Definition 4.10

Eine **polynomielle Reduktion** einer Sprache $A \subseteq \Sigma_1^*$ auf eine Sprache $B \subseteq \Sigma_2^*$ ist eine Many-One-Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$, die in polynomieller Zeit berechnet werden kann. Existiert eine solche Reduktion, so heißt A auf B **polynomiell reduzierbar** und wir schreiben $A \leq_p B$.

Definition 4.10

Eine **polynomielle Reduktion** einer Sprache $A \subseteq \Sigma_1^*$ auf eine Sprache $B \subseteq \Sigma_2^*$ ist eine Many-One-Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$, die in polynomieller Zeit berechnet werden kann. Existiert eine solche Reduktion, so heißt A auf B **polynomiell reduzierbar** und wir schreiben $A \leq_p B$.

Erinnerung: Many-One-Reduktion $f : \Sigma_1^* \to \Sigma_2^*$ erfüllt für alle $x \in \Sigma_1^*$:

$$x \in A \iff f(x) \in B.$$

Definition 4.10

Eine **polynomielle Reduktion** einer Sprache $A \subseteq \Sigma_1^*$ auf eine Sprache $B \subseteq \Sigma_2^*$ ist eine Many-One-Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$, die in polynomieller Zeit berechnet werden kann. Existiert eine solche Reduktion, so heißt A auf B **polynomiell reduzierbar** und wir schreiben $A \leq_p B$.

Erinnerung: Many-One-Reduktion $f : \Sigma_1^* \to \Sigma_2^*$ erfüllt für alle $x \in \Sigma_1^*$:

$$x \in A \iff f(x) \in B.$$

Polynomielle Berechenbarkeit:

$\exists k \in \mathbb{N}$: $\exists$ TM M: $\forall x \in \Sigma_1^*$: M berechnet $f(x)$ in Zeit $t_M(|x|) = O(|x|^k)$.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.

Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.

Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Beweis: Sei $A \leq_p B$ mit polynomieller Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$ und sei $B \in \mathrm{P}$.

Sei M_B die TM, die B in polynomieller Zeit entscheidet.

Sei M_f die TM, die f in polynomieller Zeit berechnet.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.
Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Beweis: Sei $A \leq_p B$ mit polynomieller Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$ und sei $B \in \mathrm{P}$.

Sei M_B die TM, die B in polynomieller Zeit entscheidet.
Sei M_f die TM, die f in polynomieller Zeit berechnet.

Konstruktion einer TM M_A für A:

1. Berechne bei einer Eingabe x zunächst $f(x)$ mittels M_f.
2. Simuliere anschließend M_B auf $f(x)$.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.
Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Beweis: Sei $A \leq_p B$ mit polynomieller Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$ und sei $B \in \mathrm{P}$.

Sei M_B die TM, die B in polynomieller Zeit entscheidet.
Sei M_f die TM, die f in polynomieller Zeit berechnet.

Konstruktion einer TM M_A für A:

1. Berechne bei einer Eingabe x zunächst $f(x)$ mittels M_f.
2. Simuliere anschließend M_B auf $f(x)$.

Korrektheit: Folgt direkt aus der Definition von $\leq_p$.

Laufzeit: Es gilt

- $t_{M_B}(n) \leq p(n)$ für ein Polynom p,
- $t_{M_f}(n) \leq q(n)$ für ein Polynom q.

Laufzeit: Es gilt

- $t_{M_B}(n) \leq p(n)$ für ein Polynom p,
- $t_{M_f}(n) \leq q(n)$ für ein Polynom q.

Laufzeit von M_A bei einer Eingabe der Länge n:
$O(q(n) + p(q(n) + n))$.

Laufzeit: Es gilt

- $t_{M_B}(n) \leq p(n)$ für ein Polynom p,
- $t_{M_f}(n) \leq q(n)$ für ein Polynom q.

Laufzeit von M_A bei einer Eingabe der Länge n:
$O(q(n) + p(q(n) + n))$.

Die **Verschachtelung zweier Polynome** ist wieder ein Polynom. □

Vertex-Cover-Problem (VC)

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Vertex-Cover-Problem (VC)

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Vertex-Cover-Problem (VC)

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Beweis: Reduktion f mit $f((G, k)) = ((G', k'))$.

Vertex-Cover-Problem (VC)

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Beweis: Reduktion f mit $f((G, k)) = ((G', k'))$.

Sei $G' = (V', E')$ mit $V' = V$.

E' enthält genau die Kanten, die E nicht enthält, d. h.

$$E' = \{\{x, y\} \mid x, y \in V, x \neq y, \{x, y\} \notin E\}.$$

Außerdem sei $k' = n - k$ für $n = |V|$.

Vertex-Cover-Problem (VC)

Eingabe: ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Beweis: Reduktion f mit $f((G, k)) = ((G', k'))$.

Sei $G' = (V', E')$ mit $V' = V$.

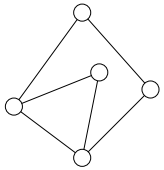
E' enthält genau die Kanten, die E nicht enthält, d. h.

$$E' = \{\{x, y\} \mid x, y \in V, x \neq y, \{x, y\} \notin E\}.$$

Außerdem sei $k' = n - k$ für $n = |V|$.

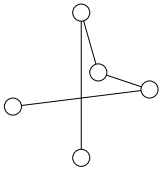
Reduktion f kann in **polynomieller Zeit berechnet** werden.

$G = (V, E)$



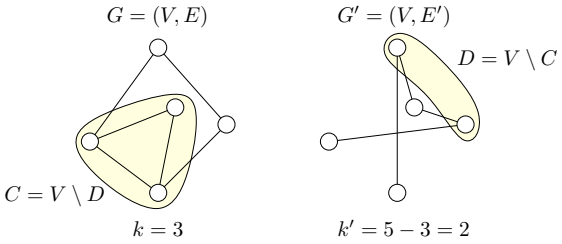
$k = 3$

$G' = (V, E')$



$k' = 5 - 3 = 2$

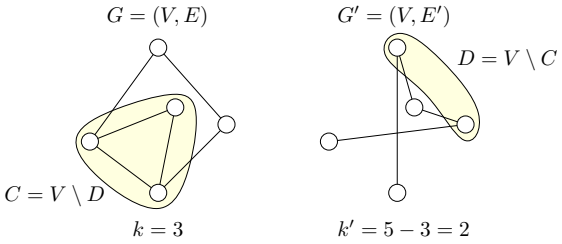
zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Rightarrow$“: Sei **C $\subseteq$ V eine k-Clique** in G.

