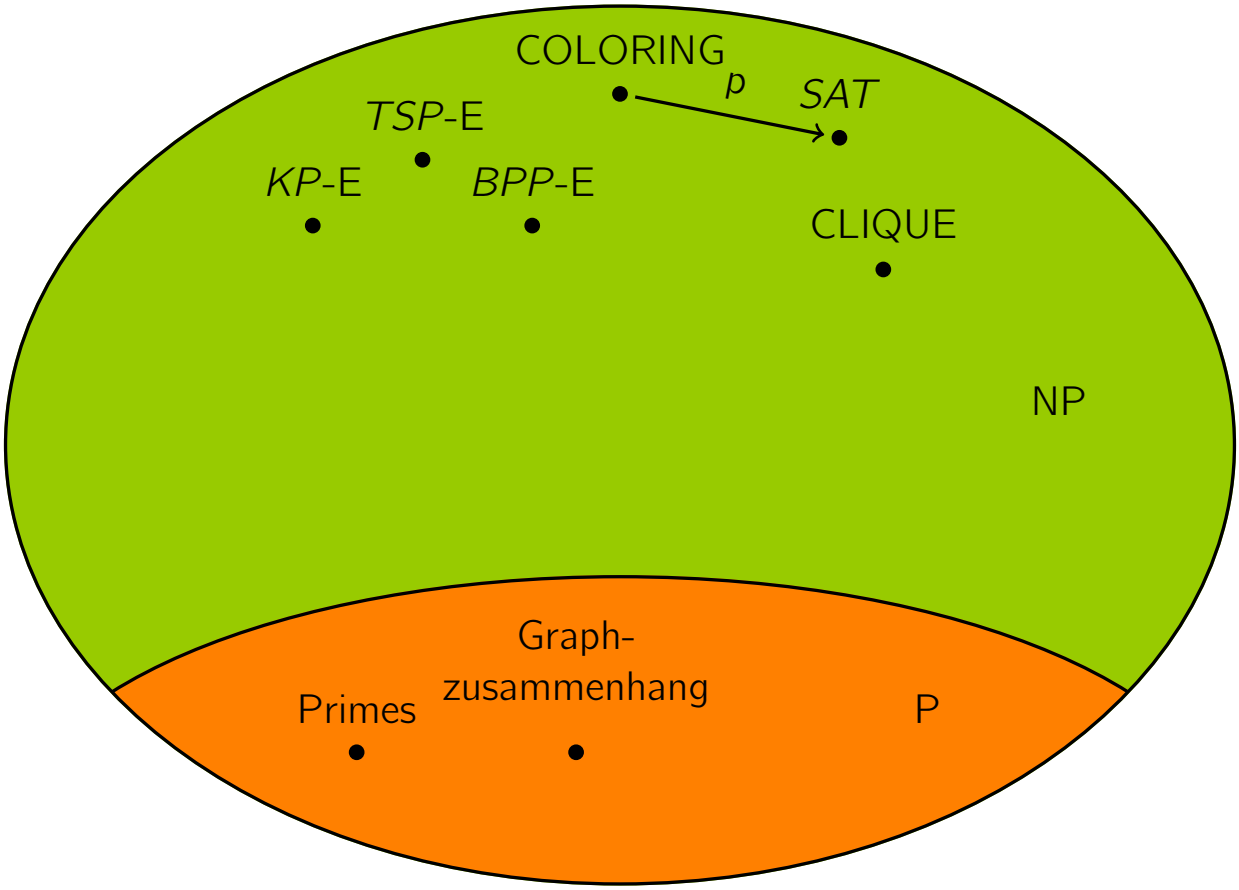
Vorlesung 15
NP-Vollständigkeit

Wdh.: Die Komplexitätslandschaft



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

Wdh.: Optimierungs- versus Entscheidungsproblem

Mit Hilfe eines Algorithmus, der ein Optimierungsproblem löst, kann man die Entscheidungsvariante lösen.

Umgekehrt gilt:

Satz

Wenn die Entscheidungsvariante von KP in polynomieller Zeit lösbar ist, dann auch die Optimierungsvariante.

Dieser Satz gilt auch für TSP und BPP.

Wdh.: Alternative Charakterisierung der Klasse NP

Satz

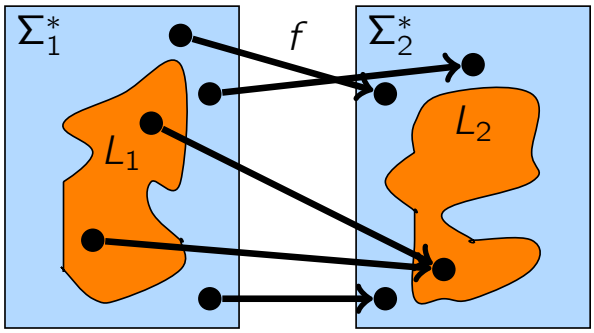
Eine Sprache $L \subseteq \Sigma^*$ ist genau dann in NP*, wenn es einen Polynomialzeitalgorithmus V (einen sogenannten Verifizierer) und ein Polynom p mit der folgenden Eigenschaft gibt:*

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

Wdh.: Polynomielle Reduktionen

Definition (Polynomielle Reduktion)

L_1 und L_2 seien zwei Sprachen über Σ_1 bzw. Σ_2. Dann heißt L_1 polynomiell reduzierbar auf L_2, wenn es eine Reduktion von L_1 nach L_2 gibt, die in polynomieller Zeit berechenbar ist. Wir schreiben $L_1 \leq_p L_2$.



