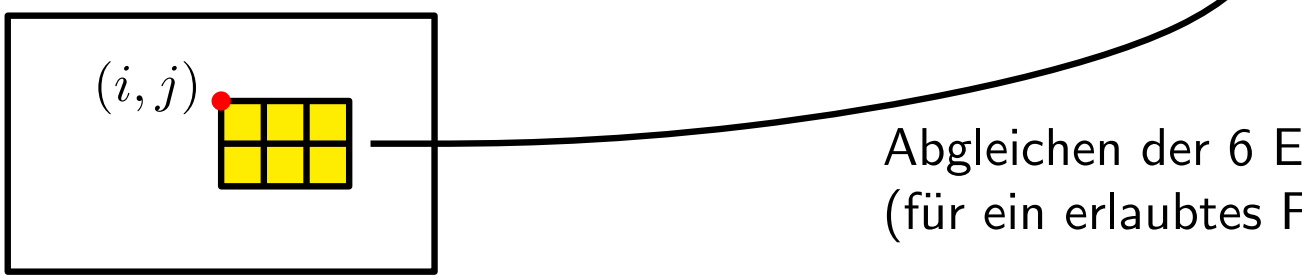
Abgleichen der 6 Einträge des Fensters
(für ein erlaubtes Fenster)

- wenn wir die Tabelle mit erlaubten Fenstern überdecken können, handelt es sich um ein korrekt ausgefülltes Tableau

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

Für jedes Paar (i, j) mit $i \leq n^k - 3$ und $i \leq n^k - 2$



Abgleichen der 6 Einträge des Fensters
(für ein erlaubtes Fenster)

- wenn wir die Tabelle mit erlaubten Fenstern überdecken können, handelt es sich um ein korrekt ausgefülltes Tableau

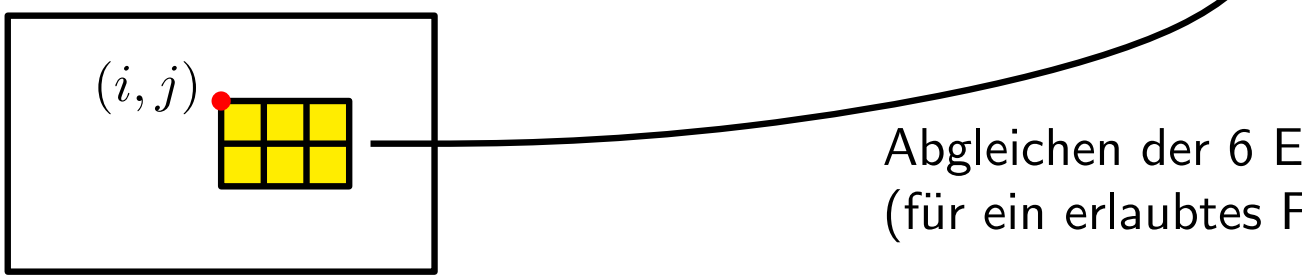
---

- Also: Wenn ϕ erfüllbar ist, existiert ein korrekt ausgefülltes akz. Tableau, und somit auch ein akzeptierender Lauf für $N(w)$

④ ϕ_{trans} sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

Für jedes Paar (i, j) mit $i \leq n^k - 3$ und $i \leq n^k - 2$



Abgleichen der 6 Einträge des Fensters
(für ein erlaubtes Fenster)

- wenn wir die Tabelle mit erlaubten Fenstern überdecken können, handelt es sich um ein korrekt ausgefülltes Tableau

---

- Also: Wenn ϕ erfüllbar ist, existiert ein korrekt ausgefülltes akz. Tableau, und somit auch ein akzeptierender Lauf für $N(w)$
- Weil N polyzeit beschränkt, reicht es aus die Tableaubreite n^k zu wählen

Ist diese Reduktion eine polyzeit Reduktion?