Dann ist **D $=$ V $\setminus$ C ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.



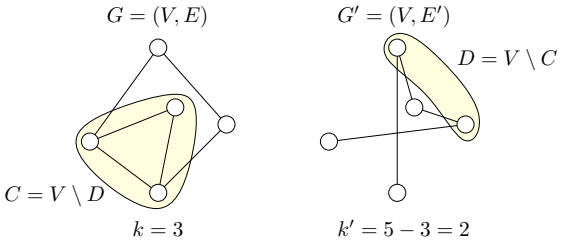
zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Rightarrow$“: Sei **$C \subseteq V$ eine k-Clique** in G.

Dann ist **$D = V \setminus C$ ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.

Annahme: D kein VC in G'.

$\Rightarrow$ Es existiert $\{x, y\} \in E'$ mit $x \notin D$ und $y \notin D$, also $x, y \in C$.



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

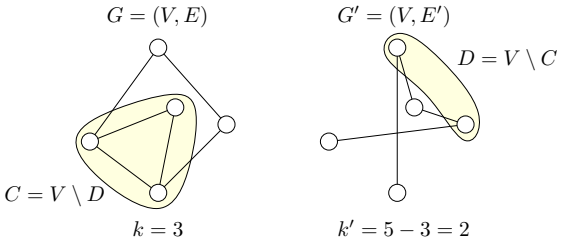
„$\Rightarrow$“: Sei **$C \subseteq V$ eine k-Clique** in G.

Dann ist **$D = V \setminus C$ ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.

Annahme: D kein VC in G'.

$\Rightarrow$ Es existiert $\{x, y\} \in E'$ mit $x \notin D$ und $y \notin D$, also $x, y \in C$.

$\Rightarrow \{x, y\} \in E$ (da C Clique in G)



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Rightarrow$“: Sei **$C \subseteq V$ eine k-Clique** in G.

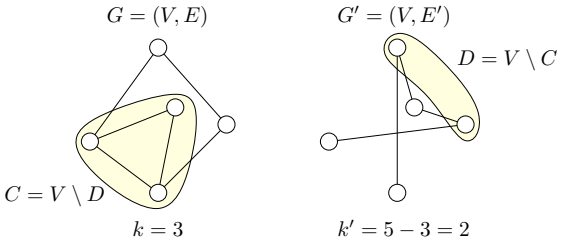
Dann ist **$D = V \setminus C$ ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.

Annahme: D kein VC in G'.

$\Rightarrow$ Es existiert $\{x, y\} \in E'$ mit $x \notin D$ und $y \notin D$, also $x, y \in C$.

$\Rightarrow \{x, y\} \in E$ (da C Clique in G)

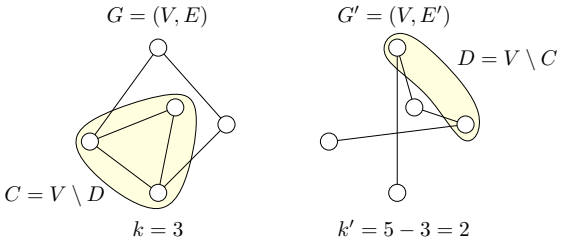
$\Rightarrow \{x, y\} \notin E'$



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„⇐“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.



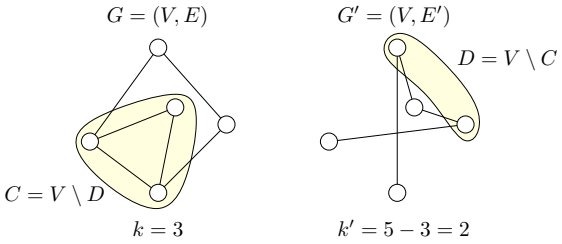
zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Leftarrow$“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.

Annahme: C keine Clique in G.

$\Rightarrow$ Es existieren $x, y \in C$ mit $\{x, y\} \notin E$.



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

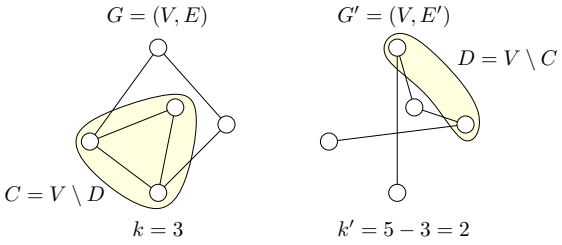
„⇐“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.

Annahme: C keine Clique in G.

$\Rightarrow$ Es existieren $x, y \in C$ mit $\{x, y\} \notin E$.

$\Rightarrow \{x, y\} \in E'$.



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„⇐“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.

Annahme: C keine Clique in G.

$\Rightarrow$ Es existieren $x, y \in C$ mit $\{x, y\} \notin E$.

$\Rightarrow \{x, y\} \in E'$.

$\Rightarrow x \in D$ oder $y \in D$ (da D Vertex Cover in G') □

Kürzeste-Wege-Problem:

Eingabe: gerichteter Graph $G = (V, E)$ mit $w\colon E \to \mathbb{N}_0$, Start $s \in V$, Ziel $t \in V$, $W \in \mathbb{N}_0$

Frage: Existiert in G ein s-t-Weg P mit Gewicht $\sum_{e \in P} w(e)$ höchstens W?

beschränktes Kürzeste-Wege-Problem (BKWP):

Eingabe: gerichteter Graph $G = (V, E)$ mit $w\colon E \to \mathbb{N}_0$, $\mathbf{c}\colon \mathbf{E} \to \mathbb{N}_0$, Start $s \in V$, Ziel $t \in V$, $W \in \mathbb{N}_0$, $\mathbf{C} \in \mathbb{N}_0$

Frage: Existiert in G ein s-t-Weg P mit Gewicht $\sum_{e \in P} w(e)$ höchstens W **und Kosten** $\sum_{\mathbf{e} \in \mathbf{P}} \mathbf{c}(\mathbf{e})$ **höchstens C**?

beschränktes Kürzeste-Wege-Problem (BKWP):

Eingabe: gerichteter Graph $G = (V, E)$ mit $w\colon E \to \mathbb{N}_0$, $\mathbf{c}\colon \mathbf{E} \to \mathbb{N}_0$, Start $s \in V$, Ziel $t \in V$, $W \in \mathbb{N}_0$, $\mathbf{C} \in \mathbb{N}_0$

Frage: Existiert in G ein s-t-Weg P mit Gewicht $\sum_{e \in P} w(e)$ höchstens W **und Kosten** $\sum_{\mathbf{e} \in \mathbf{P}} \mathbf{c(e)}$ **höchstens C**?