Lemma

Angenommen $L_1 \leq_p L_2$, dann gilt: $L_2 \in P \Rightarrow L_1 \in P$.

Satz

COLORING $\leq_p$ SAT.

Wdh.: Das Erfüllbarkeitsproblem – SAT

Problem (Erfüllbarkeitsproblem / Satisfiability – SAT)

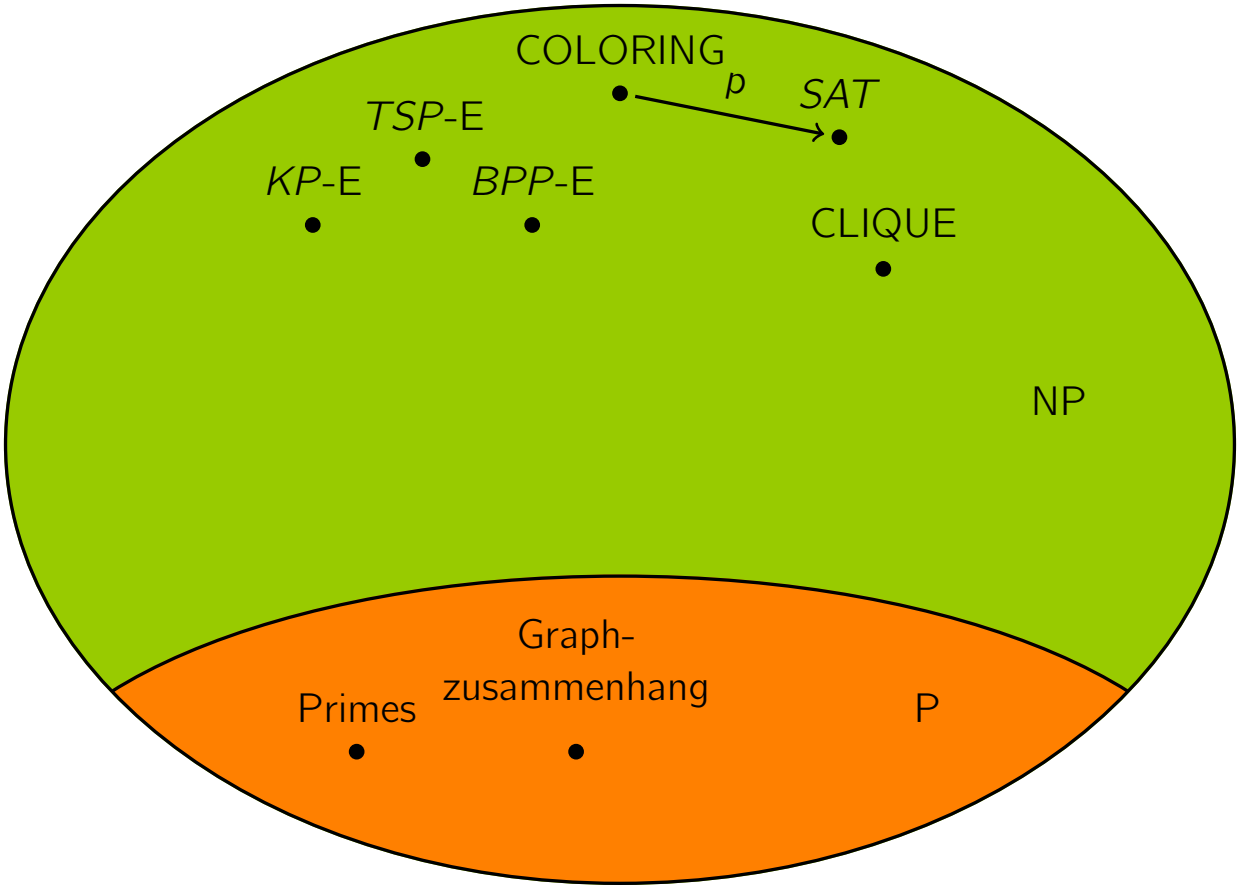
Eingabe: Aussagenlogische Formel φ in KNF
Frage: Gibt es eine erfüllende Belegung für φ?

SAT-Beispiel 1:

$$\varphi = (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee x_3 \vee x_4)$$

φ ist erfüllbar, denn $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ ist eine erfüllende Belegung.

Wdh.: Die Komplexitätslandschaft



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

NP-schwere Probleme

Definition (NP-schwer)

Ein Problem L heißt NP-schwer (engl. NP-hard), wenn gilt:

$$\forall L' \in \mathrm{NP} : L' \leq_p L.$$

Satz

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

Beweis: Ein Polynomialzeitalgorithmus für L liefert mit der Reduktion $L' \leq_p L$ einen Polynomialzeitalgorithmus für alle $L' \in \mathrm{NP}$. □

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

Def: NP-Vollständigkeit

Definition (NP-vollständig)

Ein Problem L heißt NP-vollständig (engl. NP-complete), falls gilt

1. $L \in \mathrm{NP}$, und
2. L ist NP-schwer.

