# VL-14: Der Satz von Cook & Levin

(Berechenbarkeit und Komplexität, WS 2018)

Gerhard Woeginger

WS 2018, RWTH

# Organisatorisches

- Nächste Vorlesung:
  Donnerstag, Januar 11, 12:30–14:00 Uhr, Aula

- Webseite:
  http://algo.rwth-aachen.de/Lehre/WS1819/BuK.php
  **($\longrightarrow$ Arbeitsheft zur Berechenbarkeit)**

# Wiederholung

# Wdh.: Alternative Charakterisierung von NP

**Satz (Zertifikat Charakterisierung von NP)**

Eine Sprache $L \subseteq \Sigma^*$ liegt genau dann in NP,
wenn es einen polynomiellen (deterministischen) Algorithmus $V$
und ein Polynom $p$ mit der folgenden Eigenschaft gibt:

$$x \in L \iff \exists y \in \{0,1\}^*,\ |y| \leq p(|x|) :\ V \text{ akzeptiert } y\#x$$

Anmerkungen:

- Der polynomielle Algorithmus $V$ heisst auch Verifizierer
- Das Wort $y \in \{0,1\}^*$ heisst auch Zertifikat

# Wdh.: Komplexitätsklasse EXPTIME

**Definition: Komplexitätsklasse EXPTIME**

EXPTIME ist die Klasse aller Entscheidungsprobleme, die durch eine DTM $M$ entschieden werden, deren Worst Case Laufzeit durch $2^{q(n)}$ mit einem Polynom $q$ beschränkt ist,

Laufzeit-Beispiele: $2^{\sqrt{n}}$, $2^n$, $3^n$, $n!$, $n^n$. Aber nicht: $2^{2^n}$

**Satz**

$P \subseteq NP \subseteq EXPTIME$

# Wdh.: Polynomielle Reduktionen

**Definition**

Es seien $L_1$ und $L_2$ Sprachen über $\Sigma_1$ bzw. $\Sigma_2$.
Dann ist $L_1$ polynomiell reduzierbar auf $L_2$ (mit der Notation $L_1 \leq_p L_2$),
wenn eine polynomiell berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ existiert,
so dass für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.

**Satz**

Falls $L_1 \leq_p L_2$ und falls $L_2 \in P$, so gilt $L_1 \in P$

**Satz**

COLORING $\leq_p$ SAT

# Wdh.: Ex-Cover $\leq_p$ SAT

**Problem: Exact Cover (Ex-Cover)**

Eingabe: Eine endliche Menge $X$; Teilmengen $S_1, \ldots, S_m$ von $X$

Frage: Existiert eine Indexmenge $I \subseteq \{1, \ldots, m\}$,
sodass die Mengen $S_i$ mit $i \in I$ eine Partition von $X$ bilden?

