

**Script** generated by TTT

Title: Seidl: Theoretische\_Informatik  
(11.07.2013)

Date: Thu Jul 11 16:03:42 CEST 2013

Duration: 85:10 min

Pages: 102

**Lemma 5.16**

*Die Klassen  $P$  und  $NP$  sind unter polynomieller Reduzierbarkeit nach unten abgeschlossen:*

$$A \leq_p B \in P/NP \implies A \in P/NP$$

**Beweis:**

- Sei  $A \leq_p B$  mittels  $f$ , die von DTM  $M_f$  berechnet wird. Polynom  $p$  begrenzt Rechenzeit von  $M_f$ .
- Sei  $B \in P$  mittels DTM  $M$ . Polynom  $q$  begrenzt Rechenzeit von  $M$ .

Damit ist  $M_f; M$  polynomiell zeitbeschränkt in  $|w|$ :

- $M_f[w]$  macht  $\leq p(|w|)$  Schritte.
- Ausgabe  $f(w)$  von  $M_f$  hat Länge  $\leq |w| + p(|w|)$ .
- $M$  macht  $\leq q(|f(w)|) \leq q(|w| + p(|w|))$  Schritte ( $q$  monoton)

Daher macht  $(M_f; M)[w]$  maximal  $p(|w|) + q(|w| + p(|w|))$  Schritte, ein Polynom in  $|w|$ .

Analog:  $A \leq_p B \in NP \implies A \in NP$  □

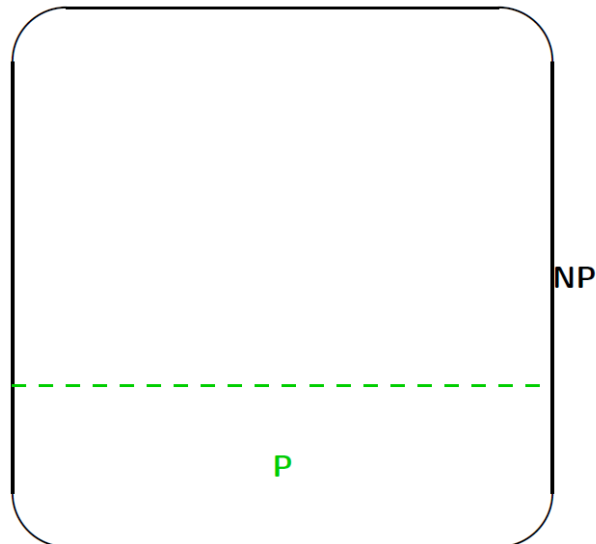
Ein Problem ist **NP-hart**,  
wenn es mindestens so hart wie alles in NP ist:

Ein Problem ist **NP-hart**,  
wenn es mindestens so hart wie alles in NP ist:

Ein Problem ist NP-hart,  
wenn es mindestens so hart wie alles in NP ist:

**Definition 5.17**

Eine Sprache  $L$  heißt NP-hart gdw  $A \leq_p L$  für alle  $A \in NP$ .



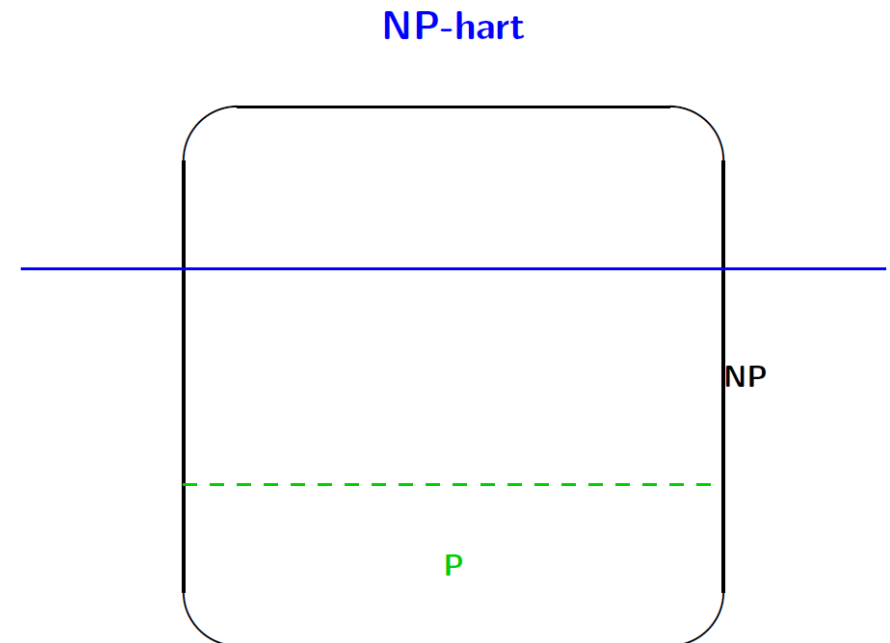
Ein Problem ist NP-hart,  
wenn es mindestens so hart wie alles in NP ist:

**Definition 5.17**

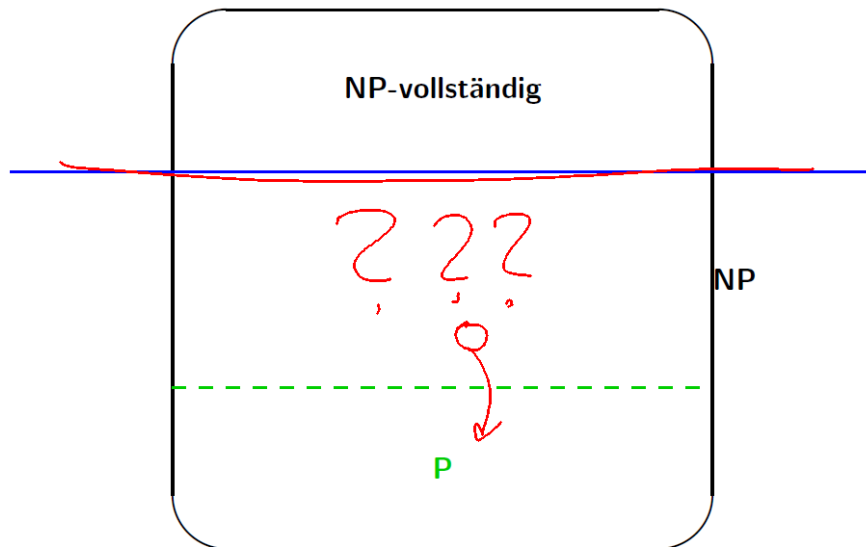
Eine Sprache  $L$  heißt NP-hart gdw  $A \leq_p L$  für alle  $A \in NP$ .

**Definition 5.18**

Eine Sprache  $L$  heißt NP-vollständig gdw  
 $L$  NP-hart ist und  $L \in NP$ .



## NP-hart



Wie man  $P \stackrel{?}{=} NP$  lösen kann:

[Lemma 5.19](#)

*Es gilt  $P=NP$  gdw ein NP-vollständiges Problem in  $P$  liegt.*

**Beweis:**

„ $\Rightarrow$ “: Falls  $P=NP$ , so liegt jedes NP-vollständige Problem in  $P$ .

Wie man  $P \stackrel{?}{=} NP$  lösen kann:

[Lemma 5.19](#)

*Es gilt  $P=NP$  gdw ein NP-vollständiges Problem in  $P$  liegt.*

Wie man  $P \stackrel{?}{=} NP$  lösen kann:

[Lemma 5.19](#)

*Es gilt  $P=NP$  gdw ein NP-vollständiges Problem in  $P$  liegt.*

**Beweis:**

„ $\Rightarrow$ “: Falls  $P=NP$ , so liegt jedes NP-vollständige Problem in  $P$ .

„ $\Leftarrow$ “: Sei  $L$  ein NP-vollständiges Problem in  $P$ .

Wie man  $P \stackrel{?}{=} NP$  lösen kann:

### Lemma 5.19

*Es gilt  $P=NP$  gdw ein NP-vollständiges Problem in  $P$  liegt.*

#### Beweis:

„ $\Rightarrow$ “: Falls  $P=NP$ , so liegt jedes NP-vollständige Problem in  $P$ .

„ $\Leftarrow$ “: Sei  $L$  ein NP-vollständiges Problem in  $P$ .

Dann gilt  $P \supseteq NP$ :

Ist  $A \in NP$ , so gilt  $A \leq_p L$  (da  $L$  NP-hart)

und nach Lemma 5.16  $A \in P$  (da  $L \in P$ ). □

## Aussagenlogik

Wie man  $P \stackrel{?}{=} NP$  lösen kann:

### Lemma 5.19

*Es gilt  $P=NP$  gdw ein NP-vollständiges Problem in  $P$  liegt.*

#### Beweis:

„ $\Rightarrow$ “: Falls  $P=NP$ , so liegt jedes NP-vollständige Problem in  $P$ .

„ $\Leftarrow$ “: Sei  $L$  ein NP-vollständiges Problem in  $P$ .

Dann gilt  $P \supseteq NP$ :

Ist  $A \in NP$ , so gilt  $A \leq_p L$  (da  $L$  NP-hart)

und nach Lemma 5.16  $A \in P$  (da  $L \in P$ ). □

Starke Vermutung:

- $P \neq NP$
- dh kein NP-vollständiges Problem ist in  $P$ .

Aber gibt es überhaupt NP-vollständige Probleme?