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

$\Big\}$ $|\phi| = O(n^{2k})$

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

$\Big\}$ $|\phi| = O(n^{2k})$

- die meisten Teile von ϕ sind Konstanten

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

$\Big\}$ $|\phi| = O(n^{2k})$

- die meisten Teile von ϕ sind Konstanten
- bei verschiedenen w ändert sich für ϕ_{trans}, ϕ_{akz}, und ϕ_{zelle} nur die Anzahl der Bedingungen

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

$\Big\}$ $|\phi| = O(n^{2k})$

- die meisten Teile von ϕ sind Konstanten
- bei verschiedenen w ändert sich für ϕ_{trans}, ϕ_{akz}, und ϕ_{zelle} nur die Anzahl der Bedingungen
- ϕ kann durch Kopieren von Konstanten und durch die Modifikation von ϕ_{start} gebildet werden

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

$\Big\}$ $|\phi| = O(n^{2k})$

- die meisten Teile von ϕ sind Konstanten
- bei verschiedenen w ändert sich für ϕ_{trans}, ϕ_{akz}, und ϕ_{zelle} nur die Anzahl der Bedingungen
- ϕ kann durch Kopieren von Konstanten und durch die Modifikation von ϕ_{start} gebildet werden
- Laufzeit t_f ist ein Polynom in n^k, also auch ein Polynom in n

Ist diese Reduktion eine polyzeit Reduktion?

- Es gibt $O(n^{2k})$ Variablen in ϕ
- $|\phi_{\text{zelle}}| = O(n^{2k})$
- $|\phi_{\text{start}}| = O(n^{k})$
- $|\phi_{\text{akz}}| = O(n^{2k})$
- $|\phi_{\text{trans}}| = O(n^{2k})$

$\Big\}\ |\phi| = O(n^{2k})$

- die meisten Teile von ϕ sind Konstanten
- bei verschiedenen w ändert sich für ϕ_{trans}, ϕ_{akz}, und ϕ_{zelle} nur die Anzahl der Bedingungen
- ϕ kann durch Kopieren von Konstanten und durch die Modifikation von ϕ_{start} gebildet werden
- Laufzeit t_f ist ein Polynom in n^k, also auch ein Polynom in n
- Reduktion f ist polyzeit Reduktion

□

SAT-Varianten

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen
- eine boolsche Formel ist in konjunktiver Normalform (CNF), gdw. sie eine Konjunktion von Klauseln ist

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen
- eine boolsche Formel ist in konjunktiver Normalform (CNF), gdw. sie eine Konjunktion von Klauseln ist

Bsp. $(x_1 \vee \neg x_3) \wedge x_2 \wedge (\neg x_2 \vee x_1 \vee x_4 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen
- eine boolsche Formel ist in konjunktiver Normalform (CNF), gdw. sie eine Konjunktion von Klauseln ist

Bsp. $(x_1 \vee \neg x_3) \wedge x_2 \wedge (\neg x_2 \vee x_1 \vee x_4 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Klauseln

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen
- eine boolsche Formel ist in konjunktiver Normalform (CNF), gdw. sie eine Konjunktion von Klauseln ist

Bsp. $(x_1 \vee \neg x_3) \wedge x_2 \wedge (\neg x_2 \vee x_1 \vee x_4 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Klauseln Literale

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen
- eine boolsche Formel ist in konjunktiver Normalform (CNF), gdw. sie eine Konjunktion von Klauseln ist

Bsp. $(x_1 \vee \neg x_3) \wedge x_2 \wedge (\neg x_2 \vee x_1 \vee x_4 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Klauseln Literale

CNF-SAT

Eingabe: Boolsche Formel ϕ in CNF in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

SAT-Varianten

- ein Literal ist eine Variable, oder eine negierte Variable
- eine Klausel ist eine Disjunktion von Literalen
- eine boolsche Formel ist in konjunktiver Normalform (CNF), gdw. sie eine Konjunktion von Klauseln ist

Bsp. $(x_1 \vee \neg x_3) \wedge x_2 \wedge (\neg x_2 \vee x_1 \vee x_4 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

Klauseln Literale

CNF-SAT

Eingabe: Boolsche Formel ϕ in CNF in den Variablen x_i
Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

3SAT

Eingabe: Boolsche Formel ϕ in CNF mit ≤ 3 Literalen pro Klausel in den Variablen x_i
Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

Satz 41

CNF-SAT ist NP-vollständig.