**Übung**

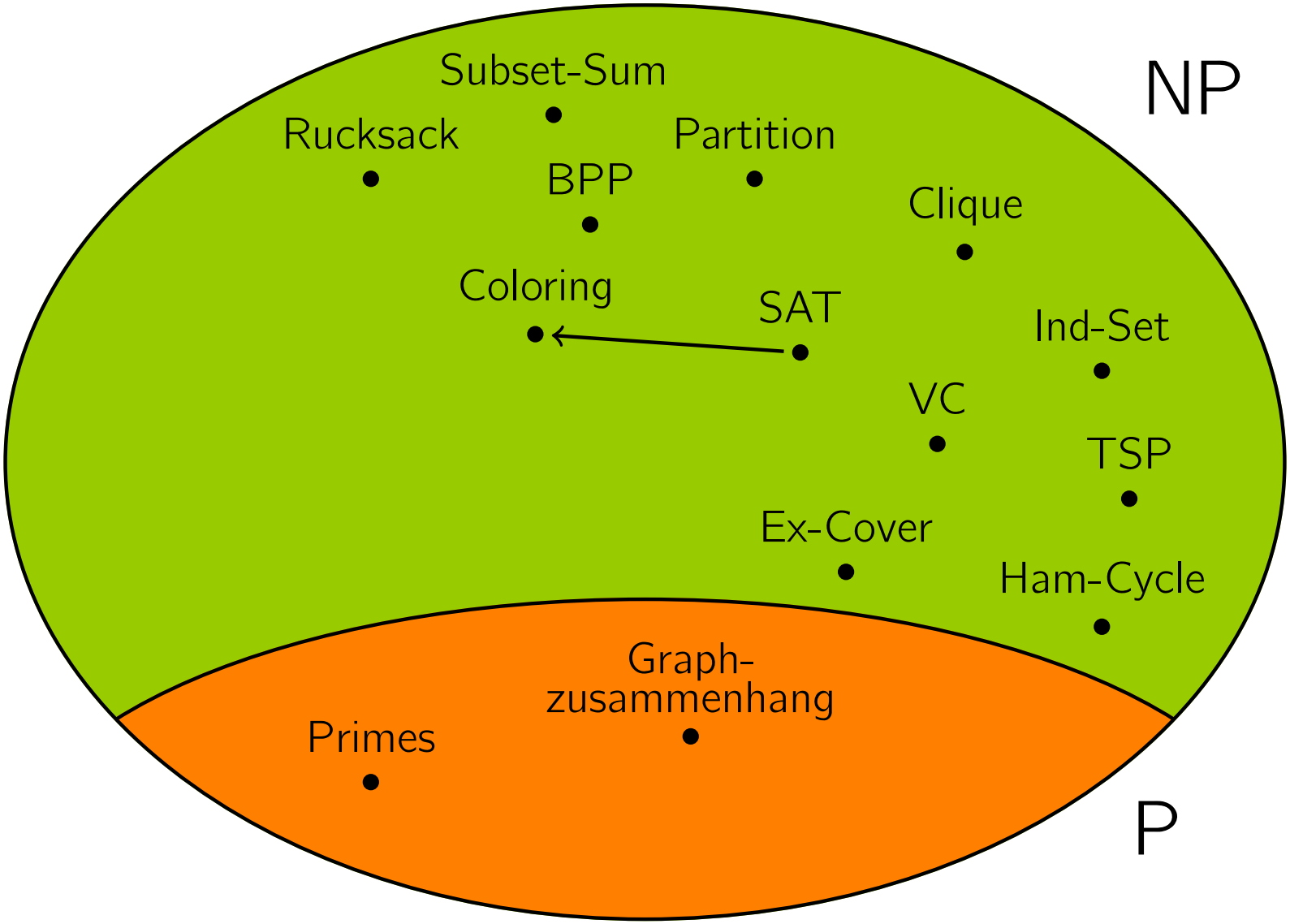
Zeigen Sie: Ex-Cover $\leq_p$ SAT

# Vorlesung VL-14
# Der Satz von Cook & Levin

- NP-Vollständigkeit
- Satz von Cook & Levin

- Kochrezept für NP-Vollständigkeitsbeweise
- NP-Vollständigkeit von 3-SAT
- Karp's Liste von Problemen

# Die Komplexitätslandschaft




Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

# NP-Vollständigkeit

# NP-schwere Probleme

**Definition: NP-schwer**

Ein Problem $L$ heisst NP-schwer (engl.: NP-hard), falls gilt:

$$\forall L' \in NP : \; L' \leq_p L$$

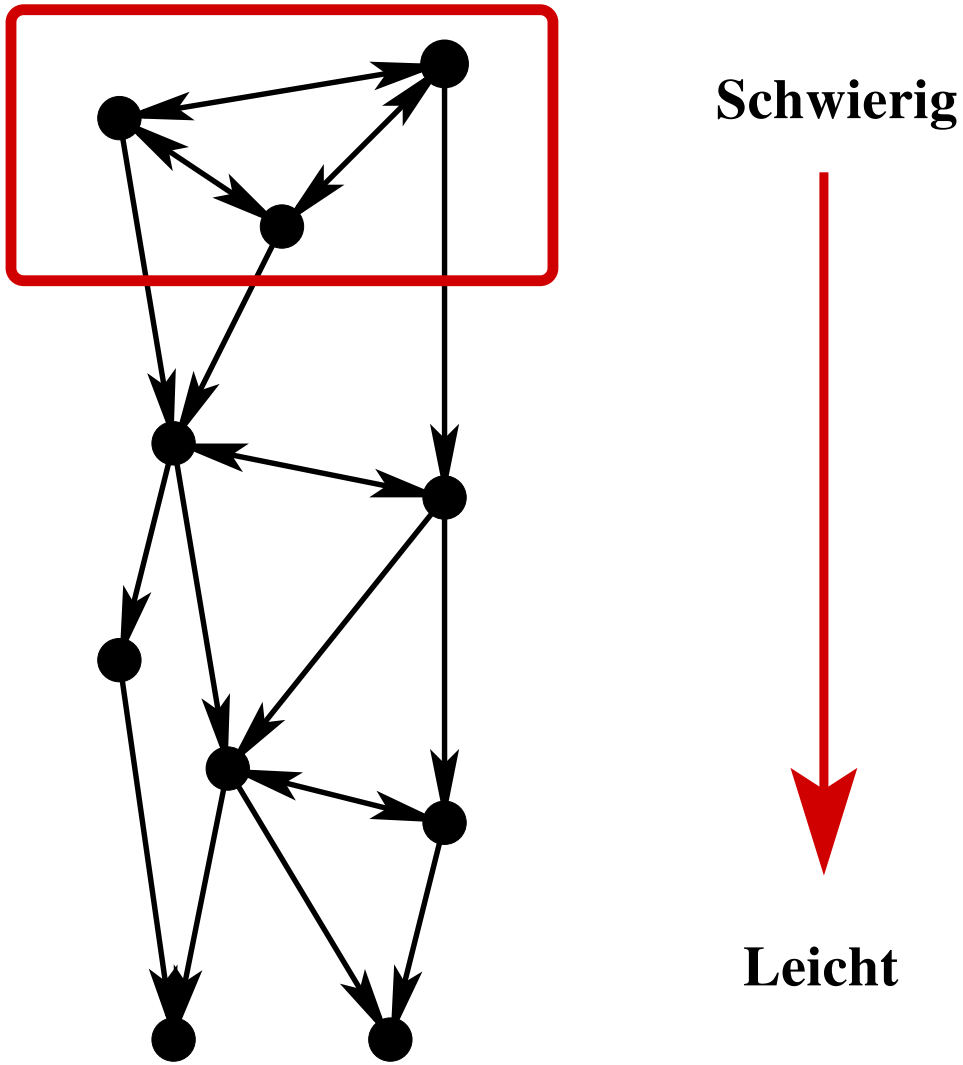
**Satz**

Wenn $L$ NP-schwer ist, dann gilt: $L \in P \Rightarrow P = NP$

Beweis: Ein polynomieller Algorithmus für $L$ liefert zusammen mit der Reduktion $L' \leq_p L$ einen polynomiellen Algorithmus für alle $L' \in NP$.

Fazit: Für NP-schwere Probleme gibt es keine polynomiellen Algorithmen, es sei denn P=NP.

# Illustration



# NP-Vollständige Probleme

**Definition: NP-vollständig**

Ein Problem $L$ heisst NP-vollständig (engl.: NP-complete), falls gilt:

- $L \in \text{NP}$, und
- $L$ ist NP-schwer.

Die Klasse der NP-vollständigen Probleme wird mit NPC bezeichnet.

Wir werden zeigen, dass SAT, CLIQUE, Ham-Cycle, PARTITION, Rucksack und viele weitere Probleme NP-vollständig sind.

Unter der Annahme $P \neq NP$ (Standardannahme) besitzt also keines dieser Probleme einen polynomiellen Algorithmus.