Theorem 4.13

Es gilt KP $\leq_p$ BKWP.

Beweis:

Eingabe $\mathcal{I}$ für KP:

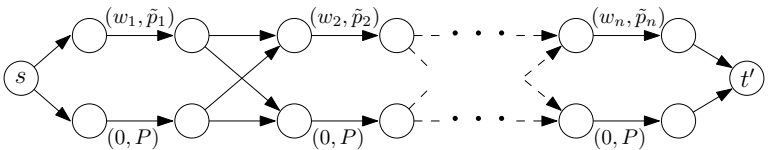
Nutzenwerte $p_1, \ldots, p_n$, Gewichte $w_1, \ldots, w_n$, Kapazität t, Nutzenschranke z

Beweis:

Eingabe $\mathcal{I}$ für KP:

Nutzenwerte $p_1, \ldots, p_n$, Gewichte $w_1, \ldots, w_n$, Kapazität t, Nutzenschranke z

Konstruiere daraus Eingabe $\mathcal{I}'$ für BKWP: $P = \max_i p_i$, $\tilde{p}_i = P - p_i$, $W = t$, $C = nP - z$

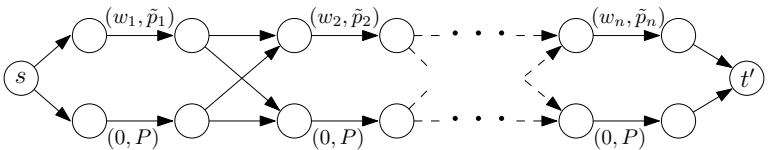


Beweis:

Eingabe $\mathcal{I}$ für KP:

Nutzenwerte $p_1, \ldots, p_n$, Gewichte $w_1, \ldots, w_n$, Kapazität t, Nutzenschranke z

Konstruiere daraus Eingabe $\mathcal{I}'$ für BKWP: $P = \max_i p_i$, $\tilde{p}_i = P - p_i$, $W = t$, $C = nP - z$

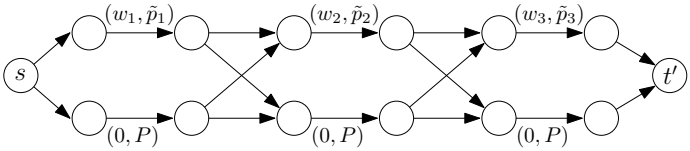


zu zeigen: $\exists I \subseteq \{1, \ldots, n\} : \sum_{i \in I} p_i \geq z$ und $\sum_{i \in I} w_i \leq t$

$\Longleftrightarrow$

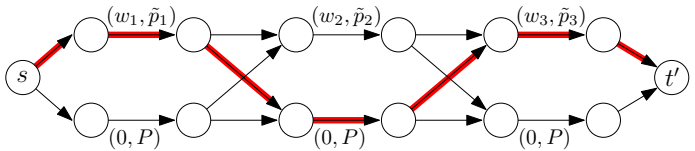
$\exists$ s-t'-Pfad T in G: $\sum_{e \in T} w(e) \leq W$ und $\sum_{e \in T} c(e) \leq C$

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



„⇒“: **Annahme:** $\exists I \subseteq \{1, \ldots, n\} : \sum_{i \in I} p_i \geq z$ und $\sum_{i \in I} w_i \leq t$

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$

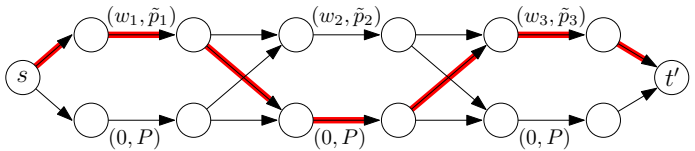


$I = \{1, 3\}$

„⇒“: **Annahme:** $\exists I \subseteq \{1, \ldots, n\} : \sum_{i \in I} p_i \geq z$ und $\sum_{i \in I} w_i \leq t$

Konstruiere s-t'-Weg T in G, der **für $i \in I$ die $(w_i, \tilde{p}_i)$-Kante** nutzt und **für $i \notin I$ die $(0, P)$-Kante**.

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



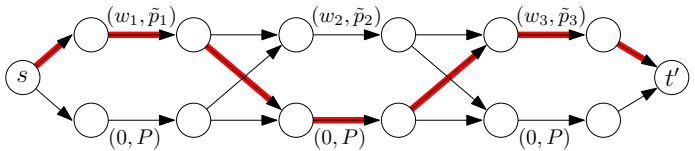
$I = \{1, 3\}$

„⇒“: **Annahme:** $\exists I \subseteq \{1, \ldots, n\} : \sum_{i \in I} p_i \geq z$ und $\sum_{i \in I} w_i \leq t$

Konstruiere s-t'-Weg T in G, der **für $i \in I$ die $(w_i, \tilde{p}_i)$-Kante** nutzt und **für $i \notin I$ die $(0, P)$-Kante**.

Es gilt $\sum_{i \in I} w_i + \sum_{i \notin I} 0 = \sum_{i \in I} w_i \leq t = W$

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



$I = \{1, 3\}$

„⇒“: **Annahme:** $\exists I \subseteq \{1, \ldots, n\} : \sum_{i \in I} p_i \geq z$ und $\sum_{i \in I} w_i \leq t$

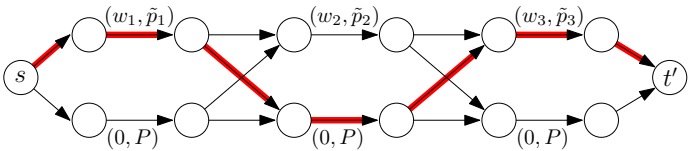
Konstruiere s-t'-Weg T in G, der **für $i \in I$ die $(w_i, \tilde{p}_i)$-Kante** nutzt und **für $i \notin I$ die $(0, P)$-Kante**.