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

Wir werden zeigen, dass SAT, CLIQUE, KP-E, BPP-E, TSP-E und weitere Probleme NP-vollständig sind.

Unter der Annahme, dass $\mathrm{P} \neq \mathrm{NP}$, hat also keines dieser Probleme einen Polynomialzeitalgorithmus.

NP-Vollständigkeit des Erfüllbarkeitsproblems

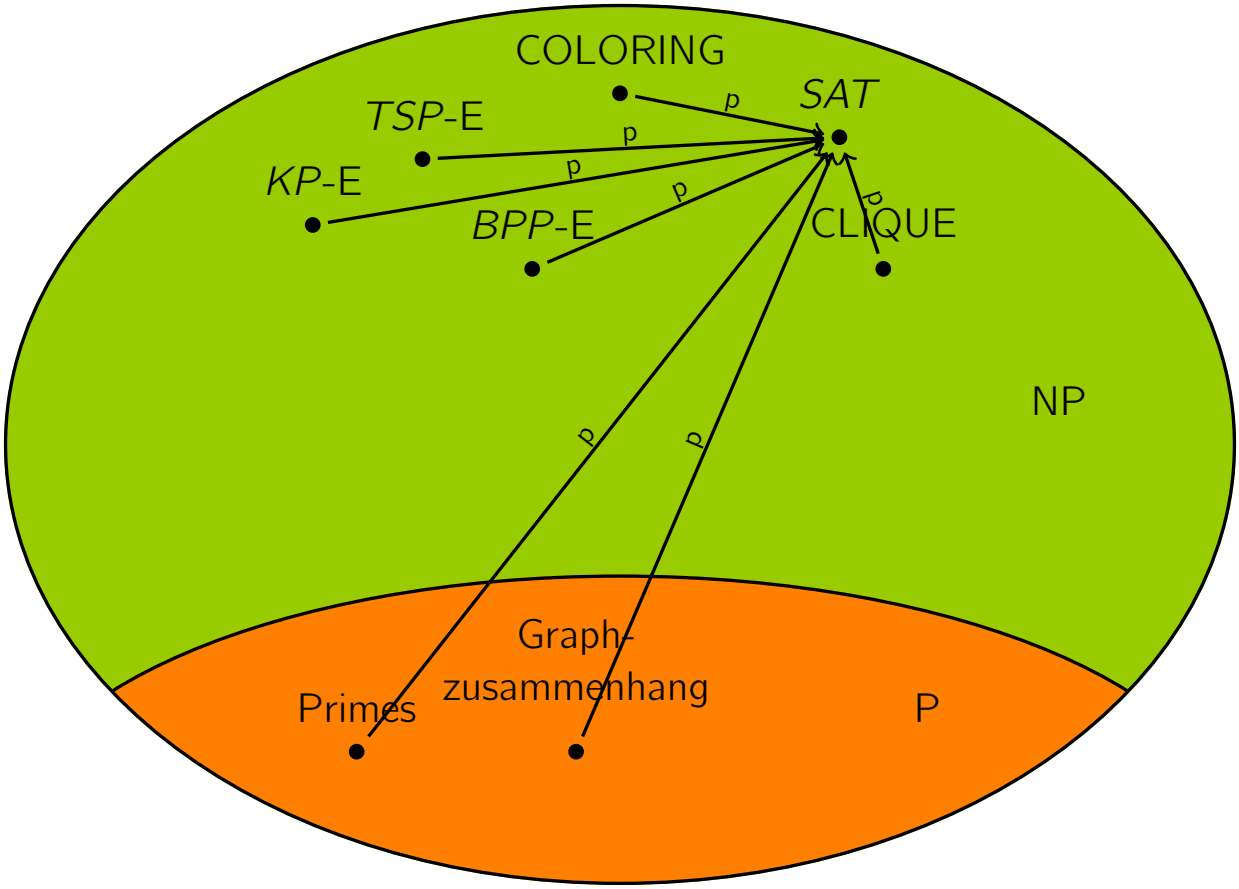
Der Ausgangspunkt für unsere NP-Vollständigkeitsbeweise ist das Erfüllbarkeitsproblem.

Satz (Cook und Levin)

SAT ist NP-vollständig.

Unter der Annahme, dass $\mathrm{P} \neq \mathrm{NP}$, hat SAT also keinen Polynomialzeitalgorithmus.

Die Komplexitätslandschaft



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

Beweis des Satzes von Cook und Levin

- Es gilt SAT $\in$ NP, denn die erfüllende Belegung kann als Zertifikat verwendet werden.
- Wir müssen jetzt noch zeigen, dass SAT NP-schwer ist.

Sei $L \subseteq \Sigma^*$ ein Problem aus NP. Wir müssen zeigen, dass $L \leq_p$ SAT.