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

$$(1 \wedge x) \vee 0 \equiv x$$

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  ~~$(\neg x \wedge y) \vee (x \wedge \neg z)$~~

Konvention: man darf äußerste Klammern weglassen

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

Konvention: man darf äußerste Klammern weglassen  
und  $\wedge$  bindet stärker als  $\vee$ :  $x \wedge y \vee z$  ist Abk. für  $(x \wedge y) \vee z$

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

Konvention: man darf äußerste Klammern weglassen  
und  $\wedge$  bindet stärker als  $\vee$ :  $x \wedge y \vee z$  ist Abk. für  $(x \wedge y) \vee z$

Semantik der Aussagenlogik:

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

Konvention: man darf äußerste Klammern weglassen  
und  $\wedge$  bindet stärker als  $\vee$ :  $x \wedge y \vee z$  ist Abk. für  $(x \wedge y) \vee z$

Semantik der Aussagenlogik:

- Eine **Belegung** ist eine Funktion von Variablen auf  $\{0, 1\}$ .  
Bsp:  $\sigma = \{x \mapsto 0, y \mapsto 1, z \mapsto 0, \dots\}$
- Belegungen werde mittels Wahrheitstabellen auf Formeln erweitert.

## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

Konvention: man darf äußerste Klammern weglassen  
und  $\wedge$  bindet stärker als  $\vee$ :  $x \wedge y \vee z$  ist Abk. für  $(x \wedge y) \vee z$

Semantik der Aussagenlogik:

- Eine **Belegung** ist eine Funktion von Variablen auf  $\{0, 1\}$ .

## Aussagenlogik

Syntax der Aussagenlogik:

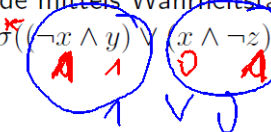
Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$   
Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

Konvention: man darf äußerste Klammern weglassen  
und  $\wedge$  bindet stärker als  $\vee$ :  $x \wedge y \vee z$  ist Abk. für  $(x \wedge y) \vee z$

Semantik der Aussagenlogik:

- Eine **Belegung** ist eine Funktion von Variablen auf  $\{0, 1\}$ .  
Bsp:  $\sigma = \{x \mapsto 0, y \mapsto 1, z \mapsto 0, \dots\}$
- Belegungen werde mittels Wahrheitstabellen auf Formeln erweitert. Bsp:  $\sigma((\neg x \wedge y) \vee (x \wedge \neg z)) = 1$



## Aussagenlogik

Syntax der Aussagenlogik:

Formeln:  $F \rightarrow \neg F \mid (F \wedge F) \mid (F \vee F) \mid X$

Variablen:  $X \rightarrow x \mid y \mid z \mid \dots$

Bsp:  $((\neg x \wedge y) \vee (x \wedge \neg z))$

Konvention: man darf äußerste Klammern weglassen  
und  $\wedge$  bindet stärker als  $\vee$ :  $x \wedge y \vee z$  ist Abk. für  $(x \wedge y) \vee z$

Semantik der Aussagenlogik:

- Eine **Belegung** ist eine Funktion von Variablen auf  $\{0, 1\}$ .  
Bsp:  $\sigma = \{x \mapsto 0, y \mapsto 1, z \mapsto 0, \dots\}$
- Belegungen werde mittels Wahrheitstabellen auf Formeln erweitert. Bsp:  $\sigma((\neg x \wedge y) \vee (x \wedge \neg z)) = 1$
- Eine Formel  $F$  ist **erfüllbar** gdw es eine Belegung  $\sigma$  gibt mit  $\sigma(F) = 1$

## SAT

**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

Praktische Bedeutung von SAT:

Äquivalenztest von Schaltungen

## SAT

**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

## SAT

**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

Praktische Bedeutung von SAT:

Äquivalenztest von Schaltungen (und vieles mehr)

## SAT


**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

Praktische Bedeutung von SAT:

Äquivalenztest von Schaltungen (und vieles mehr)

### Fakt 5.20

Zwei Formeln  $F_1$  und  $F_2$  sind genau dann äquivalent ( $F_1 \leftrightarrow F_2$ ), wenn  $(F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2)$  nicht erfüllbar ist. 

## SAT

**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

Praktische Bedeutung von SAT:

Äquivalenztest von Schaltungen (und vieles mehr)

### Fakt 5.20

Zwei Formeln  $F_1$  und  $F_2$  sind genau dann äquivalent ( $F_1 \leftrightarrow F_2$ ), wenn  $(F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2)$  nicht erfüllbar ist.

### Lemma 5.21

$SAT \in NP$

### Beweis:

Belegungen sind Zertifikate, die in polynomieller Zeit geprüft werden können:

## SAT

**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

Praktische Bedeutung von SAT:

Äquivalenztest von Schaltungen (und vieles mehr)

### Fakt 5.20

Zwei Formeln  $F_1$  und  $F_2$  sind genau dann äquivalent ( $F_1 \leftrightarrow F_2$ ), wenn  $(F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2)$  nicht erfüllbar ist.

### Lemma 5.21

$SAT \in NP$

## SAT

**Gegeben:** Eine aussagenlogische Formel  $F$ .

**Problem:** Ist  $F$  erfüllbar?

Praktische Bedeutung von SAT:

Äquivalenztest von Schaltungen (und vieles mehr)

### Fakt 5.20

Zwei Formeln  $F_1$  und  $F_2$  sind genau dann äquivalent ( $F_1 \leftrightarrow F_2$ ), wenn  $(F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2)$  nicht erfüllbar ist.