Es gilt $\sum_{i \in I} w_i + \sum_{i \notin I} 0 = \sum_{i \in I} w_i \leq t = W$

und

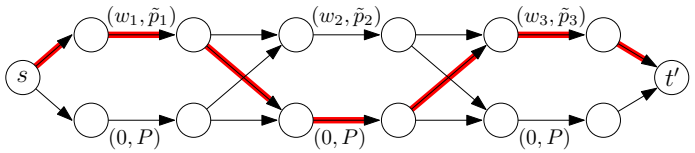
$$\sum_{i \in I} \tilde{p}_i + \sum_{i \notin I} P = \sum_{i \in I} (P - p_i) + \sum_{i \notin I} P = nP - \sum_{i \in I} p_i \leq nP - z = C.$$

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



„⇐“: **Annahme:** $\exists$ s-t'-Weg mit $w(T) \leq W$ und $c(T) \leq C$.

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$

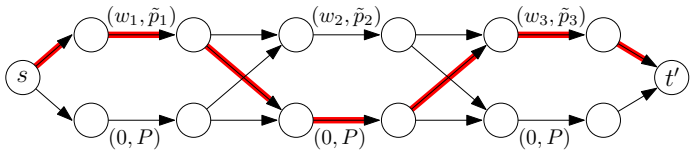


$I = \{1, 3\}$

„⇐“: **Annahme:** $\exists$ s-t'-Weg mit $w(T) \leq W$ und $c(T) \leq C$.

Konstruiere Lösung I für KP: Es sei **$i \in I$ genau dann, wenn T die $(w_i, \tilde{p}_i)$-Kante nutzt**.

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



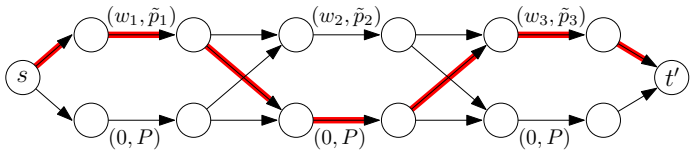
$I = \{1, 3\}$

„⇐“: **Annahme:** $\exists$ s-t'-Weg mit $w(T) \leq W$ und $c(T) \leq C$.

Konstruiere Lösung I für KP: Es sei **$i \in I$ genau dann, wenn T die $(w_i, \tilde{p}_i)$-Kante nutzt**.

Es gilt $\sum_{i \in I} w_i = \sum_{e \in T} w(e) \leq W = t$

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



$I = \{1, 3\}$

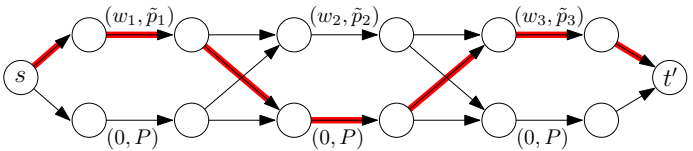
„⇐“: **Annahme:** $\exists$ s-t'-Weg mit $w(T) \leq W$ und $c(T) \leq C$.

Konstruiere Lösung I für KP: Es sei **$i \in I$ genau dann, wenn T die $(w_i, \tilde{p}_i)$-Kante nutzt**.

Es gilt $\sum_{i \in I} w_i = \sum_{e \in T} w(e) \leq W = t$ und

$$c(T) = \sum_{i \in I} \tilde{p}_i + \sum_{i \notin I} P = \sum_{i \in I} (P - p_i) + \sum_{i \notin I} P = nP - \sum_{i \in I} p_i \leq C = nP - z.$$

$P = \max_i p_i$ und $\tilde{p}_i = P - p_i$ $W = t$ $C = nP - z$



$I = \{1, 3\}$

„⇐“: **Annahme:** $\exists$ s-t'-Weg mit $w(T) \leq W$ und $c(T) \leq C$.

Konstruiere Lösung I für KP: Es sei **$i \in I$ genau dann, wenn T die $(w_i, \tilde{p}_i)$-Kante nutzt**.

Es gilt $\sum_{i \in I} w_i = \sum_{e \in T} w(e) \leq W = t$ und

$$c(T) = \sum_{i \in I} \tilde{p}_i + \sum_{i \notin I} P = \sum_{i \in I} (P - p_i) + \sum_{i \notin I} P = nP - \sum_{i \in I} p_i \leq C = nP - z.$$

Daraus folgt

$$\sum_{i \in I} p_i \geq z. \qquad \square$$

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \text{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \text{NP}$, so heißt L **NP-vollständig**.

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \text{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \text{NP}$, so heißt L **NP-vollständig**.

Theorem 4.15

Gibt es eine NP-schwere Sprache $L \in \text{P}$, so gilt $\text{P} = \text{NP}$.

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \mathrm{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \mathrm{NP}$, so heißt L **NP-vollständig**.

Theorem 4.15

Gibt es eine NP-schwere Sprache $L \in \mathrm{P}$, so gilt $\mathrm{P} = \mathrm{NP}$.

Beweis: Sei $L' \in \mathrm{NP}$ beliebig. Dann gilt $L' \leq_p L$. Wegen $L \in \mathrm{P}$ folgt daraus $L' \in \mathrm{P}$. □

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \mathrm{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \mathrm{NP}$, so heißt L **NP-vollständig**.

Theorem 4.15

Gibt es eine NP-schwere Sprache $L \in \mathrm{P}$, so gilt $\mathrm{P} = \mathrm{NP}$.

Beweis: Sei $L' \in \mathrm{NP}$ beliebig. Dann gilt $L' \leq_p L$. Wegen $L \in \mathrm{P}$ folgt daraus $L' \in \mathrm{P}$. □

Korollar 4.16

Es sei L eine NP-vollständige Sprache. Dann gilt $L \in \mathrm{P}$ genau dann, wenn $\mathrm{P} = \mathrm{NP}$ gilt.

Definition 4.17

Eine Formel der Form x oder $\neg x$ für eine Variable x heißt **Literal**. Ein Literal x nennen wir **positives Literal** und ein Literal $\neg x$ nennen wir **negatives Literal**.

Eine aussagenlogische Formel φ ist in **konjunktiver Normalform (KNF)**, wenn sie eine Konjunktion von Disjunktionen von Literalen ist, d. h. wenn sie die Gestalt

$$\varphi = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} \ell_{i,j} \right)$$

hat, wobei $n, m_1, \ldots, m_n \in \mathbb{N}$ gilt und $\ell_{i,j}$ für jedes i und j ein Literal ist.

Die Teilformeln $\bigvee_{j=1}^{m_i} \ell_{i,j}$ nennen wir die **Klauseln** von φ.

Definition 4.17

Eine Formel der Form x oder $\neg x$ für eine Variable x heißt **Literal**. Ein Literal x nennen wir **positives Literal** und ein Literal $\neg x$ nennen wir **negatives Literal**.