Dazu konstruieren wir eine polynomiell berechenbare Funktion f, die jedes $x \in \Sigma^*$ auf eine Formel φ abbildet, so dass gilt

$$x \in L \;\Leftrightarrow\; \varphi \in SAT \;\; .$$

Beweis des Satzes von Cook und Levin

M sei eine NTM, die L in polynomieller Zeit erkennt. Wir werden zeigen

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

Eigenschaften von M

- O.B.d.A. besuche M keine Bandpositionen links von der Startposition.
- Eine akzeptierende Rechnung von M gehe in den Zustand q_{accept} über und bleibe dort in einer Endlosschleife.
- Sei $p(\cdot)$ ein Polynom, so dass M eine Eingabe x genau dann akzeptiert, wenn es einen Rechenweg gibt, der nach $p(n)$ Schritten im Zustand q_{accept} ist, wobei n die Länge von x bezeichne.

Beweis des Satzes von Cook und Levin

Beobachtung:

Sei $K_0 = q_0 x$ die Startkonfiguration von M. M akzeptiert genau dann, wenn es einen Rechenweg, d.h. eine mögliche Konfigurationsfolge

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

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

Weiteres Vorgehen:

Wir konstruieren die Formel φ derart, dass φ genau dann erfüllbar ist, wenn es solch eine akzeptierende Konfigurationsfolge gibt.

Beweis des Satzes von Cook und Levin

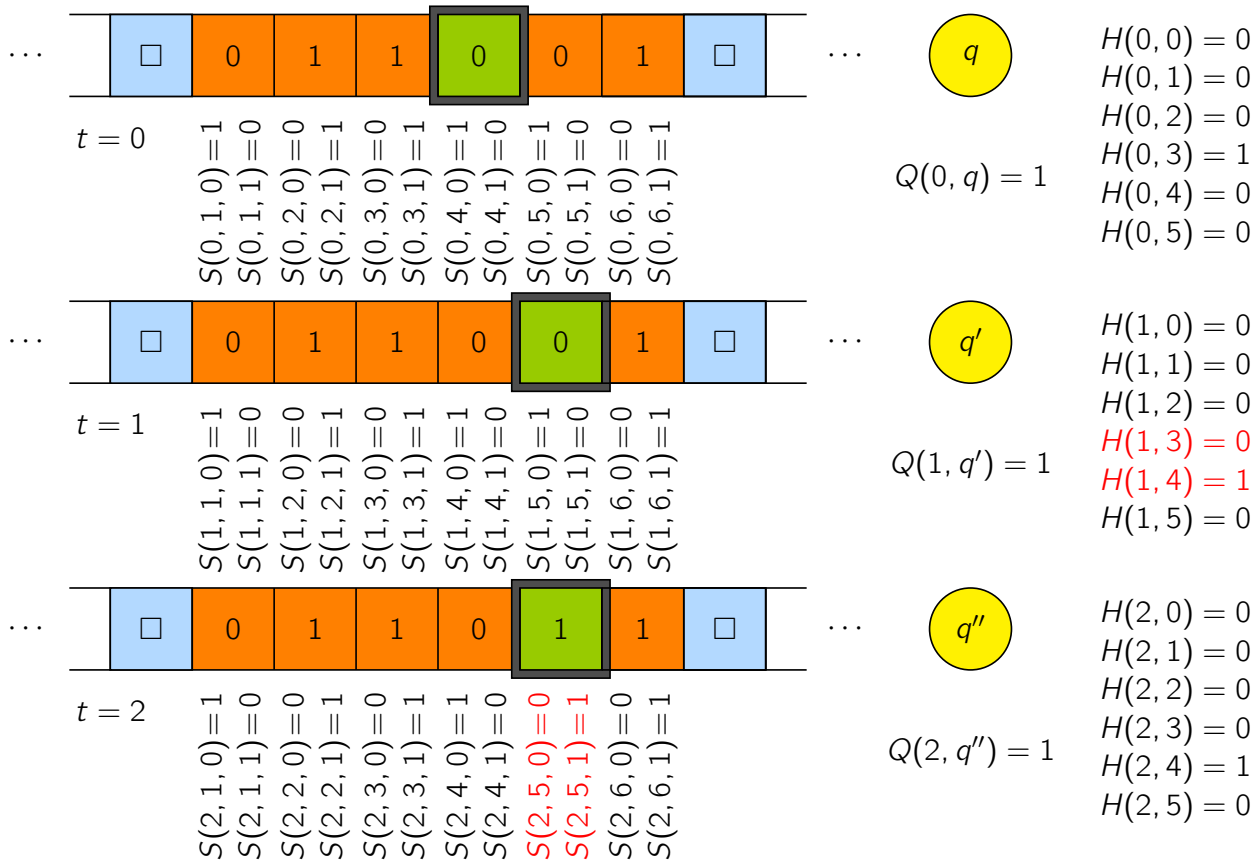
Variablen in φ

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

Interpretation der Variablen:

- Die Belegung $Q(t,k)=1$ soll besagen, dass sich die Rechnung zum Zeitpunkt t im Zustand k befindet.
- Die Belegung $H(t,j)=1$ steht dafür, dass sich der Kopf zum Zeitpunkt t an Bandposition j befindet.
- die Belegung $S(t,j,a)=1$ bedeutet, dass zum Zeitpunkt t an Bandposition j das Zeichen a geschrieben steht.