### Lemma 5.21

$SAT \in NP$

### Beweis:

Belegungen sind Zertifikate, die in polynomieller Zeit geprüft werden können: Es gibt eine DTM, die bei Eingabe einer Formel  $F$  und einer Belegung  $\sigma$  für die Variablen in  $F$ , in polynomieller Zeit  $\sigma(F)$  berechnet. □



Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Sei  $A \in NP$ . Wir zeigen  $A \leq_p SAT$ .

Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Sei  $A \in NP$ . Wir zeigen  $A \leq_p SAT$ .

Da  $A \in NP$  gibt es NTM  $M$  mit  $A = L(M)$

### Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

#### Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Sei  $A \in NP$ . Wir zeigen  $A \leq_p SAT$ .

Da  $A \in NP$  gibt es NTM  $M$  mit  $A = L(M)$  und  
Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

$w \in A$

### Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

#### Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Sei  $A \in NP$ . Wir zeigen  $A \leq_p SAT$ .

Da  $A \in NP$  gibt es NTM  $M$  mit  $A = L(M)$  und  
Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

In polynomieller Zeit.

### Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

#### Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Sei  $A \in NP$ . Wir zeigen  $A \leq_p SAT$ .

Da  $A \in NP$  gibt es NTM  $M$  mit  $A = L(M)$  und  
Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

### Satz 5.22 (Cook 1971)

*SAT ist NP-vollständig.*

#### Beweis:

Da  $SAT \in NP$ , bleibt noch zu zeigen,  $SAT$  ist NP-hart.

Sei  $A \in NP$ . Wir zeigen  $A \leq_p SAT$ .

Da  $A \in NP$  gibt es NTM  $M$  mit  $A = L(M)$  und  
Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

In polynomieller Zeit. So dass  $w \in L(M) \Leftrightarrow F \in SAT$ .

### Satz 5.22 (Cook 1971)

SAT ist NP-vollständig.

#### Beweis:

Da SAT  $\in$  NP, bleibt noch zu zeigen, SAT ist NP-hart.

Sei  $A \in$  NP. Wir zeigen  $A \leq_p$  SAT.

Da  $A \in$  NP gibt es NTM  $M$  mit  $A = L(M)$  und

Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

In polynomieller Zeit. So dass  $w \in L(M) \Leftrightarrow F \in SAT$ .

Die Variablen beschreiben das mögliche Verhalten von  $M[w]$ :

### Satz 5.22 (Cook 1971)

SAT ist NP-vollständig.

#### Beweis:

Da SAT  $\in$  NP, bleibt noch zu zeigen, SAT ist NP-hart.

Sei  $A \in$  NP. Wir zeigen  $A \leq_p$  SAT.

Da  $A \in$  NP gibt es NTM  $M$  mit  $A = L(M)$  und

Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

In polynomieller Zeit. So dass  $w \in L(M) \Leftrightarrow F \in SAT$ .

Die Variablen beschreiben das mögliche Verhalten von  $M[w]$ :

$zust_{t,q}$	$t = 0, \dots, p(n)$ $q \in Q$	$zust_{t,q} = 1 \Leftrightarrow$ Zustand nach $t$ Schritten ist $q$
$pos_{t,i}$	$t = 0, \dots, p(n)$ $i = -p(n), \dots, p(n)$	$pos_{t,i} = 1 \Leftrightarrow$ Kopfposition nach $t$ Schritten ist $i$

### Satz 5.22 (Cook 1971)

SAT ist NP-vollständig.

#### Beweis:

Da SAT  $\in$  NP, bleibt noch zu zeigen, SAT ist NP-hart.

Sei  $A \in$  NP. Wir zeigen  $A \leq_p$  SAT.

Da  $A \in$  NP gibt es NTM  $M$  mit  $A = L(M)$  und

Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

In polynomieller Zeit. So dass  $w \in L(M) \Leftrightarrow F \in SAT$ .

Die Variablen beschreiben das mögliche Verhalten von  $M[w]$ :

$zust_{t,q}$	$t = 0, \dots, p(n)$ $q \in Q$	$zust_{t,q} = 1 \Leftrightarrow$ Zustand nach $t$ Schritten ist $q$
--------------	-----------------------------------	--

### Satz 5.22 (Cook 1971)

SAT ist NP-vollständig.

#### Beweis:

Da SAT  $\in$  NP, bleibt noch zu zeigen, SAT ist NP-hart.

Sei  $A \in$  NP. Wir zeigen  $A \leq_p$  SAT.

Da  $A \in$  NP gibt es NTM  $M$  mit  $A = L(M)$  und

Polynom  $p$  mit  $ntime_M(w) \leq p(|w|)$ .

Reduktion bildet  $w = x_1 \dots x_n \in \Sigma^*$  auf eine Formel  $F$  ab.