Eine aussagenlogische Formel φ ist in **konjunktiver Normalform (KNF)**, wenn sie eine Konjunktion von Disjunktionen von Literalen ist, d. h. wenn sie die Gestalt

$$\varphi = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} \ell_{i,j} \right)$$

hat, wobei $n, m_1, \ldots, m_n \in \mathbb{N}$ gilt und $\ell_{i,j}$ für jedes i und j ein Literal ist.

Die Teilformeln $\bigvee_{j=1}^{m_i} \ell_{i,j}$ nennen wir die **Klauseln** von φ.

SAT: Entscheide für gegebene Formel φ in KNF, ob sie eine **erfüllende Belegung** besitzt.

Theorem 4.18 (Satz von Cook und Levin)

SAT ist NP-vollständig.

Theorem 4.18 (Satz von Cook und Levin)

SAT ist NP-vollständig.

Beweis:

SAT $\in$ NP: Beweis analog zu CLIQUE und zum Rucksackproblem

Theorem 4.18 (Satz von Cook und Levin)

SAT ist NP-vollständig.

Beweis:

SAT $\in$ NP: Beweis analog zu CLIQUE und zum Rucksackproblem

SAT ist NP-schwer: Sei $L \in$ NP beliebig. Zu zeigen: $L \leq_p$ SAT.

Theorem 4.18 (Satz von Cook und Levin)

SAT ist NP-vollständig.

Beweis:

SAT $\in$ NP: Beweis analog zu CLIQUE und zum Rucksackproblem

SAT ist NP-schwer: Sei $L \in$ NP beliebig. Zu zeigen: $L \leq_p$ SAT.

Es gibt NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die **L in polynomieller Zeit entscheidet**.
Sei p ein Polynom, sodass $t_M(n) \leq p(n)$ für alle $n \in \mathbb{N}$ gilt.

Theorem 4.18 (Satz von Cook und Levin)

SAT ist NP-vollständig.

Beweis:

SAT $\in$ NP: Beweis analog zu CLIQUE und zum Rucksackproblem

SAT ist NP-schwer: Sei $L \in \text{NP}$ beliebig. Zu zeigen: $L \leq_p \text{SAT}$.

Es gibt NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die **L in polynomieller Zeit entscheidet**.
Sei p ein Polynom, sodass $t_M(n) \leq p(n)$ für alle $n \in \mathbb{N}$ gilt.

Ziel: Finde polynomiell berechenbare Funktion $f\colon \Sigma^* \to \{0,1\}^*$, die $x \in \Sigma^*$ in eine aussagenlogische Formel $f(x)$ in KNF übersetzt, sodass

$$x \in L \iff f(x) \text{ ist erfüllbar.}$$

Idee: Simuliere mithilfe der Formel $\varphi = f(x)$ die NTM M auf der Eingabe x.

Idee: Simuliere mithilfe der Formel $\varphi = f(x)$ die NTM M auf der Eingabe x.

Seien K und K' Konfigurationen von M.
Gibt es in K einen Rechenschritt, der zu K' führt, so schreiben wir $K \vdash K'$.

Idee: Simuliere mithilfe der Formel $\varphi = f(x)$ die NTM M auf der Eingabe x.

Seien K und K' Konfigurationen von M.
Gibt es in K einen Rechenschritt, der zu K' führt, so schreiben wir $K \vdash K'$.

K_0 sei die **initiale Konfiguration**

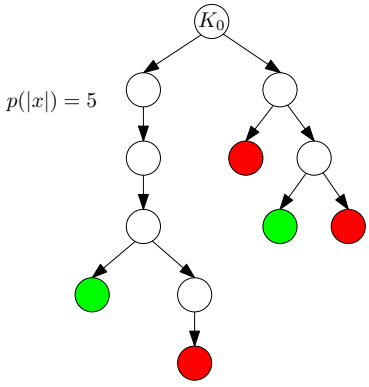
Idee: Simuliere mithilfe der Formel $\varphi = f(x)$ die NTM M auf der Eingabe x.

Seien K und K' Konfigurationen von M.
Gibt es in K einen Rechenschritt, der zu K' führt, so schreiben wir $K \vdash K'$.

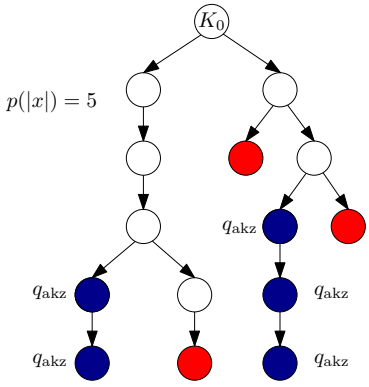
K_0 sei die **initiale Konfiguration**

φ soll genau dann erfüllbar sein, wenn es eine Folge von Konfigurationen $K_1, K_2, \ldots, K_\ell$ mit $\ell \leq p(n)$ und $K_0 \vdash K_1 \vdash K_2 \vdash \ldots \vdash K_\ell$ gibt, wobei K_ℓ eine **akzeptierende Endkonfiguration** ist.

Modifikation von M: Füge neuen Zustand q_{akz} hinzu. Geht M in $\bar{q}$ über und akzeptiert x, so wird stattdessen **q_{akz} erreicht und nicht mehr verlassen**.



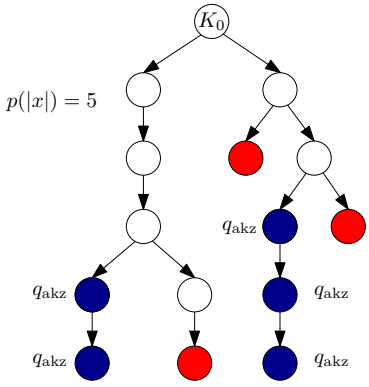
Modifikation von M: Füge neuen Zustand q_{akz} hinzu. Geht M in $\bar{q}$ über und akzeptiert x, so wird stattdessen **q_{akz} erreicht und nicht mehr verlassen**.



Modifikation von M: Füge neuen Zustand q_{akz} hinzu.

Geht M in $\bar{q}$ über und akzeptiert x, so wird stattdessen **q_{akz} erreicht und nicht mehr verlassen**.

$\Rightarrow$ Nur noch Rechenwege der **Länge genau $p(n)$** relevant.

