Definition von ϕ_{move}

Ziel: Zeile $i+1$ muss Nachfolgekonfiguration von Zeile i sein.

- Definieren Fenster F der Größe 2×3.
- (i,j)-Fenster besitzt Einträge $(i,j-1),(i,j),(i,j+1)$ und $(i+1,j-1),(i+1,j),(i+1,j+1)$.
- Tabelle T besitzt (i,j)-Fenster für $i=1,\ldots,n^k$, $j=2,\ldots,n^k$.
- Fenster F heißt legal gwd F's Einträge δ nicht widersprechen.

Beispiele für legale Fenster

Sei δ wie folgt definiert

- $\delta(q_1, a) = \{(q_1, b, R)\}$.
- $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$.

a	q_1	b
q_2	a	c

legal

a	q_1	b
a	a	q_2

legal

a	b	b
a	a	b

nicht legal

a	a	q_1
a	a	b

legal

a	q_1	b
q_1	a	a

nicht legal

$\sqcup$	b	a
$\sqcup$	b	a

legal

a	q_1	b
q_2	b	q_2

nicht legal

a	b	a
a	b	q_2

legal

b	b	b
c	b	b

legal

Korrektheit der Konstruktion

Lemma Korrektheit Berechnungstabelle

Sei T eine Tabelle mit den folgenden Eigenschaften.

1. Die erste Zeile ist die Startkonfiguration von N auf w.
2. Jedes Fenster ist legal.

Dann ist T eine Berechnungstabelle von N auf Eingabe w.

Beweis:

- $T(i,j) \neq T(i+1,j)$ ist nur dann möglich, falls einer der Einträge $T(i,j-1), T(i,j)$ oder $T(i,j+1)$ einen Zustand enthält.
- Falls die obere Zeile einen Zustand ändert, muss sich die untere Zeile gemäß δ ändern.
- D.h. jede Zeile ist eine Nachfolgekonfiguration der Vorgängerzeile.
- Damit ist T eine Berechnungstabelle.

Konstruktion von ϕ_{move}

- Informal gilt: $\phi_{move} = \bigwedge_{1 \leq i \leq n^k, 2 \leq j \leq n^k}$ Fenster (i,j) ist legal.
- Die Anzahl legaler Fenster hängt nur von den möglichen Übergängen in N ab, nicht von der Eingabe w.
- D.h. es gibt eine Menge F von 6-Tupeln $(f_1, \ldots, f_6)$, so dass F alle legalen Fenster beschreibt.
- Damit können wir das Prädikt [Fenster (i,j) ist legal] formalisieren

$$\bigvee_{(f_1,\ldots,f_6)\in F} (x_{i,j-1,f_1} \wedge x_{i,j,f_2} \wedge x_{i,j+1,f_3} \wedge x_{i+1,j-1,f_4} \wedge x_{i+1,j,f_5} \wedge x_{i+1,j+1,f_6}).$$

Reduktion ist polynomiell

Lemma Länge von ϕ

Sei N eine NTM mit Laufzeit n^k bei Eingabe w, $|w| = n$. Dann besitzt die Formel $\phi = \phi_{Start} \wedge \phi_{accept} \wedge \phi_{Eintrag} \wedge \phi_{move}$ Länge $\mathcal{O}(n^{2k})$, d.h. ihre Länge ist polynomiell in n.
Zudem ist ϕ bei Eingabe (N, w) in Zeit $\mathcal{O}(n^{2k})$ berechenbar.

ϕ_{Start}:
- Anzahl Literale: $\mathcal{O}(n^k)$, Berechnung direkt aus w

ϕ_{accept}:
- Anzahl Literale: $\mathcal{O}(n^{2k})$

$\phi_{Eintrag}$:
- Anzahl Literale in $\phi_{\geq 1}, \phi_{\leq 1}$: $\mathcal{O}(1)$, unabhängig von w.
- Anzahl Literale in $\phi_{Eintrag}$: $\mathcal{O}(n^{2k})$.

ϕ_{move}:
- Anzahl legaler Fenster $|F|$: $\mathcal{O}(1)$, unabhängig von w.
- Anzahl Literale in ϕ_{move}: $\mathcal{O}(n^{2k})$.

Von SAT zu 3SAT

Satz

3SAT ist $\mathcal{NP}$-vollständig.