Beweis des Satzes von Cook und Levin – Illustration



Beweis des Satzes von Cook und Levin

Kodierung einzelner Konfigurationen in der Teilformel φ_t:

Für jedes $t \in \{0, \ldots, p(n)\}$, benötigen wir eine Formel φ_t, die nur dann erfüllt ist, wenn es

1. genau einen Zustand $k \in Q$ mit $Q(t, k) = 1$ gibt,
2. genau eine Bandposition $j \in \{0, \ldots, p(n)\}$ mit $H(t, j) = 1$ gibt, und
3. für jedes $j \in \{0, \ldots, p(n)\}$ jeweils genau ein Zeichen $a \in \Gamma$ mit $S(t, j, a) = 1$ gibt.

Beweis des Satzes von Cook und Levin

Erläuterung zur Formel φ_t:

- Für eine beliebige Variablenmenge $\{y_1, \ldots, y_m\}$ besagt das folgende Prädikat in KNF, dass genau eine der Variablen y_i den Wert 1 annimmt:

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

- Die Anzahl der Literale in dieser Formel ist quadratisch in der Anzahl der Variablen.
- Die drei Anforderungen können also jeweils durch eine Formel in polynomiell beschränkter Länge kodiert werden.

Wir betrachten nun nur noch Belegungen, welche die Teilformeln $\varphi_0, \ldots, \varphi_{p(n)}$ erfüllen und somit Konfigurationen $K_0, \ldots, K_{p(n)}$ beschreiben.

Beweis des Satzes von Cook und Levin

Als Nächstes konstruieren wir eine Formel φ'_t für $1 \leq t \leq p(n)$, die nur für solche Belegungen erfüllt ist, bei denen K_t eine direkte Nachfolgekonfiguration von K_{t-1} ist.

Die Formel φ'_t kodiert zwei Eigenschaften:

1. Die Bandinschrift von K_t stimmt an allen Positionen außer möglicherweise der Position, an der der Kopf zum Zeitpunkt $t-1$ ist, mit der Inschrift von K_{t-1} überein.
2. Zustand, Kopfposition und Bandinschrift an der Kopfposition verändern sich gemäß der Übergangsrelation δ.

Beweis des Satzes von Cook und Levin

Die Eigenschaft, dass die Bandinschrift von K_t an allen Positionen außer möglicherweise der Position, an der der Kopf zum Zeitpunkt $t-1$ ist, mit der Inschrift von K_{t-1} übereinstimmt, kann wie folgt kodiert werden:

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

Dabei steht $A \Rightarrow B$ für $\neg A \vee B$. D.h., die Formel lautet eigentlich

$$\bigwedge_{i=0}^{p(n)} \bigwedge_{z \in \Gamma} (\neg(S(t-1,i,z) \wedge \neg H(t-1,i)) \vee S(t,i,z))$$

Das De Morgansche Gesetz besagt, dass $\neg(A \wedge B)$ äquivalent zu $\neg A \vee \neg B$ ist. Dadurch ergibt sich folgende Teilformel in KNF:

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

Beweis des Satzes von Cook und Levin

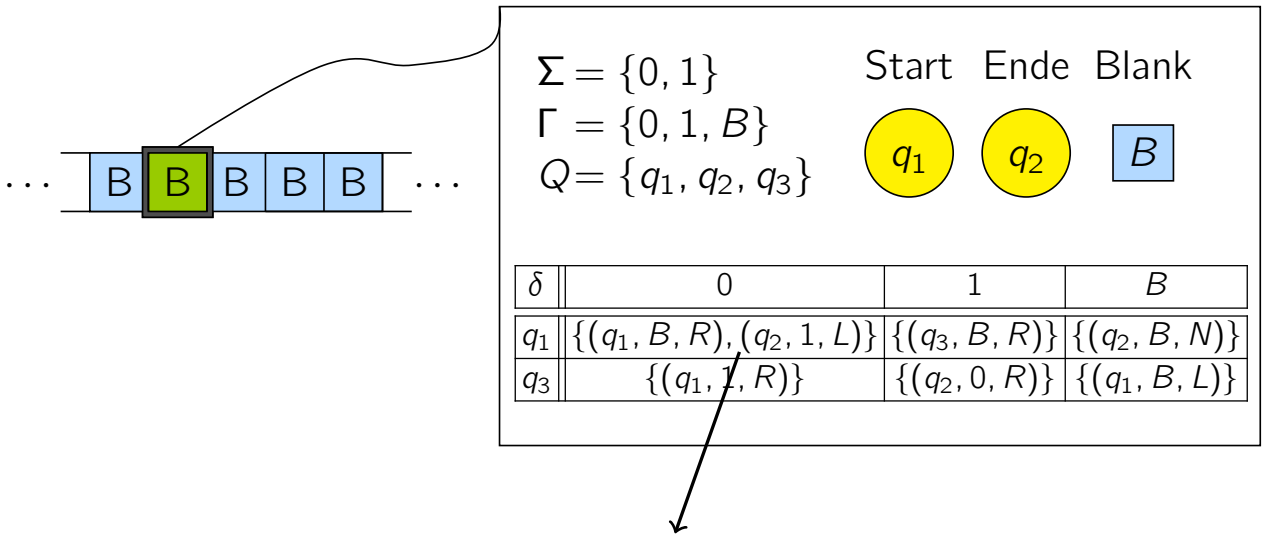
Für die Eigenschaft, dass Zustand, Kopfposition und Bandinschrift an der Kopfposition sich gemäß der Ubergangsrelation δ verändern, ergänzen wir für alle $k \in Q$, $j \in \{0, \ldots, p(n)-1\}$ und $a \in \Gamma$ die folgende Teilformel

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

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

Wie lässt sich diese Teilformel in die KNF transformieren?