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$$

CNF!

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$ CNF!

$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,4,a_n} \wedge x_{1,4,\square} \wedge \cdots \wedge x_{1,n^k,\#}$ CNF!

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$ CNF!

$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,4,a_n} \wedge x_{1,4,\square} \wedge \cdots \wedge x_{1,n^k,\#}$ CNF!

$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A}$ CNF!

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$ CNF!

$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,4,a_n} \wedge x_{1,4,\square} \wedge \cdots \wedge x_{1,n^k,\#}$ CNF!

$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A}$ CNF!

$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$ CNF!

$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,4,a_n} \wedge x_{1,4,\square} \wedge \cdots \wedge x_{1,n^k,\#}$ CNF!

$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A}$ CNF!

$\phi_{\text{trans}} = \bigwedge_{i,j} \underbrace{\bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})}_{\text{nicht in CNF}}$

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) \right]$$ CNF!

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,4,a_n} \wedge x_{1,4,\square} \wedge \cdots \wedge x_{1,n^k,\#}$$ CNF!

$$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A}$$ CNF!

$$\phi_{\text{trans}} = \bigwedge_{i,j} \underbrace{\bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})}_{\text{nicht in CNF}}$$

Kann aber in CNF umgewandelt werden. Da dieser Teil unabhängig von der Eingabe ist kostet dies nur einen konstanten Faktor mehr Aufwand!

Satz 41

CNF-SAT ist NP-vollständig.

Beweis

- wir modifizieren den Beweis des Satzes von Cook-Levin, sodass die Reduktion eine Formel ϕ in CNF liefert

$$\phi = \phi_{\text{zelle}} \wedge \phi_{\text{start}} \wedge \phi_{\text{akz}} \wedge \phi_{\text{trans}}$$

- wir müssen zeigen, dass die 4 Teile in CNF sind

$$\phi_{\text{zelle}} = \bigwedge_{i,j \leq n^k} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in S, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right] \quad \text{CNF!}$$

$$\phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,a_1} \wedge \cdots x_{1,4,a_n} \wedge x_{1,4,\square} \wedge \cdots \wedge x_{1,n^k,\#} \quad \text{CNF!}$$

$$\phi_{\text{akz}} = \bigvee_{i,j \leq n^k} x_{i,j,q_A} \quad \text{CNF!}$$

$$\phi_{\text{trans}} = \bigwedge_{i,j} \underbrace{\bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})}_{\text{nicht in CNF}}$$

Kann aber in CNF umgewandelt werden. Da dieser Teil unabhängig von der Eingabe ist kostet dies nur einen konstanten Faktor mehr Aufwand!

- die modifizierte Formel ist in CNF! □

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

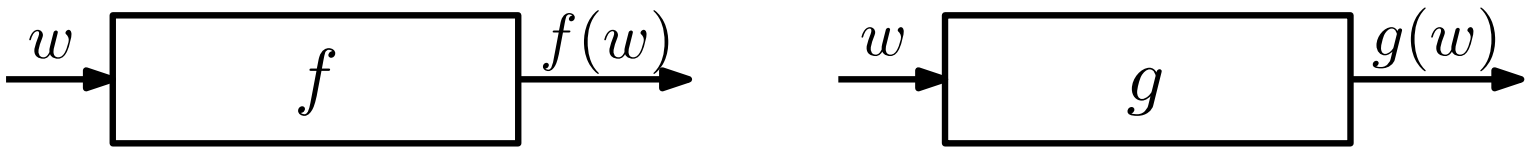
- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)