- Modifizieren zunächst vorigen Beweis derart, dass ϕ in KNF ist.
- ϕ_{start} und ϕ_{accept} sind bereits in KNF.
- $\phi_{Eintrag} = \bigwedge_{i,j}(\phi_{\geq 1} \wedge \phi_{\leq 1}) = \bigwedge_{i,j} \phi_{\geq 1} \wedge \bigwedge_{i,j} \phi_{\leq 1}$
  - $\phi_{\geq 1}$ besteht aus einer Klausel.
  - Schreiben $\phi_{\leq 1}$ als Konjunktion von Klauseln:

$$\phi_{\leq 1} = \bigwedge_{s \neq t} \left(\neg x_{i,j,s} \vee \neg x_{i,j,t} \right) .$$

- ϕ_{move}: Wandle disjunktive Normalform des Prädikats für legale Fenster

$$\bigvee_{(f_1,\ldots,f_6) \in F} (x_{i,j-1,f_1} \wedge x_{i,j,f_2} \wedge \ldots \wedge x_{i+1,j+1,f_6}).$$

in KNF um. Umwandlung in $\mathcal{O}(1)$, da $|F|$ unabhängig von $|w| = n$.

Umwandlung von KNF in 3-KNF

Sei $\phi = k_1 \wedge \ldots \wedge k_m$ eine KNF-Formel, wobei $k_j = a_1 \vee \ldots \vee a_n$ eine Klausel mit $n > 3$ Literalen ist.

- Führen neue Variablen $z_1, \ldots, z_{n-3}$ ein.
- Ersetzen Klausel k_j durch die 3-KNF Formel

$$k_j' = (a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee z_3) \wedge \ldots \wedge (\neg z_{n-3} \vee a_{n-1} \vee a_n)$$

- zu zeigen: k_j ist erfüllbar gdw. k_j' erfüllbar ist.
- B ist eine erfüllende Belegung für k_j gdw ein Literal a_i wahr ist.
- Dann ist aber k_j' erfüllbar mit $a_i = 1$ und
$$z_j = 1 \text{ für } j < i - 1 \quad \text{und} \quad z_j = 0 \text{ für } j \geq i - 1.$$

- Sei andererseits k_j' erfüllbar.
- Dann muss ein Literal a_i wahr sein, und damit ist k erfüllbar.

- Können ϕ in KNF bzw. in 3-KNF in $\mathcal{O}(|\phi|)$ Schritten umwandeln.

$\mathcal{NP}$-Vollständigkeit von CLIQUE

Satz

CLIQUE ist $\mathcal{NP}$-vollständig.

Beweis: zu zeigen

1. CLIQUE $\in \mathcal{NP}$
   - Übung
2. $\exists$ $\mathcal{NP}$-vollständige Sprache L mit $L \leq_p$ CLIQUE
   - Bereits gezeigt: 3SAT ist $\mathcal{NP}$-vollständig.
   - Bereits gezeigt: 3SAT $\leq_p$ CLIQUE.

Knotenüberdeckung

Definition k-Knotenüberdeckung

Sei $G = (V, E)$ ein ungerichteter Graph. Eine Knotenmenge $U \subseteq V$, $|U| = k$ heißt *k-Knotenüberdeckung*, falls

$$e \cap U \neq \emptyset \text{ für alle } e \in E.$$

Wir definieren die folgende Sprache.

$$\textsc{Knotenüberdeckung} := \{(G, k) \mid G \text{ besitzt eine } k\text{-Knotenüberdeckung.}\}$$

Satz

KNOTENÜBERDECKUNG ist $\mathcal{NP}$-vollständig.

Beweis: zu zeigen

1. KNOTENÜBERDECKUNG $\in \mathcal{NP}$ (Übung)
2. 3-SAT $\leq_p$ KNOTENÜBERDECKUNG, d.h. es gibt berechenbares f:

$$\phi \in \text{3SAT} \Leftrightarrow f(\phi) = (G, k) \in \textsc{Knotenüberdeckung}$$

Die Reduktion f

Idee der Reduktion f:

- Konstruieren für jedes Literal x_i Knotenpaar mit Labeln x_i und $\neg x_i$.
- Knotenlabel einer Überdeckung bilden erfüllende Belegung.

Algorithmus M_f

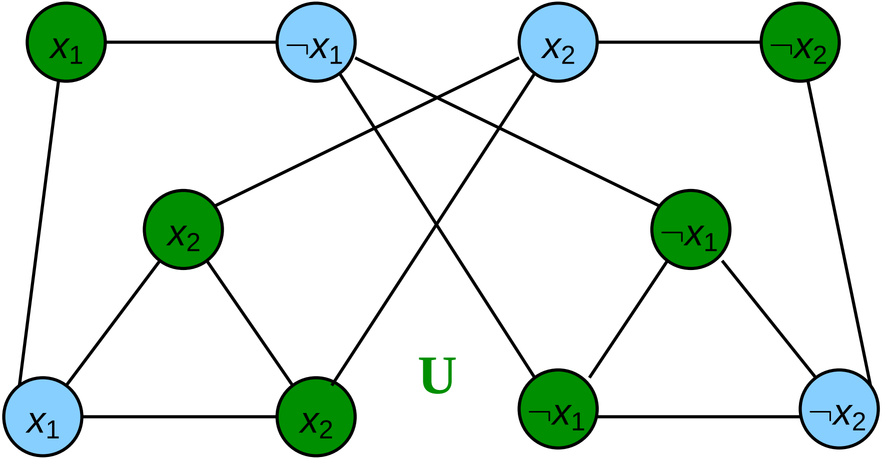
Eingabe: $\phi(x_1, \dots, x_n) = K_1 \wedge \dots \wedge K_m$ mit $K_j = \ell_{j1} \vee \ell_{j2} \vee \ell_{j3}$.