In polynomieller Zeit. So dass  $w \in L(M) \Leftrightarrow F \in SAT$ .

Die Variablen beschreiben das mögliche Verhalten von  $M[w]$ :

$zust_{t,q}$	$t = 0, \dots, p(n)$ $q \in Q$	$zust_{t,q} = 1 \Leftrightarrow$ Zustand nach $t$ Schritten ist $q$
$pos_{t,i}$	$t = 0, \dots, p(n)$ $i = -p(n), \dots, p(n)$	$pos_{t,i} = 1 \Leftrightarrow$ Kopfposition nach $t$ Schritten ist $i$
$band_{t,i,a}$	$t = 0, \dots, p(n)$ $i = -p(n), \dots, p(n)$ $a \in \Gamma$	$band_{t,i,a} = 1 \Leftrightarrow$ Bandinhalt nach $t$ Schritten auf Bandposition $i$ ist Zeichen $a$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k})$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

Beweis (Forts.):

$$T_1 := \bigwedge_{t,q,i,a} [\text{zust}_{t,q} \wedge \text{pos}_{t,i} \wedge \text{band}_{t,i,a} \\ \rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (\text{zust}_{t+1,q'} \wedge \text{pos}_{t+1,i+y} \wedge \text{band}_{t+1,i,a'})]$$

$$T_2 := \bigwedge_{t,i,a} ((\neg \text{pos}_{t,i} \wedge \text{band}_{t,i,a}) \rightarrow \text{band}_{t+1,i,a})$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)})$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

$$A := \text{zust}_{0,q_1} \wedge \text{pos}_{0,1}$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

$$A := \text{zust}_{0,q_1}$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

$$A := \text{zust}_{0,q_1} \wedge \text{pos}_{0,1} \wedge \bigwedge_{j=1}^n \text{band}_{0,j,x_j}$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

$$A := \text{zust}_{0,q_1} \wedge \text{pos}_{0,1} \wedge \bigwedge_{j=1}^n \text{band}_{0,j,x_j} \wedge \bigwedge_{j=-p(n)}^0 \text{band}_{0,j,\square} \wedge \bigwedge_{j=n+1}^{p(n)} \text{band}_{0,j,\square}$$

Beweis (Forts.):

$$T_1 := \bigwedge_{t,q,i,a} [\text{zust}_{t,q} \wedge \text{pos}_{t,i} \wedge \text{band}_{t,i,a}$$

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$F := R \wedge A \wedge T_1 \wedge T_2 \wedge E$$

$$R := \bigwedge_t [G(\text{zust}_{t,q_1}, \dots, \text{zust}_{t,q_k}) \wedge G(\text{pos}_{t,-p(n)}, \dots, \text{pos}_{t,p(n)}) \wedge \bigwedge_i G(\text{band}_{t,i,a_1}, \dots, \text{band}_{t,i,a_l})]$$

$$A := \text{zust}_{0,q_1} \wedge \text{pos}_{0,1} \wedge \bigwedge_{j=1}^n \text{band}_{0,j,x_j} \wedge \bigwedge_{j=-p(n)}^0 \text{band}_{0,j,\square} \wedge \bigwedge_{j=n+1}^{p(n)} \text{band}_{0,j,\square}$$

Beweis (Forts.):

$$T_1 := \bigwedge_{t,q,i,a} [\text{zust}_{t,q} \wedge \text{pos}_{t,i} \wedge \text{band}_{t,i,a} \rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (\text{zust}_{t+1,q'} \wedge \text{pos}_{t+1,i+y} \wedge \text{band}_{t+1,i,a'})]$$

Beweis (Forts.):

$$x \rightarrow y \equiv \neg x \vee y$$

$$\begin{aligned} T_1 &:= \bigwedge_{t,q,i,a} [zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \\ &\rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (zust_{t+1,q'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})] \\ T_2 &:= \bigwedge_{t,i,a} ((\neg pos_{t,i} \wedge band_{t,i,a}) \rightarrow band_{t+1,i,a}) \end{aligned}$$

Beweis (Forts.):

$$\begin{aligned} T_1 &:= \bigwedge_{t,q,i,a} [zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \\ &\rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (zust_{t+1,q'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})] \\ T_2 &:= \bigwedge_{t,i,a} ((\neg pos_{t,i} \wedge band_{t,i,a}) \rightarrow band_{t+1,i,a}) \\ E &:= \bigvee_t \bigvee_{q \in F} zust_{t,q} \end{aligned}$$

$$G(v_1, \dots, v_r) := \left( \bigvee_{i=1}^r v_i \right) \wedge \left( \bigwedge_{i=1}^{r-1} \bigwedge_{j=i+1}^r \neg(v_i \wedge v_j) \right)$$

□

Beweis (Forts.):

$$\begin{aligned} T_1 &:= \bigwedge_{t,q,i,a} [zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \\ &\rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (zust_{t+1,q'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})] \\ T_2 &:= \bigwedge_{t,i,a} ((\neg pos_{t,i} \wedge band_{t,i,a}) \rightarrow band_{t+1,i,a}) \\ E &:= \bigvee_t \bigvee_{q \in F} zust_{t,q} \end{aligned}$$