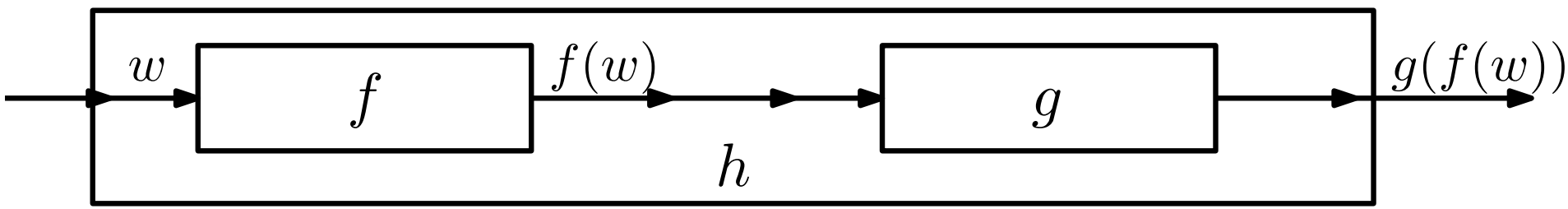


Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)

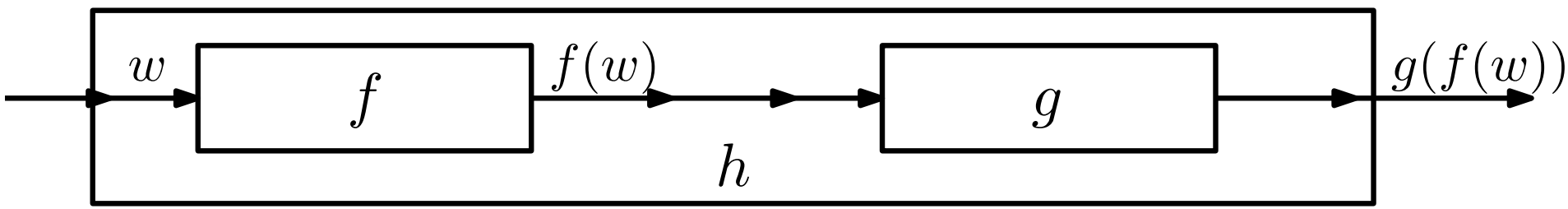


Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)



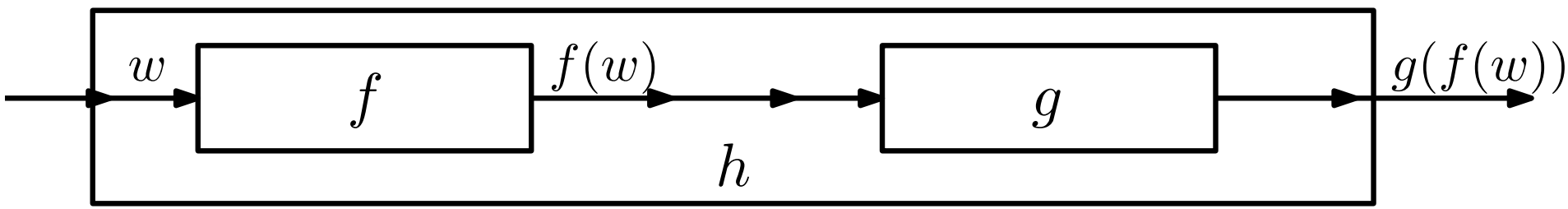
  - $w \in A \iff f(w) \in B \iff g(f(w)) \in C$

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)