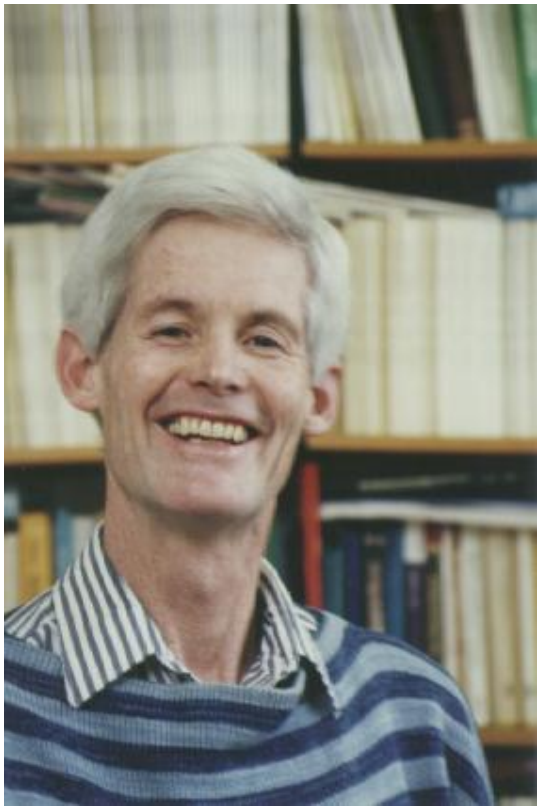
# Satz von Cook & Levin

# Stephen Arthur Cook OC (1939)

Wikipedia: Steve Cook is an American-Canadian computer scientist and mathematician who has made major contributions to the fields of complexity theory and proof complexity.

His seminal paper *"The complexity of theorem proving procedures"* (presented at the 1971 Symposium on the Theory of Computing) laid the foundations for the theory of NP-Completeness.

The ensuing exploration of the boundaries and nature of the class of NP-complete problems has become one of the most active and important research areas in computer science.



# Leonid Anatolievich Levin (1948)

Wikipedia: Leonid Levin is a Soviet-American computer scientist. He obtained his master's degree at Moscow University in 1970 where he studied under Andrej Kolmogorov.

Leonid Levin and Stephen Cook independently discovered the existence of NP-complete problems. Levin is known for his work in randomness in computing, average-case complexity, algorithmic probability, theory of computation, and information theory.

# Der Satz von Cook & Levin

Der Ausgangspunkt für alle unsere NP-Vollständigkeitsbeweise ist das Erfüllbarkeitsproblem SAT.

**Problem: Satisfiability (SAT)**

Eingabe: Boole'sche Formel $\varphi$ in CNF über der Variablenmenge $X$

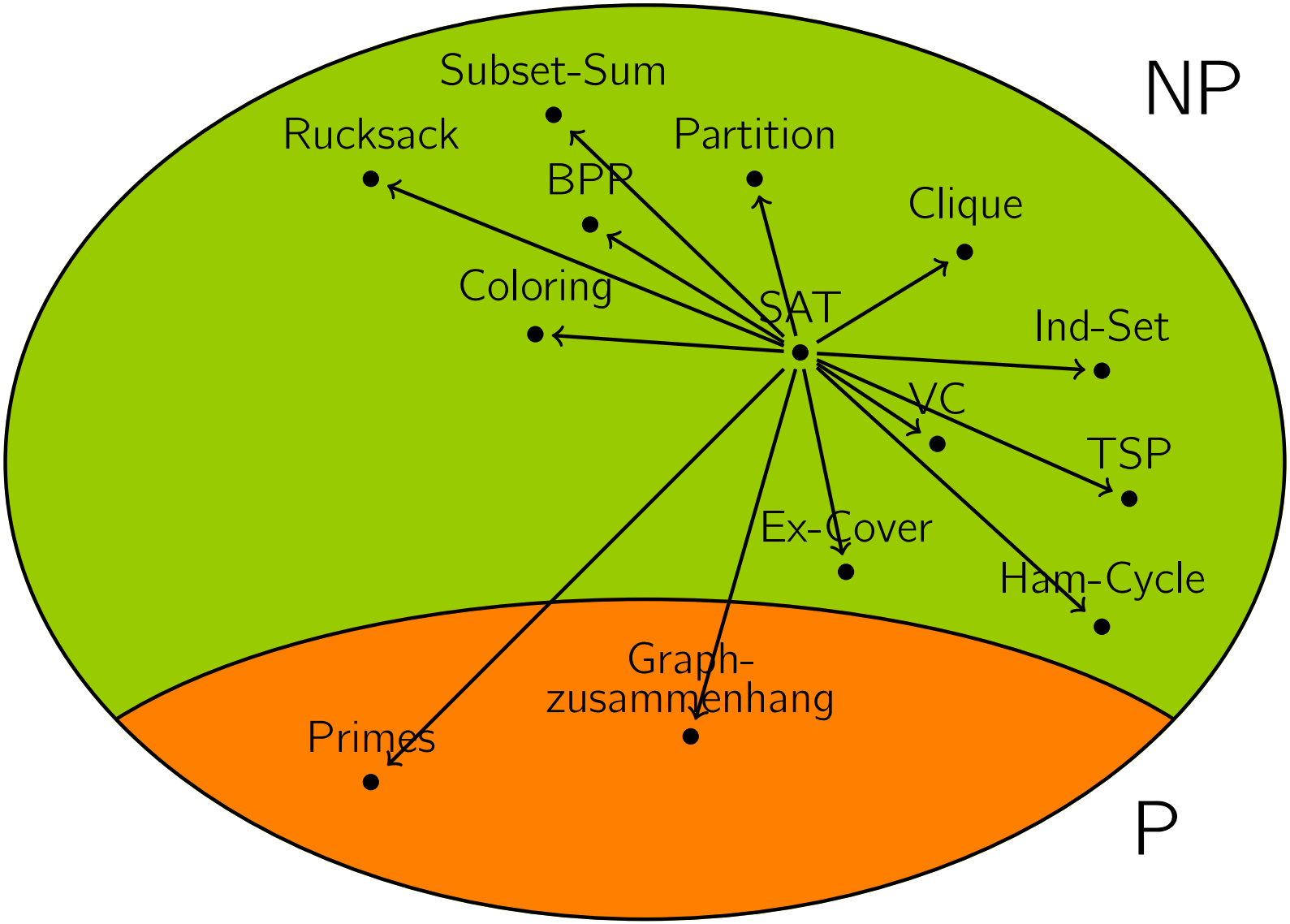
Frage: Existiert eine Wahrheitsbelegung von $X$, die $\varphi$ erfüllt?