Beweis (Forts.):

$$\begin{aligned} T_1 &:= \bigwedge_{t,q,i,a} [zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \\ &\rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (zust_{t+1,q'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})] \\ T_2 &:= \bigwedge_{t,i,a} ((\neg pos_{t,i} \wedge band_{t,i,a}) \rightarrow band_{t+1,i,a}) \\ E &:= \bigvee_t \bigvee_{q \in F} zust_{t,q} \end{aligned}$$

Beweis (Forts.):

$$\begin{aligned}
 T_1 &:= \bigwedge_{t,q,i,a} [zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \\
 &\quad \rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (zust_{t+1,q'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})] \\
 T_2 &:= \bigwedge_{t,i,a} ((\neg pos_{t,i} \wedge band_{t,i,a}) \rightarrow band_{t+1,i,a})
 \end{aligned}$$

Beweis (Forts.):

$$\begin{aligned}
 T_1 &:= \bigwedge_{t,q,i,a} [zust_{t,q} \wedge pos_{t,i} \wedge band_{t,i,a} \\
 &\quad \rightarrow \bigvee_{(q',a',y) \in \delta(q,a)} (zust_{t+1,q'} \wedge pos_{t+1,i+y} \wedge band_{t+1,i,a'})] \\
 T_2 &:= \bigwedge_{t,i,a} ((\neg pos_{t,i} \wedge band_{t,i,a}) \rightarrow band_{t+1,i,a}) \\
 E &:= \bigvee_t \bigvee_{q \in F} zust_{t,q}
 \end{aligned}$$

$$G(v_1, \dots, v_r) := \left( \bigvee_{i=1}^r v_i \right) \wedge \left( \bigwedge_{i=1}^{r-1} \bigwedge_{j=i+1}^r \neg(v_i \wedge v_j) \right)$$

□

Beweis (Forts.):

Sei  $Q = \{q_1, \dots, q_k\}$  und  $\Gamma = \{a_1, \dots, a_l\}$ .

$$\begin{aligned}
 F &:= R \wedge A \wedge T_1 \wedge T_2 \wedge E \\
 R &:= \bigwedge_t [G(zust_{t,q_1}, \dots, zust_{t,q_k}) \wedge G(pos_{t,-p(n)}, \dots, pos_{t,p(n)}) \wedge \\
 &\quad \bigwedge_i G(band_{t,i,a_1}, \dots, band_{t,i,a_l})] \\
 A &:= zust_{0,q_1} \wedge pos_{0,1} \wedge \bigwedge_{j=1}^n band_{0,j,x_j} \wedge \\
 &\quad \bigwedge_{j=-p(n)}^0 band_{0,j,\square} \wedge \bigwedge_{j=n+1}^{p(n)} band_{0,j,\square}
 \end{aligned}$$

Von der Lösbarkeit zur Lösung



### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
```

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
```

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
    if  $F[x_i := 0] \in \text{SAT}$  then  $b := 0$  else  $b := 1$ 
```

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
    if  $F[x_i := 0] \in \text{SAT}$  then  $b := 0$  else  $b := 1$ 
    output( $x_i$  "="  $b$ )
     $F := F[x_i := b]$ 
```

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
    if  $F[x_i := 0] \in \text{SAT}$  then  $b := 0$  else  $b := 1$ 
    output( $x_i$  "="  $b$ )
     $F := F[x_i := b]$ 
```

wobei  $F[x := b] = F$  mit  $x$  ersetzt durch  $b$ .

Entscheidung von SAT in Zeit  $O(f(n))$

$\implies$  Berechnung einer erf. Bel. in Zeit  $O(n \cdot (f(n) + n))$

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
    if  $F[x_i := 0] \in \text{SAT}$  then  $b := 0$  else  $b := 1$ 
    output( $x_i$  "="  $b$ )
     $F := F[x_i := b]$ 
```

wobei  $F[x := b] = F$  mit  $x$  ersetzt durch  $b$ .

Entscheidung von SAT in Zeit  $O(f(n))$

### Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
    if  $F[x_i := 0] \in \text{SAT}$  then  $b := 0$  else  $b := 1$ 
    output( $x_i$  "="  $b$ )
     $F := F[x_i := b]$ 
```

wobei  $F[x := b] = F$  mit  $x$  ersetzt durch  $b$ .

Entscheidung von SAT in Zeit  $O(f(n))$

$\implies$  Berechnung einer erf. Bel. in Zeit  $O(n \cdot (f(n) + n))$   
(falls es eine gibt.)

$f(n)$  polynomiell  $\implies n \cdot (f(n) + n)$  polynomiell

## Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT reduziert werden:

Sei  $F$  eine Formel mit den Variablen  $x_1, \dots, x_k$ :