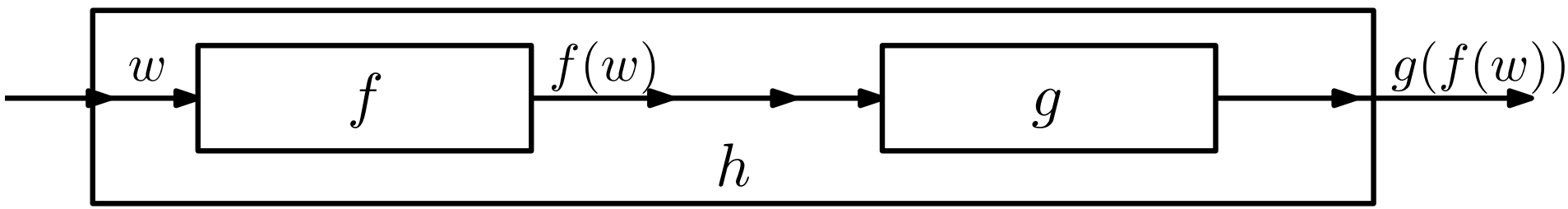
  - $w \in A \iff f(w) \in B \iff g(f(w)) \in C$
  - die Funktion $h = f \circ g$ ist eine Reduktion von A nach C, deren Laufzeit ein Polynom ist

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)



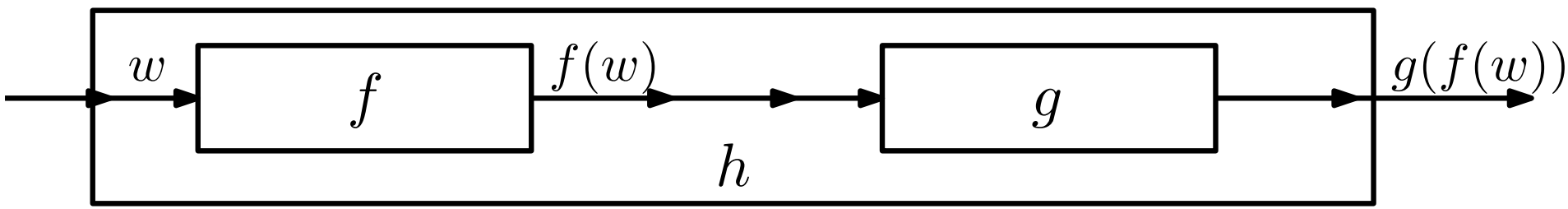
  - $w \in A \iff f(w) \in B \iff g(f(w)) \in C$
  - die Funktion $h = f \circ g$ ist eine Reduktion von A nach C, deren Laufzeit ein Polynom ist
  - also: $A \leq_p C$

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)



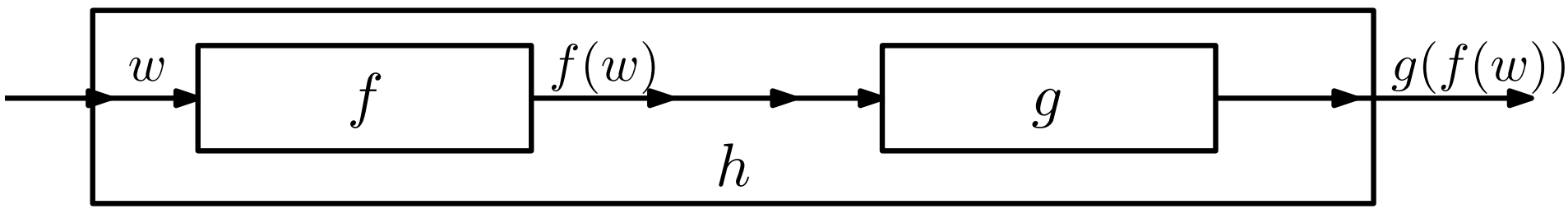
  - $w \in A \iff f(w) \in B \iff g(f(w)) \in C$
  - die Funktion $h = f \circ g$ ist eine Reduktion von A nach C, deren Laufzeit ein Polynom ist
  - also: $A \leq_p C$
- für A gilt nach Def.: $\forall L' \in$ NP$: L' \leq_p A$

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)