**Satz (Cook & Levin)**

SAT ist NP-vollständig.

Fazit: Wenn $P \neq NP$, besitzt SAT keinen polynomiellen Algorithmus.

# Die Komplexitätslandschaft




Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

# Beweis des Satzes von Cook & Levin

# Cook & Levin (1a): Grundideen des Beweises

- Es sei $L \subseteq \Sigma^*$ ein beliebiges Problem in $NP$.
  Es sei $M$ eine NTM, die $L$ in polynomieller Zeit erkennt.
- Wir müssen/werden zeigen, dass $L \leq_p \text{SAT}$ gilt.
- Dazu konstruieren wir eine polynomiell berechenbare Funktion $f$, die jedes $x \in \Sigma^*$ auf eine Formel $\varphi =: f(x)$ abbildet, sodass gilt:

$$x \in L \quad \Leftrightarrow \quad M \text{ akzeptiert } x \quad \Leftrightarrow \quad \varphi \in \text{SAT}$$

**Wir nehmen folgende Eigenschaften der NTM $M$ an:**

- $M$ besucht keine Bandzelle links von der Startzelle.
- Eine akzeptierende Rechnung von $M$ geht in den Zustand $q_{accept}$ über, und bleibt dann dort in einer Endlosschleife.
- Es gibt ein Polynom $p(\cdot)$, sodass $M$ eine Eingabe $x$ genau dann akzeptiert, wenn es einen Rechenweg gibt, der nach $p(|x|)$ Schritten im Zustand $q_{accept}$ gelandet ist.

# Cook & Levin (1b): Grundidee des Beweises

## Beobachtung

Es sei $K_0 = q_0 x$ die Startkonfiguration von $M$. Die NTM $M$ akzeptiert ein Wort $x$ mit $|x| = n$ genau dann, wenn es eine Konfigurationsfolge

$$K_0 \vdash K_1 \vdash K_2 \vdash \cdots \vdash K_{p(n)}$$

gibt, bei der $K_{p(n)}$ im Zustand $q_{\text{accept}}$ ist.

Unsere Formel $\varphi$ wird derart konstruiert,
dass $\varphi$ genau dann erfüllbar ist,
wenn solch eine akzeptierende Konfigurationsfolge existiert.

# Cook & Levin (2a): Die Boole'schen Variablen

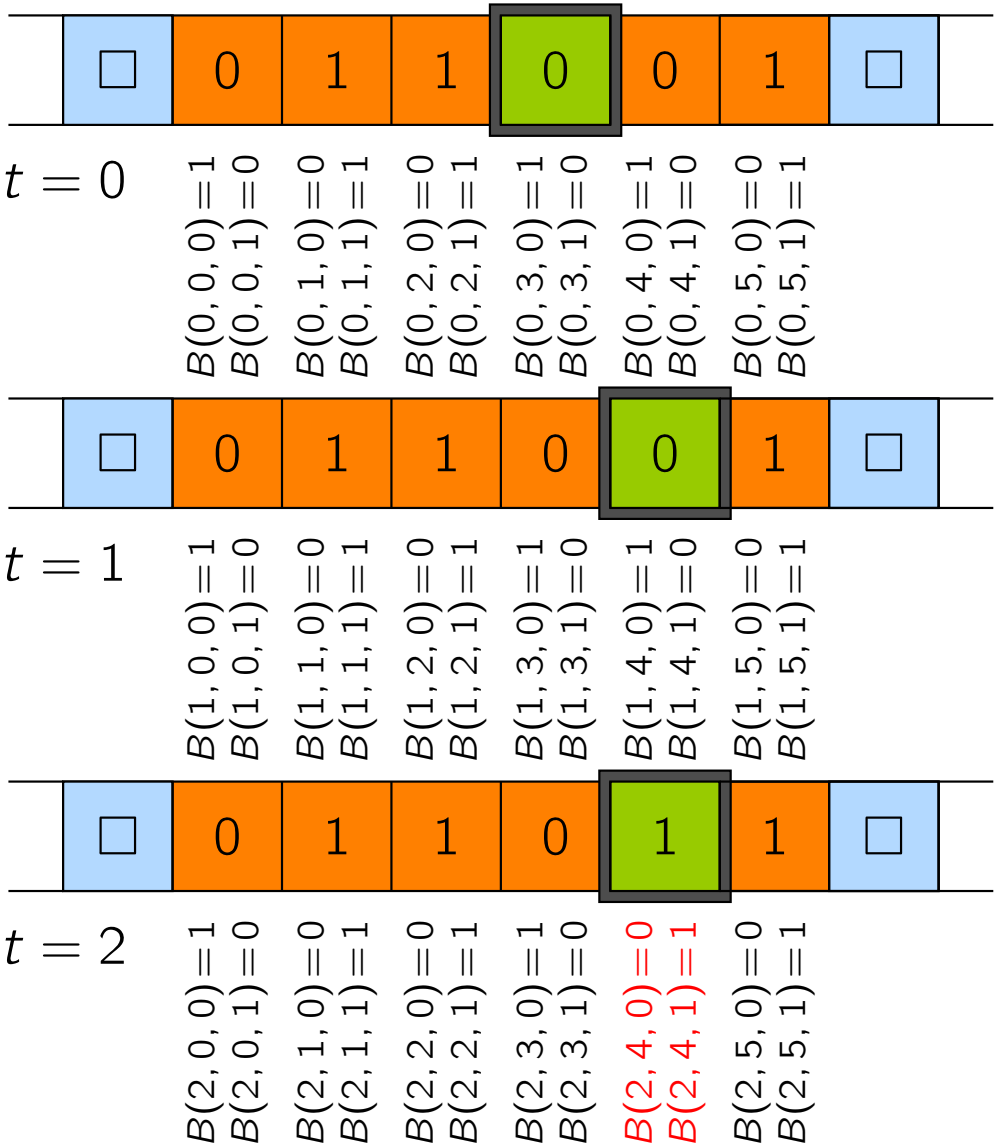
## Die Variablen in der Formel $\varphi$