```
if  $F \notin \text{SAT}$  then output("nicht lösbar")
else
  for  $i := 1$  to  $k$  do
    if  $F[x_i := 0] \in \text{SAT}$  then  $b := 0$  else  $b := 1$ 
    output( $x_i$  "="  $b$ )
     $F := F[x_i := b]$ 
```

wobei  $F[x := b] = F$  mit  $x$  ersetzt durch  $b$ .

Entscheidung von SAT in Zeit  $O(f(n))$

$\implies$  Berechnung einer erf. Bel. in Zeit  $O(n \cdot (f(n) + n))$   
(falls es eine gibt.)

$f(n)$  polynomiell  $\implies n \cdot (f(n) + n)$  polynomiell

$f(n)$  exponentiell  $\implies n \cdot (f(n) + n)$  exponentiell

### Bemerkungen:

- Die Reduzierung der Lösungsberechnung auf SAT ist eine rein theoretische Konstruktion.
- Sie zeigt, dass man sich auf SAT beschränken kann, wenn man nur an polynomiell/exponentiell interessiert ist.

### Bemerkungen:

- Die Reduzierung der Lösungsberechnung auf SAT ist eine rein theoretische Konstruktion.

## Von NP-hart zu „NP-leicht“

- Bis vor 10 Jahren:

NP-vollständig = Todesurteil

### Von NP-hart zu „NP-leicht“

- Bis vor 10 Jahren:  
NP-vollständig = Todesurteil
- In den letzten 10 Jahren:  
Spektakuläre Fortschritte bei *Implementierung* von  
SAT-Lösern (*SAT-solver*):

### Von NP-hart zu „NP-leicht“

- Bis vor 10 Jahren:  
NP-vollständig = Todesurteil
- In den letzten 10 Jahren:  
Spektakuläre Fortschritte bei *Implementierung* von  
SAT-Lösern (*SAT-solver*): <http://satcompetition.org>  
Stand der Kunst: Erfüllbar: bis  $10^5$  Variablen  
Unerfüllbar: bis  $10^3$  Variablen

### Von NP-hart zu „NP-leicht“

- Bis vor 10 Jahren:  
NP-vollständig = Todesurteil
- In den letzten 10 Jahren:  
Spektakuläre Fortschritte bei *Implementierung* von  
SAT-Lösern (*SAT-solver*): <http://satcompetition.org>  
Stand der Kunst: Erfüllbar: bis  $10^5$  Variablen

### Von NP-hart zu „NP-leicht“

- Bis vor 10 Jahren:  
NP-vollständig = Todesurteil
- In den letzten 10 Jahren:  
Spektakuläre Fortschritte bei *Implementierung* von  
SAT-Lösern (*SAT-solver*): <http://satcompetition.org>  
Stand der Kunst: Erfüllbar: bis  $10^5$  Variablen  
Unerfüllbar: bis  $10^3$  Variablen
- Jetzt:  
NP-vollständig = Hoffnung durch SAT

## Von NP-hart zu „NP-leicht“

- Bis vor 10 Jahren:  
NP-vollständig = Todesurteil
- In den letzten 10 Jahren:  
Spektakuläre Fortschritte bei *Implementierung* von  
SAT-Lösern (*SAT-solver*): <http://satcompetition.org>  
Stand der Kunst: Erfüllbar: bis  $10^5$  Variablen  
Unerfüllbar: bis  $10^3$  Variablen
- Jetzt:  
NP-vollständig = Hoffnung durch SAT
- Paradigma:  
SAT (Logik!) als universelle Sprache  
zur Kodierung kombinatorischer Probleme

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

## Von NP-hart zu „NP-leicht“

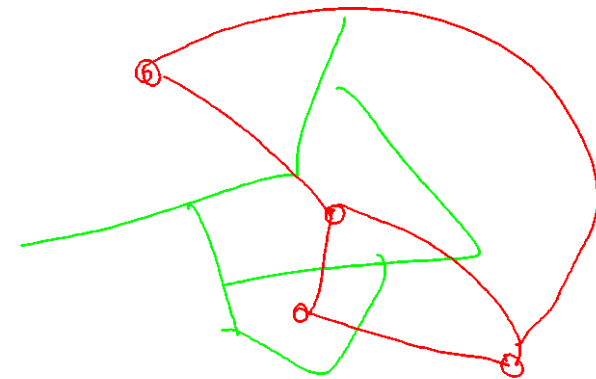
- Bis vor 10 Jahren:  
NP-vollständig = Todesurteil
- In den letzten 10 Jahren:  
Spektakuläre Fortschritte bei *Implementierung* von  
SAT-Lösern (*SAT-solver*): <http://satcompetition.org>  
Stand der Kunst: Erfüllbar: bis  $10^5$  Variablen  
Unerfüllbar: bis  $10^3$  Variablen
- Jetzt:  
NP-vollständig = Hoffnung durch SAT
- Paradigma:  
SAT (Logik!) als universelle Sprache  
zur Kodierung kombinatorischer Probleme  
Reduktion auf SAT manchmal schneller als problemspezifische  
Löser!

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine  
benachbarten Knoten gleich gefärbt sind?



Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

- Variablen =  $V \times \{1, 2, 3\}$ .

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

- Variablen =  $V \times \{1, 2, 3\}$ . Notation:  $x_{vi}$

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

- Variablen =  $V \times \{1, 2, 3\}$ . Notation:  $x_{vi}$
- Interpretation:  $x_{vi} = 1$  gdw Knoten  $v$  hat Farbe  $i$