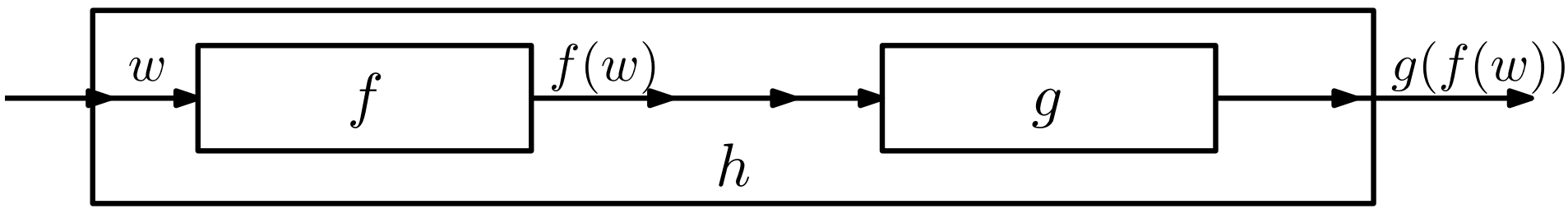
  - $w \in A \iff f(w) \in B \iff g(f(w)) \in C$
  - die Funktion $h = f \circ g$ ist eine Reduktion von A nach C, deren Laufzeit ein Polynom ist
  - also: $A \leq_p C$
- für A gilt nach Def.: $\forall L' \in$ NP: $L' \leq_p A$
- auf Grund der Transitivität gilt da $\forall L' \in$ NP: $L' \leq_p A \leq_p B$ auch $\forall L' \in$ NP: $L' \leq_p B$

Lemma

Wenn $A \in$ NPC, $B \in$ NP und $A \leq_p B$, dann ist $B \in$ NPC.

Beweis

- die Relation $\leq_p$ ist transitiv
  - sei $A \leq_p B$ (via Reduktion f) und $B \leq_p C$ (via Reduktion g)



  - $w \in A \iff f(w) \in B \iff g(f(w)) \in C$
  - die Funktion $h = f \circ g$ ist eine Reduktion von A nach C, deren Laufzeit ein Polynom ist
  - also: $A \leq_p C$
- für A gilt nach Def.: $\forall L' \in$ NP: $L' \leq_p A$
- auf Grund der Transitivität gilt da $\forall L' \in$ NP: $L' \leq_p A \leq_p B$ auch $\forall L' \in$ NP: $L' \leq_p B$
- $\rightarrow B$ ist NP-vollständig

□

Satz 41

3SAT ist NP-vollständig.

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel
- wir modifizieren Klausel für Klausel aus ϕ wie folgt:

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel
- wir modifizieren Klausel für Klausel aus ϕ wie folgt:
  - hat die Klausel ≤ 3 Literale, verändere sie nicht

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel
- wir modifizieren Klausel für Klausel aus ϕ wie folgt:
  - hat die Klausel ≤ 3 Literale, verändere sie nicht
  - Ansonsten sei die Klausel Klausel $C_i = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel
- wir modifizieren Klausel für Klausel aus ϕ wie folgt:
  - hat die Klausel ≤ 3 Literale, verändere sie nicht
  - Ansonsten sei die Klausel Klausel $C_i = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$
  - wir ersetzen in ϕ' C_i durch:

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel
- wir modifizieren Klausel für Klausel aus ϕ wie folgt:
  - hat die Klausel ≤ 3 Literale, verändere sie nicht
  - Ansonsten sei die Klausel Klausel $C_i = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$
  - wir ersetzen in ϕ' C_i durch:

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

  - y_*^i sind neue Variablen, die die neuen Klauseln verbinden

Satz 41

3SAT ist NP-vollständig.