- $Q(t, q)$ für $t \in \{0, \ldots, p(n)\}$ und $q \in Q$
- $H(t, j)$ für $t, j \in \{0, \ldots, p(n)\}$
- $B(t, j, a)$ für $t, j \in \{0, \ldots, p(n)\}$ und $a \in \Gamma$

Interpretation der Variablen:

- Die Belegung $Q(t, q) = 1$ besagt, dass sich die Berechnung zum Zeitpunkt $t$ im Zustand $q$ befindet.
- Die Belegung $H(t, j) = 1$ steht dafür, dass sich der Kopf zum Zeitpunkt $t$ an Bandposition $j$ befindet.
- Die Belegung $B(t, j, a) = 1$ bedeutet, dass zum Zeitpunkt $t$ an Bandposition $j$ das Zeichen $a$ geschrieben steht.

# Cook & Levin (2b): Illustration der Variablen




$q$

$Q(0, q) = 1$

$H(0,0) = 0$
$H(0,1) = 0$
$H(0,2) = 0$
$H(0,3) = 1$
$H(0,4) = 0$
$H(0,5) = 0$

$q'$

$Q(1, q') = 1$

$H(1,0) = 0$
$H(1,1) = 0$
$H(1,2) = 0$
$H(1,3) = 0$
$H(1,4) = 1$
$H(1,5) = 0$

$q''$

$Q(2, q'') = 1$

$H(2,0) = 0$
$H(2,1) = 0$
$H(2,2) = 0$
$H(2,3) = 0$
$H(2,4) = 1$
$H(2,5) = 0$

# Cook & Levin (3): Unser Arbeitsplan

Wir werden die akzeptierende Konfigurationsfolge in drei Phasen in die Formel $\varphi$ übersetzen.

Arbeitsphase A: Für jeden Zeitpunkt $t$ beschreiben die Variablen $Q(t, q)$, $H(t, j)$ und $B(t, j, a)$ eine legale Konfiguration.

Arbeitsphase B: Die Konfiguration zum Zeitpunkt $t + 1$ entsteht legal aus der Konfiguration zum Zeitpunkt $t$.

Arbeitsphase C: Startkonfiguration und Endkonfiguration sind legal.

# Cook & Levin (4a): Arbeitsphase A

Für jeden Zeitpunkt $t$ konstruieren wir eine Teilformel $\varphi_t$ von Formel $\varphi$, die nur dann erfüllt ist, wenn die Variablen $Q(t,q)$, $H(t,j)$ und $B(t,j,a)$ eine legale Konfiguration $K_t$ beschreiben.

A1. Es gibt genau einen Zustand $q \in Q$ mit $Q(t,q) = 1$.

A2. Es gibt genau eine Bandposition $j \in \{0,\ldots,p(n)\}$ mit $H(t,j) = 1$.

A3. Es gibt für jedes $j \in \{0,\ldots,p(n)\}$ jeweils genau ein Zeichen $a \in \Gamma$ mit $B(t,j,a) = 1$.

# Cook & Levin (4b): Arbeitsphase A

## Boole'sches Werkzeug

Für eine beliebige Variablenmenge $\{y_1, \ldots, y_k\}$ besagt die folgende Formel in CNF, dass genau eine der Variablen $y_i$ den Wert 1 annimmt:

$$(y_1 \vee y_2 \vee \ \ldots \ \vee y_k) \ \ \wedge \ \ \bigwedge_{i \neq j} (\bar{y}_i \vee \bar{y}_j)$$

Die Anzahl der Literale in dieser Formel ist $O(k^2)$ und quadratisch in der Anzahl der Variablen.

Die drei erwünschten Eigenschaften A1/A2/A3 zum Zeitpunkt $t$ (für legale Konfigurationen) können daher jeweils durch eine Formel $\varphi_t$ mit polynomiell beschränkter Länge kodiert werden.

Phase A ist damit abgeschlossen.

# Cook & Levin (5a): Arbeitsphase B

Wir konstruieren für jeden Zeitpunkt $t$ eine Teilformel $\varphi'_t$ von Formel $\varphi$, die erzwingt, dass Konfiguration $K_t$ eine direkte Nachfolgekonfiguration von Konfiguration $K_{t-1}$ ist.

B1. Der Bandinhalt der Konfiguration $K_t$ stimmt an allen Positionen mit dem Bandinhalt der Konfiguration $K_{t-1}$ überein, mit möglicher Ausnahme jener Position, an der der Kopf zum Zeitpunkt $t-1$ ist.

B2. Zustand, Kopfposition und Bandinhalt an Kopfposition verändern sich im Einklang mit der Übergangsrelation $\delta$.

# Cook & Levin (5b): Arbeitsphase B

Eigenschaft B1 (Bandinhalt von $K_t$ stimmt mit Bandinhalt von $K_{t-1}$ überein, ausgenommen Kopfposition) wird wie folgt kodiert:

$$\bigwedge_{i=0}^{p(n)} \bigwedge_{a \in \Gamma} B(t-1,i,a) \wedge \neg H(t-1,i) \;\Rightarrow\; B(t,i,a)$$

## Boole'sches Werkzeug

- $x_1 \Rightarrow x_2$ äquivalent zu $\neg x_1 \vee x_2$

  $\neg(x_1 \wedge x_2)$ äquivalent zu $\neg x_1 \vee \neg x_2$ (De Morgan)

- $y_1 \wedge \neg y_2 \Rightarrow y_3$ äquivalent zu $\neg(y_1 \wedge \neg y_2) \vee y_3$

  äquivalent zu $\neg y_1 \vee y_2 \vee y_3$

$$\bigwedge_{i=0}^{p(n)} \bigwedge_{a \in \Gamma} (\neg B(t-1,i,a) \;\vee\; H(t-1,i) \;\vee\; B(t,i,a))$$

# Cook & Levin (5c): Arbeitsphase B

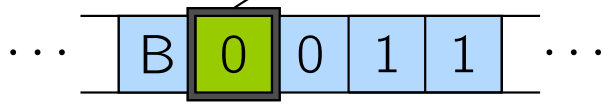
Eigenschaft B2 (Zustand, Kopfposition und Bandinhalt an Kopfposition verändern sich gemäss Übergangsrelation $\delta$) wird wie folgt kodiert.