Instanz von SAT:

$$\bigwedge_{v \in V} G(x_{v1}, x_{v2}, x_{v3})$$

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

### 3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

- Variablen =  $V \times \{1, 2, 3\}$ . Notation:  $x_{vi}$
- Interpretation:  $x_{vi} = 1$  gdw Knoten  $v$  hat Farbe  $i$

Instanz von SAT:

$$\bigwedge_{v \in V} G(x_{v1}, x_{v2}, x_{v3}) \wedge \bigwedge_{(u,v) \in E} \neg(x_{u1} \wedge x_{v1} \vee x_{u2} \wedge x_{v2} \vee x_{u3} \wedge x_{v3})$$

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

### 3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

- Variablen =  $V \times \{1, 2, 3\}$ . Notation:  $x_{vi}$
- Interpretation:  $x_{vi} = 1$  gdw Knoten  $v$  hat Farbe  $i$

Instanz von SAT:

$$\bigwedge_{v \in V} G(x_{v1}, x_{v2}, x_{v3}) \wedge \bigwedge_{(u,v) \in E} \neg(x_{u1} \wedge x_{v1} \vee x_{u2} \wedge x_{v2} \vee x_{u3} \wedge x_{v3})$$

### Bemerkungen

- Zeigt  $3COL \leq_p SAT$  und damit  $3COL \in NP$ .
- Zeigt nicht, dass 3COL NP-vollständig ist.

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

### 3COL

**Gegeben:** Ein ungerichteter Graph

**Problem:** Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph  $G = (V, E)$  auf SAT:

- Variablen =  $V \times \{1, 2, 3\}$ . Notation:  $x_{vi}$
- Interpretation:  $x_{vi} = 1$  gdw Knoten  $v$  hat Farbe  $i$

Instanz von SAT:

$$\bigwedge_{v \in V} G(x_{v1}, x_{v2}, x_{v3}) \wedge \bigwedge_{(u,v) \in E} \neg(x_{u1} \wedge x_{v1} \vee x_{u2} \wedge x_{v2} \vee x_{u3} \wedge x_{v3})$$

### Bemerkungen

- Zeigt  $3COL \leq_p SAT$  und damit  $3COL \in NP$ .

Ähnlich direkt polynomiell auf SAT reduzierbar:

- HAMILTON
- SUDOKU
- ...

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**  
gdw sie später noch gelesen wird



Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf  $k$ -Färbbarkeit:

Variable = Knoten

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf  $k$ -Färbbarkeit:

Variable = Knoten

$u$  und  $v$  verbunden =  $u \neq v$  und es gibt einen Programmpunkt, an dem  $u$  und  $v$  lebendig sind

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf  $k$ -Färbbarkeit:

Variable = Knoten

$u$  und  $v$  verbunden =  $u \neq v$  und es gibt einen Programmpunkt, an dem  $u$  und  $v$  lebendig sind

Farbe = Register

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf  $k$ -Färbbarkeit:

Variable = Knoten  
 $u$  und  $v$  verbunden =  $u \neq v$  und es gibt einen Programmpunkt, an dem  $u$  und  $v$  lebendig sind  
Farbe = Register  
 $k$ -Färbung = Zuordnung eines Registers zu jeder Variablen

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf  $k$ -Färbbarkeit:

Variable = Knoten  
 $u$  und  $v$  verbunden =  $u \neq v$  und es gibt einen Programmpunkt, an dem  $u$  und  $v$  lebendig sind  
Farbe = Register  
 $k$ -Färbung = Zuordnung eines Registers zu jeder Variablen

Sowohl  $k$ -Färbbarkeit als auch Registerverteilung ist für  $k \geq 3$  NP-vollständig.

Eine Anwendung von  $k$ -Färbbarkeit: **Registerverteilung**

*Kann man in einem Programmstück  $n$  Variablen so auf  $k$  Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie **lebendig** ist?*

Variable ist an einem Punkt **lebendig**

gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf  $k$ -Färbbarkeit:

Variable = Knoten  
 $u$  und  $v$  verbunden =  $u \neq v$  und es gibt einen Programmpunkt, an dem  $u$  und  $v$  lebendig sind  
Farbe = Register  
 $k$ -Färbung = Zuordnung eines Registers zu jeder Variablen

Sowohl  $k$ -Färbbarkeit als auch Registerverteilung ist für  $k \geq 3$  NP-vollständig.

Mehr Information: Vorlesung *Programmoptimierung*