Beispielübergangsklausel



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

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

Beweis des Satzes von Cook und Levin

Ersetzen von $A \Rightarrow B$ durch $\neg A \vee B$ und Anwenden des De Morganschen Gesetzes ergibt die Teilformel

$$\neg Q(t-1,k) \vee \neg H(t-1,j) \vee \neg S(t-1,j,a)) \vee \bigvee_{(k,a,k',a',\kappa)\in\delta} (Q(t,k') \wedge H(t,j+\kappa) \wedge S(t,j,a')) \ ,$$

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

Jetzt müssen noch die inneren $\wedge$-Verknüpfungen „ausmultipliziert“ werden, d.h. wir ersetzen wiederholt eine Formel der Form $X \vee (A \wedge B \wedge C)$ durch eine äquivalente Formel $(X \vee A) \wedge (X \vee B) \wedge (X \vee C)$.
Wiederholte Anwendung führt zu einer Formel in KNF.

Damit ist die Beschreibung von φ'_t abgeschlossen.

Beweis des Satzes von Cook und Levin

Die Gesamtformel φ ergibt sich nun wie folgt:

$$Q(0,q_0) \ \wedge \ H(0,0) \ \wedge \ \bigwedge_{i=0}^{n} S(0,i,x_i) \ \wedge \ \bigwedge_{i=n+1}^{p(n)} S(0,i,B)$$
$$\wedge \ \bigwedge_{i=0}^{p(n)} \varphi_i \ \wedge \ \bigwedge_{i=1}^{p(n)} \varphi'_i \ \wedge \ Q(p(n),q_{\text{accept}})$$

Die Länge von φ ist polynomiell beschränkt in n, und φ ist effizient aus x berechenbar.

Gemäß unserer Konstruktion ist φ genau dann erfüllbar, wenn es eine akzeptierende Konfigurationsfolge der Länge $p(n)$ für M auf x gibt. $\square$

Kochrezept für NP-Vollständigkeitsbeweise

- Um nachzuweisen, dass SAT NP-schwer ist, haben wir in einer „Master-Reduktion“ alle Probleme aus NP auf SAT reduziert.
- Die NP-Vollständigkeit von SAT können wir jetzt verwenden, um nachzuweisen, dass weitere Probleme NP-schwer sind.

Lemma

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

Beweis: Gemäß Voraussetzung gilt $\forall L' \in \text{NP}: L' \leq_p L^*$ und $L^* \leq_p L$. Aufgrund der Transitivität der polynomiellen Reduktion folgt somit $\forall L' \in \text{NP}: L' \leq_p L$. □

Kochrezept für NP-Vollständigkeitsbeweise

Wie beweist man, dass eine Sprache L NP-vollständig ist?

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

NP-Vollständigkeit von 3-SAT

Eine Formel in k-KNF besteht nur aus Klauseln mit jeweils k Literalen, sogenannten k-Klauseln.

Beispiel einer Formel in 3-KNF:

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

Problem (3-SAT)

Eingabe: Aussagenlogische Formel φ in 3-KNF
Frage: Gibt es eine erfüllende Belegung für φ?

- 3-SAT ist ein Spezialfall von SAT und deshalb wie SAT in NP.
- Um zu zeigen, dass 3-SAT ebenfalls NP-vollständig ist, müssen wir also nur noch die NP-Schwere von 3-SAT nachweisen.
- Dazu zeigen wir SAT $\leq_p$ 3-SAT.

SAT $\leq_p$ 3-SAT

Lemma

SAT $\leq_p$ 3-SAT.

Beweis:

- Gegeben sei eine Formel φ in KNF.
- Wir transformieren φ in eine erfüllbarkeitsäquivalente Formel φ' in 3KNF, d.h.

$$\varphi \text{ ist erfüllbar } \Leftrightarrow \varphi' \text{ ist erfüllbar .}$$

- Aus einer 1- bzw 2-Klausel können wir leicht eine äquivalente 3-Klausel machen, indem wir ein Literal wiederholen.
- Was machen wir aber mit k-Klauseln für $k > 3$?

SAT $\leq_p$ 3-SAT

- Sei $k \geq 4$ und C eine k-Klausel der Form

$$C \;=\; \ell_1 \vee \ell_2 \vee \ell_3 \cdots \vee \ell_k \;\; .$$

- In einer Klauseltransformation ersetzen wir C durch die Teilformel

$$C' \;=\; (\ell_1 \vee \cdots \vee \ell_{k-2} \vee h) \wedge (\bar{h} \vee \ell_{k-1} \vee \ell_k) \;\; ,$$

wobei h eine zusätzlich eingeführte Hilfsvariable bezeichnet.