Beweis

- 3SAT$\in$ NP (Zeuge ist wie bei SAT die erfüllende Variablenbelegung)
- wir zeigen NP-Schwerheit durch CNF-SAT$\leq_p$3SAT
- sei ϕ Formel in CNF-SAT
- wir möchten Formel ϕ' erzeugen, so dass ϕ erfüllbar $\iff$ ϕ' erfüllbar und ϕ' ist in CNF mit ≤ 3 Literalen je Klausel
- wir modifizieren Klausel für Klausel aus ϕ wie folgt:
  - hat die Klausel ≤ 3 Literale, verändere sie nicht
  - Ansonsten sei die Klausel Klausel $C_i = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_k$
  - wir ersetzen in ϕ' C_i durch:

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

  - y_*^i sind neue Variablen, die die neuen Klauseln verbinden

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$

$\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0$

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

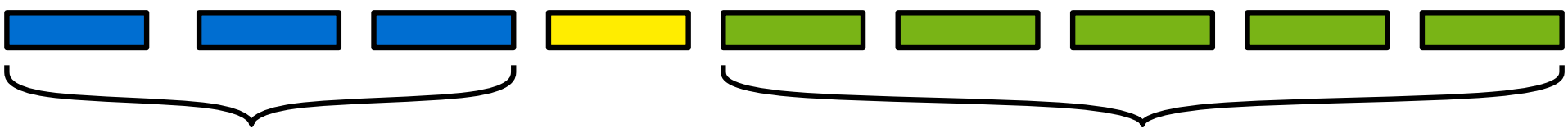
- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$

$\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

Schema



erfülle durch Setzen der hinteren y_*^i auf 1

erfülle durch Setzen der vorderen y_*^i auf 0

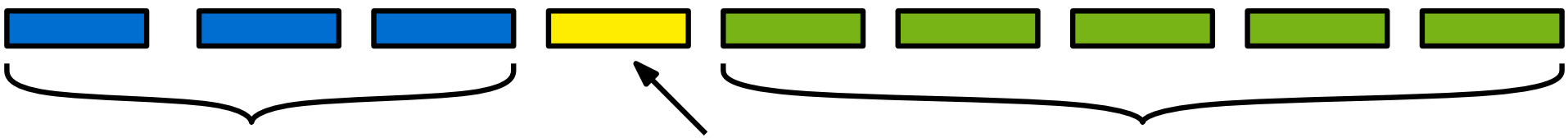
$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad 1 \qquad 0 \qquad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

Schema



erfülle durch Setzen der hinteren y_*^i auf 1

bleibt noch unerfüllt

erfülle durch Setzen der vorderen y_*^i auf 0

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

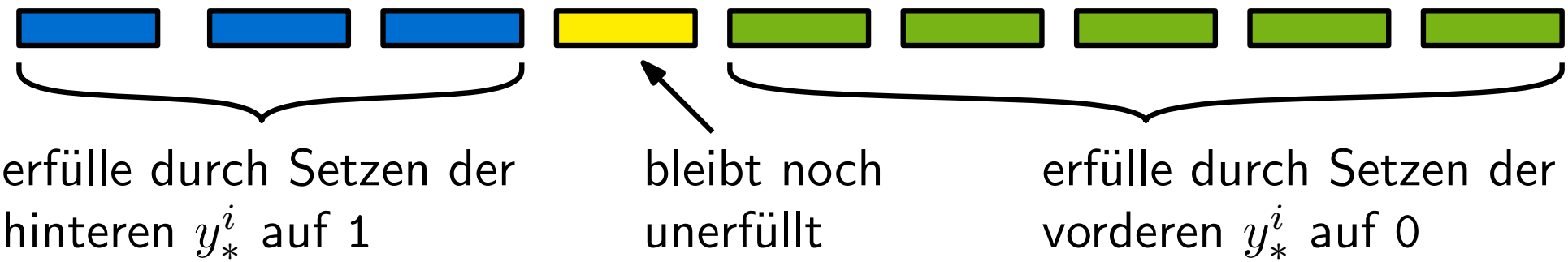
- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$