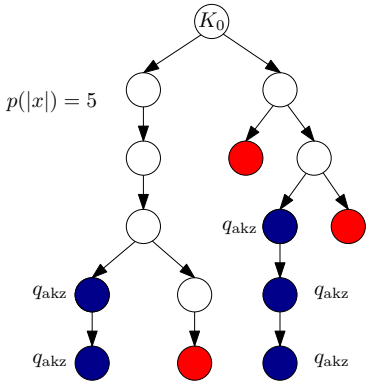


Modifikation von M: Füge neuen Zustand q_{akz} hinzu. Geht M in $\bar{q}$ über und akzeptiert x, so wird stattdessen **q_{akz} erreicht und nicht mehr verlassen**.

$\Rightarrow$ Nur noch Rechenwege der **Länge genau $p(n)$** relevant.

Ziel der Konstruktion:

Die Formel φ ist **genau dann erfüllbar**, wenn es eine Folge von Konfigurationen $K_1, K_2, \ldots, K_{p(n)}$ von M mit $K_0 \vdash K_1 \vdash K_2 \vdash \ldots \vdash K_{p(n)}$ gibt, wobei in $K_{p(n)}$ der Zustand q_{akz} angenommen wird und K_0 die initiale Konfiguration von M bei Eingabe x ist.



Definition der Variablen: Die Variablen codieren die Konfigurationen $K_0, \ldots, K_{p(n)}$.

Definition der Variablen: Die Variablen codieren die Konfigurationen $K_0, \ldots, K_{p(n)}$.

- $Q(t, q)$ für $t \in \{0, \ldots, p(n)\}$ und $q \in Q$

$$Q(t,q) = \begin{cases} 1 & \text{falls in } K_t \text{ Zustand q angenommen wird} \\ 0 & \text{sonst} \end{cases}$$

Definition der Variablen: Die Variablen codieren die Konfigurationen $K_0, \ldots, K_{p(n)}$.

- $Q(t,q)$ für $t \in \{0, \ldots, p(n)\}$ und $q \in Q$

$$Q(t,q) = \begin{cases} 1 & \text{falls in } K_t \text{ Zustand q angenommen wird} \\ 0 & \text{sonst} \end{cases}$$

- $H(t,j)$ für $t \in \{0, \ldots, p(n)\}$ und $j \in \{-p(n), \ldots, p(n)\}$

$$H(t,j) = \begin{cases} 1 & \text{falls Kopf in } K_t \text{ auf Zelle j steht} \\ 0 & \text{sonst} \end{cases}$$

Definition der Variablen: Die Variablen codieren die Konfigurationen $K_0, \ldots, K_{p(n)}$.

- $Q(t,q)$ für $t \in \{0,\ldots,p(n)\}$ und $q \in Q$

$$Q(t,q) = \begin{cases} 1 & \text{falls in } K_t \text{ Zustand q angenommen wird} \\ 0 & \text{sonst} \end{cases}$$

- $H(t,j)$ für $t \in \{0,\ldots,p(n)\}$ und $j \in \{-p(n),\ldots,p(n)\}$

$$H(t,j) = \begin{cases} 1 & \text{falls Kopf in } K_t \text{ auf Zelle j steht} \\ 0 & \text{sonst} \end{cases}$$

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

$$S(t,j,a) = \begin{cases} 1 & \text{falls Zelle j in } K_t \text{ das Zeichen a enthält} \\ 0 & \text{sonst} \end{cases}$$

Definition der Variablen: Die Variablen codieren die Konfigurationen $K_0, \ldots, K_{p(n)}$.

- $Q(t,q)$ für $t \in \{0,\ldots,p(n)\}$ und $q \in Q$

$$Q(t,q) = \begin{cases} 1 & \text{falls } \mathbf{in\ K_t\ Zustand\ q\ angenommen\ wird} \\ 0 & \text{sonst} \end{cases}$$

- $H(t,j)$ für $t \in \{0,\ldots,p(n)\}$ und $j \in \{-p(n),\ldots,p(n)\}$

$$H(t,j) = \begin{cases} 1 & \text{falls } \mathbf{Kopf\ in\ K_t\ auf\ Zelle\ j} \text{ steht} \\ 0 & \text{sonst} \end{cases}$$

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

$$S(t,j,a) = \begin{cases} 1 & \text{falls } \mathbf{Zelle\ j\ in\ K_t\ das\ Zeichen\ a\ enthält} \\ 0 & \text{sonst} \end{cases}$$

Anzahl Variablen polynomiell in n, denn p ist Polynom und $|Q|$ und $|\Gamma|$ sind Konstanten.

Codierung einzelner Konfigurationen:

Ziel: Stelle sicher, dass die Variablen für festes $t \in \{0, \ldots, p(n)\}$ **eine Konfiguration K_t codieren**.

Codierung einzelner Konfigurationen:

Ziel: Stelle sicher, dass die Variablen für festes $t \in \{0, \ldots, p(n)\}$ **eine Konfiguration K_t codieren**.

Konkret: Jede erfüllende Belegung von φ muss dergestalt sein,

- dass es **genau ein $q \in Q$ mit $Q(t, q) = 1$** gibt,

Codierung einzelner Konfigurationen:

Ziel: Stelle sicher, dass die Variablen für festes $t \in \{0, \ldots, p(n)\}$ **eine Konfiguration K_t codieren**.

Konkret: Jede erfüllende Belegung von φ muss dergestalt sein,

- dass es **genau ein $q \in Q$ mit $Q(t, q) = 1$** gibt,
- dass es **genau ein $j \in \{-p(n), \ldots, p(n)\}$ mit $H(t, j) = 1$** gibt und

Codierung einzelner Konfigurationen:

Ziel: Stelle sicher, dass die Variablen für festes $t \in \{0, \ldots, p(n)\}$ **eine Konfiguration K_t codieren**.

Konkret: Jede erfüllende Belegung von φ muss dergestalt sein,

- dass es **genau ein $q \in Q$ mit $Q(t, q) = 1$** gibt,
- dass es **genau ein $j \in \{-p(n), \ldots, p(n)\}$ mit $H(t, j) = 1$** gibt und
- dass es für **jedes $j \in \{-p(n), \ldots, p(n)\}$ genau ein $a \in \Gamma$ mit $S(t, j, a) = 1$** gibt.

Für diese Bedingungen muss jeweils für eine Variablenmenge codiert werden, dass **genau eine der Variablen auf 1 und alle anderen auf 0 gesetzt sind**.

Für diese Bedingungen muss jeweils für eine Variablenmenge codiert werden, dass **genau eine der Variablen auf 1 und alle anderen auf 0 gesetzt sind**.