Für alle $q \in Q$, für alle $j \in \{0, \ldots, p(n) - 1\}$ und für alle $a \in \Gamma$ verwenden wir die Teilformel

$$Q(t-1,q) \wedge H(t-1,j) \wedge B(t-1,j,a) \quad \Rightarrow \bigvee_{(q,a,q',a',\kappa)\in\delta} (Q(t,q') \wedge H(t,j+\kappa) \wedge B(t,j,a'))$$

wobei $\kappa$ die Werte $L = -1$, $N = 0$ und $R = 1$ annehmen kann.

# Cook & Levin (5d): Arbeitsphase B




$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, B\}$
$Q = \{q_1, q_2, q_3\}$

Start $q_1$ Ende $q_2$ Blank $B$

| $\delta$ | 0 | 1 | $B$ |
|---|---|---|---|
| $q_1$ | $\{(q_1, B, R), (q_2, 1, L)\}$ | $\{(q_3, B, R)\}$ | $\{(q_2, B, N)\}$ |
| $q_3$ | $\{(q_1, 1, R)\}$ | $\{(q_2, 0, R)\}$ | $\{(q_1, B, L)\}$ |

$$Q(t-1, q_1) \wedge H(t-1, j) \wedge B(t-1, j, 0) \quad \Rightarrow$$
$$\big(Q(t, q_1) \wedge H(t, j+1) \wedge B(t, j, B)\big)$$
$$\vee \quad \big(Q(t, q_2) \wedge H(t, j-1) \wedge B(t, j, 1)\big)$$

# Cook & Levin (5e): Arbeitsphase B

Für alle $q \in Q$, für alle $j \in \{0, \ldots, p(n)-1\}$ und für alle $a \in \Gamma$ verwenden wir die Teilformel

$$Q(t-1,q) \wedge H(t-1,j) \wedge B(t-1,j,a) \quad \Rightarrow \bigvee_{(q,a,q',a',\kappa)\in\delta} (Q(t,q') \wedge H(t,j+\kappa) \wedge B(t,j,a'))$$

wobei $\kappa$ die Werte $L=-1$, $N=0$ und $R=1$ annehmen kann.

- Die Formel in Rot ist nicht in CNF
- Die Formel in Rot besteht aus höchstens $3\Delta+3$ Literalen (wobei $\Delta$ der maximale Verzweigungsgrad der NTM $M$ ist)
- Die Formel in Rot kann in eine äquivalente Formel in CNF mit höchstens $\leq 3^{3\Delta+3}(3\Delta+3)$ Literalen umgeformt werden

Phase B ist damit abgeschlossen.

# Cook & Levin (6): Arbeitsphase C

Zum Schluss sorgen wir noch dafür, dass Startkonfiguration und Endkonfiuration korrekt beschrieben werden.

C1. Startkonfiguration:

$$Q(0, q_0) \ \wedge \ H(0,0) \ \wedge \ \bigwedge_{i=0}^{n-1} B(0, i, x_i) \ \wedge \ \bigwedge_{i=n}^{p(n)} B(0, i, B)$$

C2. Endkonfiguration:

$$Q(p(n), q_{\text{accept}})$$

Phase C ist damit abgeschlossen.

# Cook & Levin: Zusammenfassung

- Die Gesamtformel $\varphi$ setzt sich aus allen Teilformeln zusammen, die wir für A1/A2/A3 und B1/B2 und C1/C2 konstruiert haben.
- Insgesamt sind das polynomiell viele Klauseln, die jeweils aus polynomiell vielen Literalen bestehen.
- Die Länge von $\varphi$ ist daher polynomiell beschränkt in $n$, und $\varphi$ kann aus $x$ in polynomieller Zeit berechnet werden.
- Die Formel $\varphi$ genau dann erfüllbar, wenn es eine akzeptierende Konfigurationsfolge der Länge $p(n)$ für $M$ auf $x$ gibt.

**Satz (Cook & Levin)**

SAT ist NP-vollständig. □

# Kochrezept für NP-Vollständigkeitsbeweise

# Kochrezept für NP-Vollständigkeitsbeweise (1)

- Die NP-Vollständigkeit von SAT haben wir durch eine lange "Master-Reduktion" von allen Problemen aus NP auf SAT bewiesen
- Um die NP-Vollständigkeit von anderen Problemen zu zeigen, könnten wir natürlich für jedes neue Problem eine ähnlich mühsame und ähnlich langwierige Master-Reduktion erstellen
- Ein viel einfacherer Ansatz weist die NP-Vollständigkeit von neuen Problemen mit Hilfe der NP-Vollständigkeit von SAT nach

**Satz**

Wenn $L^*$ NP-schwer ist, dann gilt: $L^* \leq_p L \Rightarrow L$ ist NP-schwer

Beweis:
Für alle $L' \in NP$ gilt $L' \leq_p L^*$ und $L^* \leq_p L$.
Die Transitivität von $\leq_p$ impliziert $L' \leq_p L$ für alle $L' \in NP$.

# Kochrezept für NP-Vollständigkeitsbeweise (2)

## Hier ist das Kochrezept:

1. Man zeige $L \in NP$.

2. Man wähle eine NP-vollständige Sprache $L^*$.
3. **(Reduktionsabbildung):** Man konstruiere eine Funktion $f$, die Instanzen von $L^*$ auf Instanzen von $L$ abbildet.
4. **(Polynomielle Zeit):** Man zeige, dass $f$ in polynomieller Zeit berechnet werden kann.
5. **(Korrektheit):** Man beweise, dass $f$ tatsächlich eine Reduktion ist. Für $x \in \{0,1\}^*$ gilt $x \in L^*$ genau dann, wenn $f(x) \in L$.