$\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

Schema



- ich kann C_i' genau dann erfüllen, wenn eines der Literale ℓ_* 1 ist

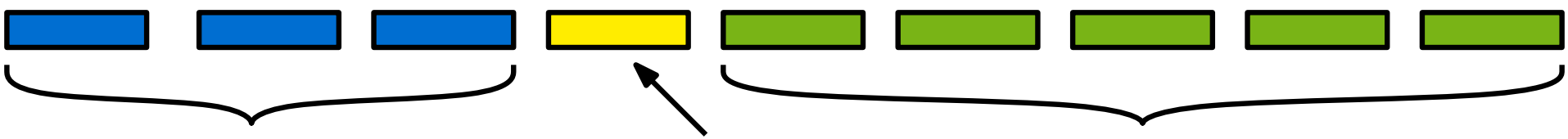
$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad 1 \qquad 0 \qquad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

Schema



erfülle durch Setzen der hinteren y_*^i auf 1

bleibt noch unerfüllt

erfülle durch Setzen der vorderen y_*^i auf 0

- ich kann C_i' genau dann erfüllen, wenn eines der Literale ℓ_* 1 ist
- C_i' ist genau dann erfüllbar, wenn C_i erfüllbar ist

$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

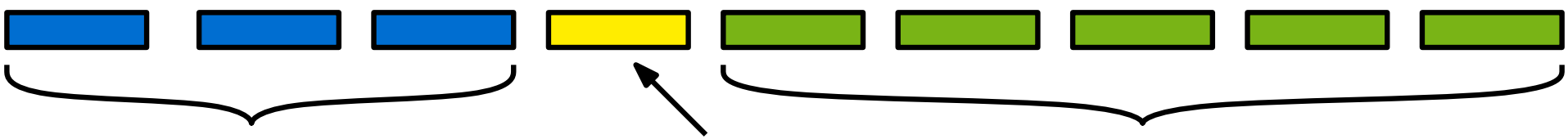
- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$

$\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0 \qquad\quad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

Schema



erfülle durch Setzen der hinteren y_*^i auf 1

bleibt noch unerfüllt

erfülle durch Setzen der vorderen y_*^i auf 0

- ich kann C_i' genau dann erfüllen, wenn eines der Literale ℓ_* 1 ist
- C_i' ist genau dann erfüllbar, wenn C_i erfüllbar ist
- ϕ' ist genau dann erfüllbar, wenn ϕ erfüllbar ist

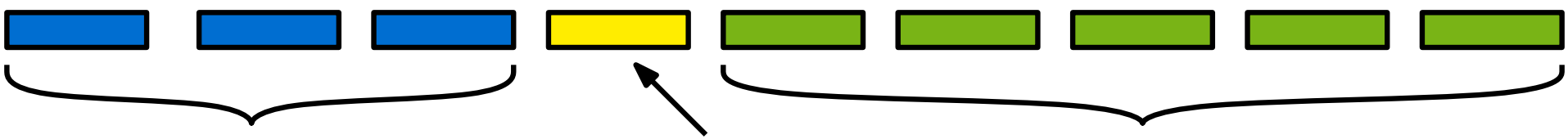
$$C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge \cdots \wedge (\neg y_{k-2}^i \vee \ell_{k-1} \vee \ell_k)$$

- durch die Belegung der y_*^i kann ich alle bis auf eine der neuen Klauseln erfüllen

Bsp. $C_i' = (\ell_1 \vee \ell_2 \vee y_1^i) \wedge (\neg y_1^i \vee \ell_3 \vee y_2^i) \wedge (\neg y_2^i \vee \ell_4 \vee y_3^i) \wedge (\neg y_3^i \vee \ell_5 \vee \ell_6)$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 1 \qquad 0 \qquad 1 \qquad 0 \qquad 1 \qquad 0$

- für jede Klausel gibt es eine Belegung der y_*^i die alle bis auf diese Klauseln erfüllt

Schema



erfülle durch Setzen der hinteren y_*^i auf 1

bleibt noch unerfüllt

erfülle durch Setzen der vorderen y_*^i auf 0

- ich kann C_i' genau dann erfüllen, wenn eines der Literale ℓ_* 1 ist
- C_i' ist genau dann erfüllbar, wenn C_i erfüllbar ist
- ϕ' ist genau dann erfüllbar, wenn ϕ erfüllbar ist
- Reduktion ist polyzeit

□