Für eine Variablenmenge $\{y_1, \ldots, y_m\}$ kann dies durch die Formel

$$(y_1 \vee \ldots \vee y_m) \wedge \bigwedge_{i \neq j} (\neg y_i \vee \neg y_j)$$

in KNF erreicht werden.

Für diese Bedingungen muss jeweils für eine Variablenmenge codiert werden, dass **genau eine der Variablen auf 1 und alle anderen auf 0 gesetzt sind**.

Für eine Variablenmenge $\{y_1, \ldots, y_m\}$ kann dies durch die Formel

$$(y_1 \vee \ldots \vee y_m) \wedge \bigwedge_{i \neq j} (\neg y_i \vee \neg y_j)$$

in KNF erreicht werden.

Länge dieser Formel: $O(m^2)$

Für diese Bedingungen muss jeweils für eine Variablenmenge codiert werden, dass **genau eine der Variablen auf 1 und alle anderen auf 0 gesetzt sind**.

Für eine Variablenmenge $\{y_1, \ldots, y_m\}$ kann dies durch die Formel

$$(y_1 \vee \ldots \vee y_m) \wedge \bigwedge_{i \neq j} (\neg y_i \vee \neg y_j)$$

in KNF erreicht werden.

Länge dieser Formel: $O(m^2)$

Wir können auf diese Weise die o. g. drei Bedingungen mit einer **Formel φ_t der Länge $O(p(n)^2)$** in KNF codieren.

Wir müssen als nächstes codieren, dass **K_t für jedes $t \in \{1, \ldots, p(n)\}$ eine direkte Nachfolgekonfiguration von K_{t-1} ist**.

Wir müssen als nächstes codieren, dass **K_t für jedes $t \in \{1, \ldots, p(n)\}$ eine direkte Nachfolgekonfiguration von K_{t-1} ist**.

Bandinhalt darf sich nur in der Zelle ändern, an der sich der Kopf befindet:

$$\bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \big((S(t-1,j,a) \wedge \neg H(t-1,j)) \Rightarrow S(t,j,a) \big).$$

Wir müssen als nächstes codieren, dass **K_t für jedes $t \in \{1, \ldots, p(n)\}$ eine direkte Nachfolgekonfiguration von K_{t-1} ist**.

Bandinhalt darf sich nur in der Zelle ändern, an der sich der Kopf befindet:

$$\bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \big((S(t-1,j,a) \wedge \neg H(t-1,j)) \Rightarrow S(t,j,a) \big).$$

$A \Rightarrow B$ kann durch den äquivalenten Ausdruck $\neg A \vee B$ ersetzt werden.

Wir müssen als nächstes codieren, dass **K_t für jedes $t \in \{1, \ldots, p(n)\}$ eine direkte Nachfolgekonfiguration von K_{t-1} ist**.

Bandinhalt darf sich nur in der Zelle ändern, an der sich der Kopf befindet:

$$\bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \big((S(t-1,j,a) \wedge \neg H(t-1,j)) \Rightarrow S(t,j,a) \big).$$

$A \Rightarrow B$ kann durch den äquivalenten Ausdruck $\neg A \vee B$ ersetzt werden.

Wendet man dann noch das **De Morgansche Gesetz** an, dass $\neg(A \wedge B)$ äquivalent zu $\neg A \vee \neg B$ ist, so erhält man die folgende Formel in KNF:

$$\bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \big(\neg S(t-1,j,a) \vee H(t-1,j) \vee S(t,j,a) \big).$$

Wir müssen als nächstes codieren, dass **K_t für jedes $t \in \{1, \ldots, p(n)\}$ eine direkte Nachfolgekonfiguration von K_{t-1} ist**.

Bandinhalt darf sich nur in der Zelle ändern, an der sich der Kopf befindet:

$$\bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \big((S(t-1,j,a) \wedge \neg H(t-1,j)) \Rightarrow S(t,j,a) \big).$$

$A \Rightarrow B$ kann durch den äquivalenten Ausdruck $\neg A \vee B$ ersetzt werden.

Wendet man dann noch das **De Morgansche Gesetz** an, dass $\neg(A \wedge B)$ äquivalent zu $\neg A \vee \neg B$ ist, so erhält man die folgende Formel in KNF:

$$\bigwedge_{j=-p(n)}^{p(n)} \bigwedge_{a \in \Gamma} \big(\neg S(t-1,j,a) \vee H(t-1,j) \vee S(t,j,a) \big).$$

Länge: $O(p(n))$

Nun müssen wir noch erreichen, dass **im Schritt von K_{t-1} zu K_t ein durch δ beschriebener Rechenschritt ausgeführt wird**.

Nun müssen wir noch erreichen, dass **im Schritt von K_{t-1} zu K_t ein durch δ beschriebener Rechenschritt ausgeführt wird**.

Betrachte für jedes $q \in Q$, jedes $j \in \{-p(n), \ldots, p(n)\}$ und jedes $a \in \Gamma$ die Formel

$$\begin{aligned} & \big(Q(t-1,q) \wedge H(t-1,j) \wedge S(t-1,j,a)\big) \\ \Rightarrow \quad & \bigvee_{((q,a),(q',a',D))\in\delta} \big(Q(t,q') \wedge H(t,j+D) \wedge S(t,j,a')\big), \end{aligned}$$

wobei $D \in \{-1, 0, +1\}$ statt $D \in \{L, N, R\}$ die Bewegung des Kopfes angibt.

Nun müssen wir noch erreichen, dass **im Schritt von K_{t-1} zu K_t ein durch δ beschriebener Rechenschritt ausgeführt wird**.

Betrachte für jedes $q \in Q$, jedes $j \in \{-p(n), \ldots, p(n)\}$ und jedes $a \in \Gamma$ die Formel

$$\begin{aligned} &\big(Q(t-1,q) \wedge H(t-1,j) \wedge S(t-1,j,a)\big) \\ \Rightarrow \quad &\bigvee_{((q,a),(q',a',D))\in\delta} \big(Q(t,q') \wedge H(t,j+D) \wedge S(t,j,a')\big), \end{aligned}$$

wobei $D \in \{-1, 0, +1\}$ statt $D \in \{L, N, R\}$ die Bewegung des Kopfes angibt.

Es gibt eine äquivalente Formel in KNF, die konstante Länge besitzt.

Nun müssen wir noch erreichen, dass **im Schritt von K_{t-1} zu K_t ein durch δ beschriebener Rechenschritt ausgeführt wird**.