SAT $\leq_p$ 3-SAT

Beispiel für die Klauseltransformation:
Aus der 5-Klausel

$$x_1 \vee \bar{x}_2 \vee x_3 \vee x_4 \vee \bar{x}_5$$

wird in einem ersten Transformationsschritt die Teilformel

$$(x_1 \vee \bar{x}_2 \vee x_3 \vee h_1) \;\wedge\; (\bar{h}_1 \vee x_4 \vee \bar{x}_5) \;\; ,$$

also eine 4- und eine 3-Klausel. Auf die 4-Klausel wird die Transformation erneut angewendet. Wir erhalten die Teilformel

$$(x_1 \vee \bar{x}_2 \vee h_2) \;\wedge\; (\bar{h}_2 \vee x_3 \vee h_1) \;\wedge\; (\bar{h}_1 \vee x_4 \vee \bar{x}_5) \;\; ,$$

die nur noch 3-Klauseln enthält.

SAT $\leq_p$ 3-SAT

Nachweis der Erfüllbarkeitsäquivalenz:

φ' sei aus φ durch Ersetzen einer Klausel C durch C' entstanden.

Zu zeigen: φ erfüllbar $\Rightarrow$ φ' erfüllbar

- Sei B eine erfüllende Belegung für φ.
- B weist mindestens einem Literal aus C hat den Wert 1 zu.
- Wir unterscheiden zwei Fälle:
  1) Falls $\ell_1 \vee \cdots \vee \ell_{k-2}$ erfüllt ist, so ist φ' erfüllt, wenn wir $h = 0$ setzen.
  2) Falls $\ell_{k-1} \vee \ell_k$ erfüllt ist, so ist φ' erfüllt, wenn wir $h = 1$ setzen.
- Also ist φ' in beiden Fällen erfüllbar.

SAT $\leq_p$ 3-SAT

Zu zeigen: φ' erfüllbar $\Rightarrow$ φ erfüllbar

- Sei B' nun eine erfüllende Belegung für φ'.
- Wir unterscheiden wiederum zwei Fälle:
  - Falls B' der Variable h den Wert 0 zuweist, so muss B' einem der Literale $\ell_1, \ldots, \ell_{k-2}$ den Wert 1 zugewiesen haben.
  - Falls B' der Variable h den Wert 1 zuweist, so muss B' einem der beiden Literale ℓ_{k-1} oder k den Wert 1 zugewiesen haben.
- In beiden Fällen erfüllt B' somit auch φ.

SAT $\leq_p$ 3-SAT

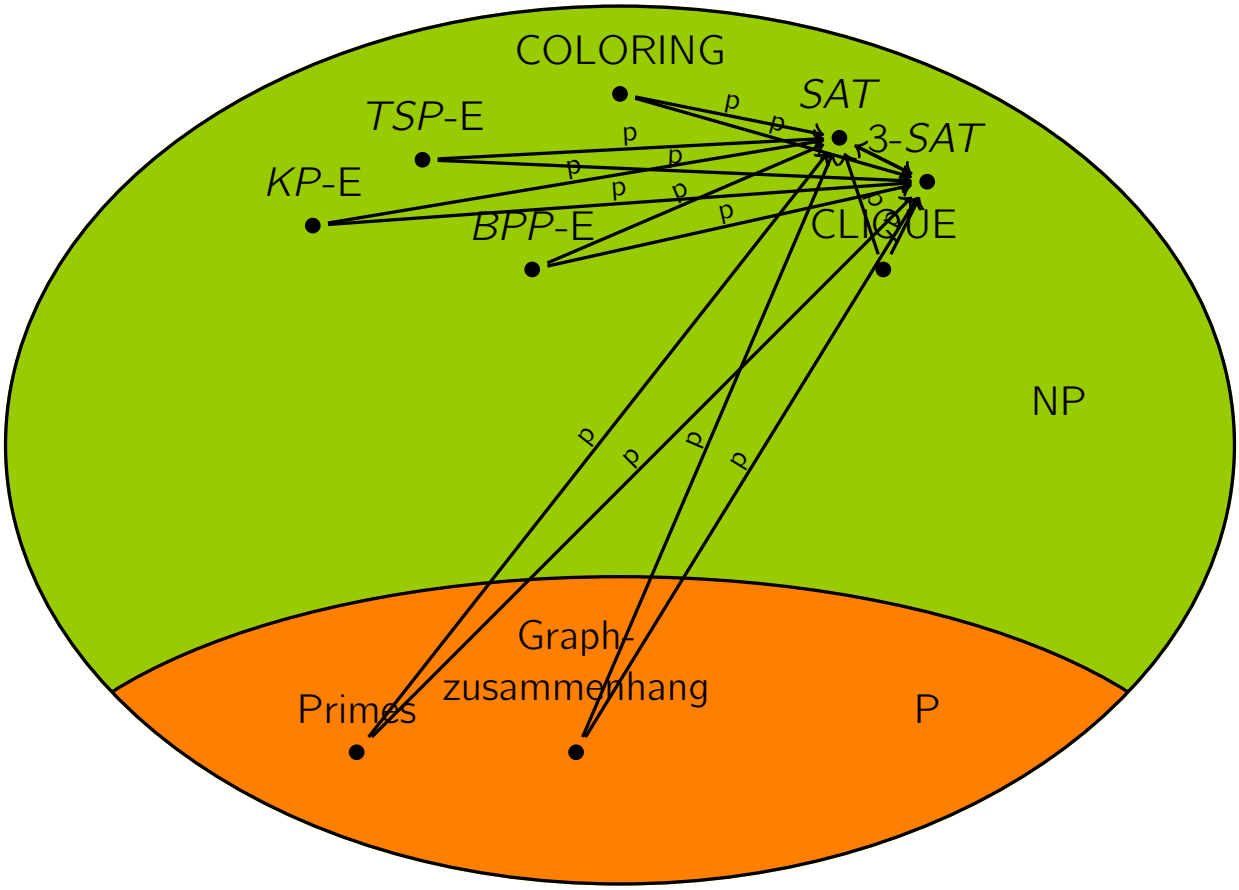
- Durch Anwendung der Klauseltransformation entstehen aus einer k-Klausel eine $(k-1)$-Klausel und eine 3-Klausel.
- Nach $k-3$ Iterationen sind aus einer k-Klausel somit $k-2$ viele 3-Klauseln entstanden.
- Diese Transformation wird solange auf die eingegebene Formel φ angewendet, bis die Formel nur noch 3-Klauseln enthält.
- Wenn n die Anzahl der Literale in φ ist, so werden insgesamt höchstens $n-3$ Klauseltransformationen benötigt.
- Die Laufzeit ist somit polynomiell beschränkt.