# NP-Vollständigkeit von 3-SAT

# 3-SAT: Definition

- Eine $k$-Klausel ist eine Klausel, die aus exakt $k$ Literalen besteht
- Eine CNF-Formel $\varphi$ ist in $k$-CNF, wenn sie aus $k$-Klauseln besteht

**Beispiel einer Formel in 3-CNF**

$$\varphi = \underbrace{(\bar{x}_1 \vee \bar{x}_2 \vee x_3)}_{\text{3 Literale}} \wedge \underbrace{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)}_{\text{3 Literale}}$$

**Problem: 3-SAT**

Eingabe: Eine Boole'sche Formel $\varphi$ in 3-CNF
Frage: Besitzt $\varphi$ eine erfüllende Belegung?

3-SAT ist ein Spezialfall von SAT und liegt deshalb wie SAT in NP

# 3-SAT: NP-Vollständigkeit (Beginn)

**Satz**

SAT $\leq_p$ 3-SAT

Beweis:

- Gegeben sei eine beliebige Formel $\varphi$ in CNF (Instanz von SAT)
- Wir werden eine zur Formel $\varphi$ äquivalente Formel $\varphi'$ in 3-CNF konstruieren: $\varphi$ ist erfüllbar $\Leftrightarrow$ $\varphi'$ ist erfüllbar

- Aus einer 1-Klausel oder 2-Klausel machen wir eine äquivalente 3-Klausel, indem wir ein oder zwei Literale duplizieren
- 3-Klauseln bleiben 3-Klauseln
- Auf $k$-Klauseln mit $k \geq 4$ wenden wir wiederholt die folgende Klauseltransformation an:
  Die Klausel $c = (\ell_1 + \ell_2 + \ell_3 + \cdots + \ell_k)$ wird ersetzt durch die beiden neuen Klauseln $(\ell_1 + \cdots + \ell_{k-2} + h)$ und $(\bar{h} + \ell_{k-1} + \ell_k)$.
  Hier bezeichnet $h$ eine neu eingeführte Hilfsvariable.

# Klauseltransformation: Beispiel

## Klauseltransformation für eine 5-Klausel

- Wir beginnen mit der 5-Klausel $(x_1 + \bar{x}_2 + x_3 + x_4 + \bar{x}_5)$
- Im ersten Transformationsschritt wird daraus eine 4-Klausel und eine 3-Klausel gemacht: $(x_1 + \bar{x}_2 + x_3 + h_1)\ (\bar{h}_1 + x_4 + \bar{x}_5)$
- Auf die 4-Klausel wird die Transformation dann erneut angewendet. Dadurch entsteht nun $(x_1 + \bar{x}_2 + h_2)\ (\bar{h}_2 + x_3 + h_1)\ (\bar{h}_1 + x_4 + \bar{x}_5)$, und es sind nur noch 3-Klauseln vorhanden.

# Klauseltransformation: Korrektheit

Alte Klausel: $c = (\ell_1 + \ell_2 + \ell_3 + \cdots + \ell_k)$
Neue Klauseln: $c' = (\ell_1 + \cdots + \ell_{k-2} + h)$ und $c'' = (\bar{h} + \ell_{k-1} + \ell_k)$

(1) Wenn eine Wahrheitsbelegung $c'$ und $c''$ erfüllt, so erfüllt sie automatisch auch $c$:

- Wenn $h = 0$, dann ist $\ell_1 + \cdots + \ell_{k-2}$ wahr
- Wenn $h = 1$, dann ist $\ell_{k-1} + \ell_k$ wahr

(2) Wenn eine Wahrheitsbelegung $c$ erfüllt, so kann sie auf $h$ erweitert werden, sodass die beiden Klauseln $c'$ und $c''$ erfüllt sind:

- Die Wahrheitsbelegung macht mindestens ein Literal aus $c$ wahr
- Falls $\ell_1 + \cdots + \ell_{k-2}$ wahr ist, setzen wir $h = 0$
- Falls $\ell_{k-1} + \ell_k$ wahr ist, so setzen wir $h = 1$