Betrachte für jedes $q \in Q$, jedes $j \in \{-p(n), \ldots, p(n)\}$ und jedes $a \in \Gamma$ die Formel

$$\begin{aligned} &\big(Q(t-1,q) \wedge H(t-1,j) \wedge S(t-1,j,a)\big) \\ \Rightarrow \quad &\bigvee_{((q,a),(q',a',D))\in\delta} \big(Q(t,q') \wedge H(t,j+D) \wedge S(t,j,a')\big), \end{aligned}$$

wobei $D \in \{-1, 0, +1\}$ statt $D \in \{L, N, R\}$ die Bewegung des Kopfes angibt.

Es gibt eine äquivalente Formel in KNF, die konstante Länge besitzt. Konjugieren wir die Formeln für jede Wahl von q, j und a, so erhalten wir eine **Formel der Länge $O(p(n))$**.

Nun müssen wir noch erreichen, dass **im Schritt von K_{t-1} zu K_t ein durch δ beschriebener Rechenschritt ausgeführt wird**.

Betrachte für jedes $q \in Q$, jedes $j \in \{-p(n), \ldots, p(n)\}$ und jedes $a \in \Gamma$ die Formel

$$\begin{aligned} &\big(Q(t-1,q) \wedge H(t-1,j) \wedge S(t-1,j,a)\big) \\ \Rightarrow \quad &\bigvee_{((q,a),(q',a',D))\in\delta} \big(Q(t,q') \wedge H(t,j+D) \wedge S(t,j,a')\big), \end{aligned}$$

wobei $D \in \{-1, 0, +1\}$ statt $D \in \{L, N, R\}$ die Bewegung des Kopfes angibt.

Es gibt eine äquivalente Formel in KNF, die konstante Länge besitzt. Konjugieren wir die Formeln für jede Wahl von q, j und a, so erhalten wir eine **Formel der Länge $O(p(n))$**.

Zusammen erhalten wir somit für jedes $t \in \{1, \ldots, p(n)\}$ eine **Formel $\varphi_{\to t}$ in KNF, die codiert, dass K_t eine direkte Nachfolgekonfiguration von K_{t-1} ist**.

Codierung der initialen Konfiguration:

$$\varphi_{\text{init}} = Q(0, q_0) \wedge H(0, 0) \wedge \bigwedge_{i=1}^{n} S(0, i-1, x_i) \wedge \bigwedge_{j=-p(n)}^{-1} S(0, j, \square) \wedge \bigwedge_{j=n}^{p(n)} S(0, j, \square)$$

Codierung der initialen Konfiguration:

$$\varphi_{\text{init}} = Q(0, q_0) \wedge H(0,0) \wedge \bigwedge_{i=1}^{n} S(0, i-1, x_i) \wedge \bigwedge_{j=-p(n)}^{-1} S(0, j, \square) \wedge \bigwedge_{j=n}^{p(n)} S(0, j, \square)$$

Zusammensetzen der Formel:

Setze alle Teilformeln zusammen und codiere zusätzlich, dass nach $p(n)$ Schritten der Zustand q_{akz} erreicht werden soll:

$$\varphi = \varphi_{\text{init}} \wedge \left(\bigwedge_{t=0}^{p(n)} \varphi_t \right) \wedge \left(\bigwedge_{t=1}^{p(n)} \varphi_{\to t} \right) \wedge Q(p(n), q_{\text{akz}}).$$

Codierung der initialen Konfiguration:

$$\varphi_{\text{init}} = Q(0, q_0) \wedge H(0,0) \wedge \bigwedge_{i=1}^{n} S(0, i-1, x_i) \wedge \bigwedge_{j=-p(n)}^{-1} S(0, j, \square) \wedge \bigwedge_{j=n}^{p(n)} S(0, j, \square)$$

Zusammensetzen der Formel:

Setze alle Teilformeln zusammen und codiere zusätzlich, dass nach $p(n)$ Schritten der Zustand q_{akz} erreicht werden soll:

$$\varphi = \varphi_{\text{init}} \wedge \left(\bigwedge_{t=0}^{p(n)} \varphi_t \right) \wedge \left(\bigwedge_{t=1}^{p(n)} \varphi_{\to t} \right) \wedge Q(p(n), q_{\text{akz}}).$$

Diese Formel ist in KNF und besitzt eine **Länge von $O(p(n)^3)$**.

Codierung der initialen Konfiguration:

$$\varphi_{\text{init}} = Q(0, q_0) \wedge H(0,0) \wedge \bigwedge_{i=1}^{n} S(0, i-1, x_i) \wedge \bigwedge_{j=-p(n)}^{-1} S(0, j, \square) \wedge \bigwedge_{j=n}^{p(n)} S(0, j, \square)$$

Zusammensetzen der Formel:

Setze alle Teilformeln zusammen und codiere zusätzlich, dass nach $p(n)$ Schritten der Zustand q_{akz} erreicht werden soll:

$$\varphi = \varphi_{\text{init}} \wedge \left(\bigwedge_{t=0}^{p(n)} \varphi_t \right) \wedge \left(\bigwedge_{t=1}^{p(n)} \varphi_{\to t} \right) \wedge Q(p(n), q_{\text{akz}}).$$

Diese Formel ist in KNF und besitzt eine **Länge von $O(p(n)^3)$**.

Sie kann für ein gegebenes $x \in \Sigma^*$ **in polynomieller Zeit konstruiert** werden.

Es gibt genau dann eine **erfüllende Belegung für φ**, wenn es eine Folge von Konfigurationen $K_1, K_2, \ldots, K_{p(n)}$ von M mit $K_0 \vdash K_1 \vdash K_2 \vdash \ldots \vdash K_{p(n)}$ gibt, wobei **in $K_{p(n)}$ der Zustand q_{akz}** angenommen wird und K_0 die initiale Konfiguration bei Eingabe x ist.

Es gibt genau dann eine **erfüllende Belegung für φ**, wenn es eine Folge von Konfigurationen $K_1, K_2, \ldots, K_{p(n)}$ von M mit $K_0 \vdash K_1 \vdash K_2 \vdash \ldots \vdash K_{p(n)}$ gibt, wobei **in $K_{p(n)}$ der Zustand q_{akz}** angenommen wird und K_0 die initiale Konfiguration bei Eingabe x ist.

Dies ist genau dann der Fall, wenn $x \in L$ gilt.

Somit gilt $L \leq_p$ SAT. □