□

Korollar

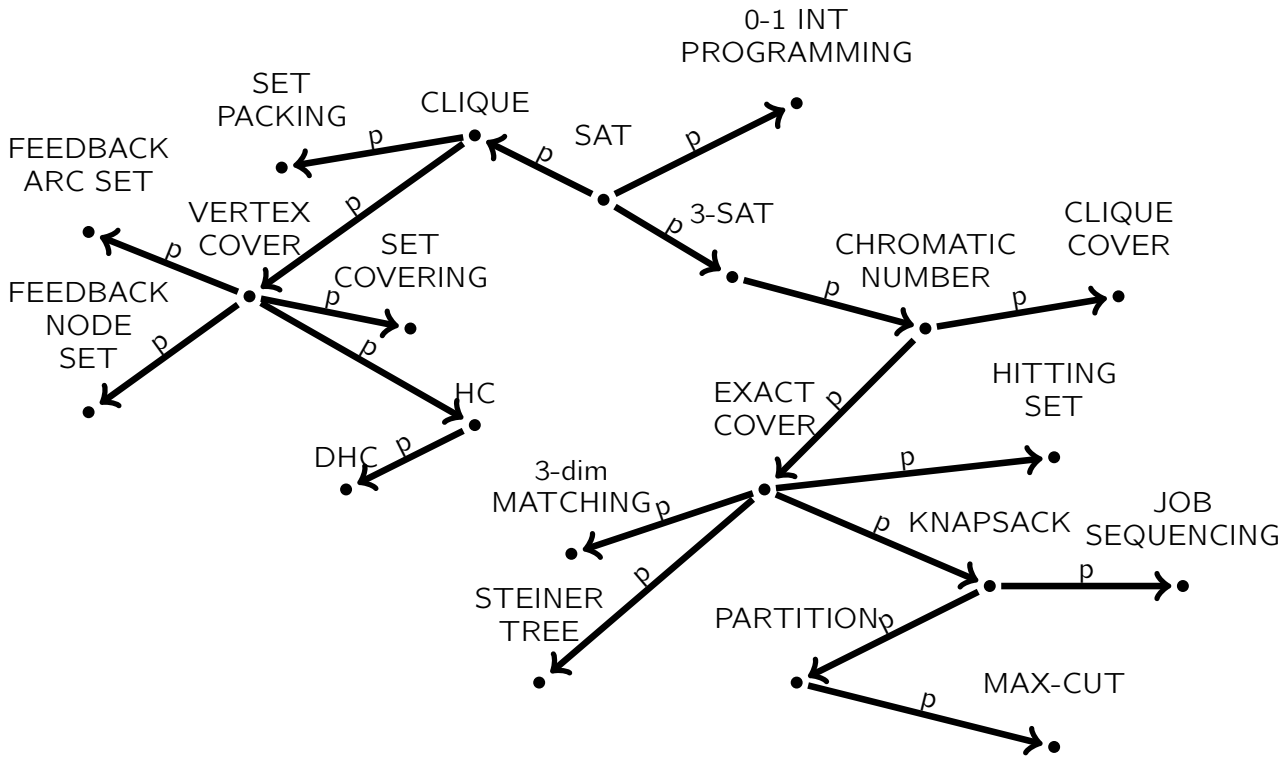
3-SAT ist NP*-vollständig.*

Die Komplexitätslandschaft



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

Karps Liste mit 21 NP-vollständigen Problemen



Es gibt mittlerweile mehrere tausende Berechnungsprobleme verschiedenster Natur, deren NP-Vollständigkeit bekannt ist.