# 3-SAT: NP-Vollständigkeit (Ende)

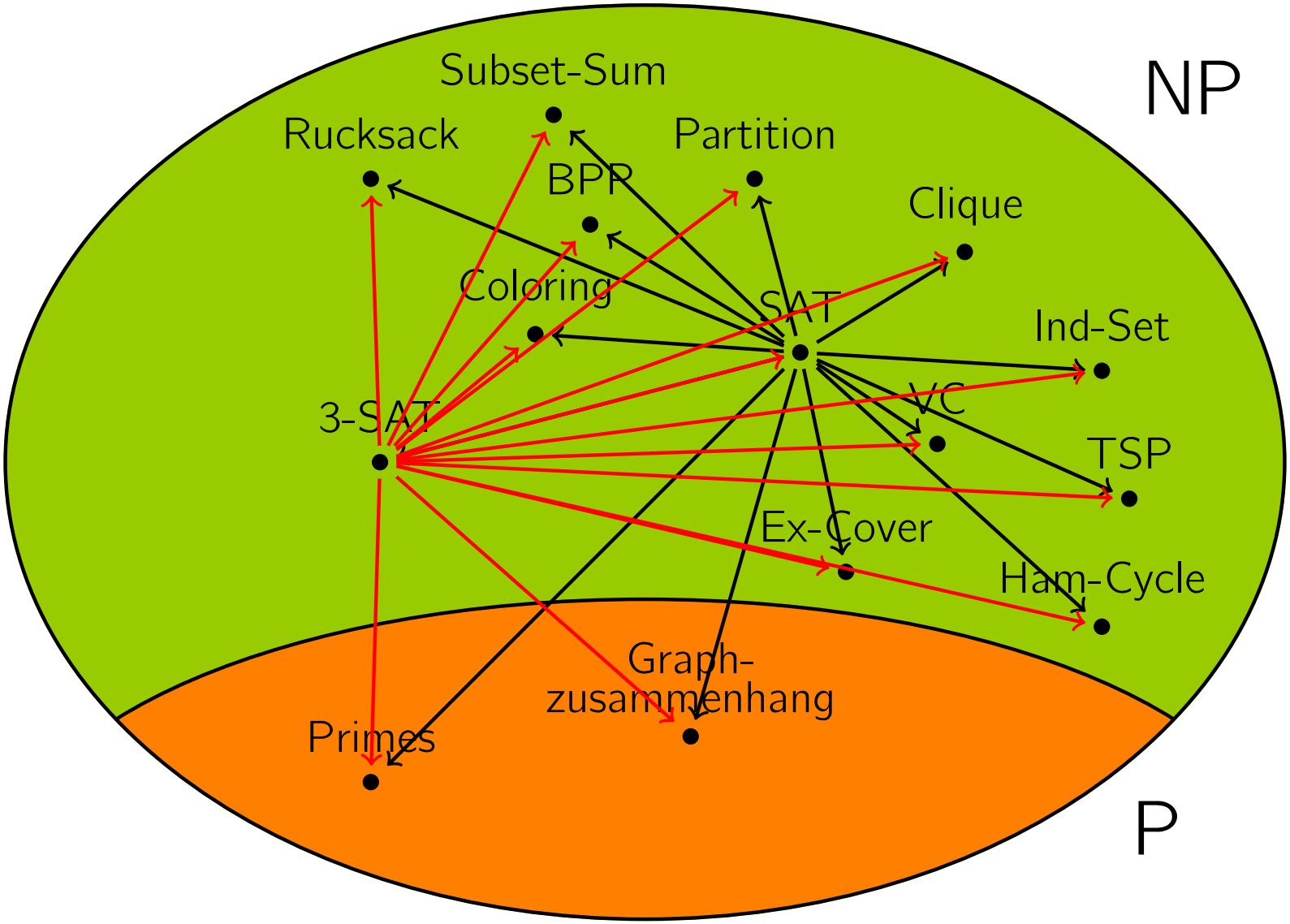
- Durch Anwendung der Klauseltransformation entstehen aus einer $k$-Klausel eine $(k-1)$-Klausel und eine 3-Klausel.
- Nach $k-3$ Iterationen sind dann aus einer einzelnen alten $k$-Klausel genau $k-2$ neue 3-Klauseln entstanden.
- Aus $k \geq 4$ alten Literalen entstehen also $3k-6$ neue Literale.
- Diese Transformation wird solange auf die Formel $\varphi$ angewandt, bis die Formel nur noch 3-Klauseln enthält.

- Wenn $\varphi$ aus $p$ Literalen besteht, so besteht $\varphi'$ aus höchstens $3p$ Literalen.
- Die Laufzeit der Reduktion ist daher polynomiell beschränkt.

**Satz**

3-SAT ist NP-vollständig.

# Die Komplexitätslandschaft




Warnung: Dieser Abbildung liegt die Annahme $P \neq NP$ zu Grunde.

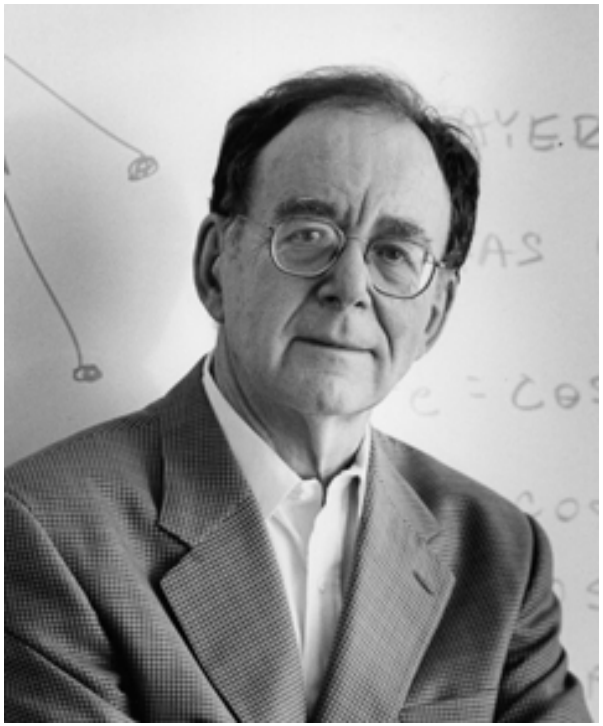
# Karp's Liste mit 21 Problemen

# Richard Manning Karp (1935)

Wikipedia: Richard Karp is an American computer scientist, who has made many important discoveries in computer science, operations research, and in the area of combinatorial algorithms.

Karp introduced the now standard methodology for proving problems to be NP-complete which has led to the identification of many practical problems as being computationally difficult.

- Edmonds-Karp algorithm for max-flow
- Hopcroft-Karp algorithm for matching
- Rabin-Karp string search algorithm
- Karp-Lipton theorem



# Karp's Liste mit 21 NP-vollständigen Problemen

Richard Karp bewies 1972 die NP-Vollständigkeit von 21 kombinatorischen und graphen-theoretischen Problemen, die sich hartnäckig einer effizienten algorithmischen Lösung entzogen hatten.

| | |
|---|---|
| SAT | 3-SAT |
| INTEGER PROGRAMMING | COLORING |
| CLIQUE | CLIQUE COVER |
| INDEP-SET | EXACT COVER |
| VERTEX COVER | 3-DIM MATCHING |
| SET COVER | STEINER TREE |
| FEEDBACK ARC SET | HITTING SET |
| FEEDBACK VERTEX SET | SUBSET-SUM |
| DIR HAM-CYCLE | JOB SEQUENCING |
| UND HAM-CYCLE | PARTITION |
| | MAX-CUT |

# Landkarte mit Karp's 20 Reduktionen