1. Variablenknoten: Für $i = 1 \dots n$:
   - Konstruiere zwei verbundene Knoten mit Labeln x_i und $\neg x_i$.
2. Klauselknoten: Für $j = 1 \dots m$:
   - Konstruiere 3 paarweise verbundene Knoten mit Labeln $\ell_{j1}, \ell_{j2}, \ell_{j3}$.
3. Verbinde Variablen- und Klauselknoten mit denselben Labeln.
4. Setze $k = n + 2m$.

Ausgabe: (G, k)

- Schritt 1: $\mathcal{O}(n)$, Schritt 2: $\mathcal{O}(m)$, Schritt 3: $\mathcal{O}(m)$, Schritt 4: $\mathcal{O}(1)$.
- $|\phi| = \mathcal{O}(n + m) = \mathcal{O}(m)$, d.h. die Laufzeit ist polynomiell in $|\phi|$.

Reduktion für $\phi = (x_1 \vee x_2 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_2)$



$\phi \in$ 3SAT $\Rightarrow f(\phi) \in$ KNOTENÜBERDECKUNG

Sei $\phi(x_1, \ldots, x_n) \in$ 3SAT

- Dann gibt es eine erfüllende Belegung der Variablen $x_1, \ldots, x_n$.
- In die Menge U werden die folgenden Knoten aufgenommen.
  - n Variablenknoten:
    Falls $x_i = 1$, ist Knoten mit Label x_i in U. Sonst Knoten mit $\neg x_i$.
  - $2m$ Klauselknoten:
    Für jede Klausel ist mindestens ein Knoten mit einem Variablenknoten aus U verbunden. Die *anderen beiden Knoten* sind in U.
- U ist eine $n + 2m$-Knotenüberdeckung:
  - Die Kanten zwischen Variablenknoten $x_i, \neg x_i$ sind überdeckt durch einen Variablenknoten.
  - Kanten zwischen Klauselknoten $\ell_{j1}, \ell_{j2}, \ell_{j3}$ sind überdeckt durch zwei Klauselknoten.
  - Kanten zwischen Variablen- und Klauselknoten sind überdeckt: Entweder der Variablenknoten überdeckt die Kante oder einer der beiden Klauselknoten.
- D.h. $f(\phi) = (G, n + 2m) \in$ KNOTENÜBERDECKUNG

Korrektheit: Rückrichtung

Sei $f(\phi) = (G, n + 2m) \in$ KNOTENÜBERDECKUNG:

- Dann gibt es eine $(n + 2m)$-Knotenüberdeckung U mit:
  - Mindestens ein Variablenknoten x_i oder $\neg x_i$ ist in U für alle i.
  - Mindestens 2 von 3 Klauselknoten $\ell_{j1}, \ell_{j2}, \ell_{j3}$ sind in U für alle j.
  - Da $|U| = n + 2m$:
    Jeweils *genau ein* Variablenknoten und *genau zwei* Klauselknoten.
- Sei B die Belegung, die die Variablenknoten aus U auf wahr setzt.
  - B ist eine konsistente Belegung.
  - Für alle Klauseln K_j mit Knoten $\ell_{j1}, \ell_{j2}, \ell_{j3}$ ist ein $\ell_{jk}, k \in [3]$ nicht in U.
  - Die Kante vom Klausel- zum Variablenknoten mit demselben Label ℓ_{jk} wird überdeckt.
  - D.h. der Variablenknoten ℓ_{jk} ist in U. Damit erfüllt ℓ_{jk} die Klausel K_j.
- D.h. B ist eine erfüllende Belegung für ϕ.
- Damit gilt $\phi \in$ 3SAT.

Subset Sum

Definition Sprache SubsetSum

Sei $M = \{m_1, \dots, m_n\} \subset \mathbb{N}$ und $t \in \mathbb{N}$. Wir definieren die Sprache

$$\textsc{SubsetSum} := \{(M, t) \mid \exists S \subseteq M : \sum_{s \in S} s = t\}.$$

Satz

$\textsc{SubsetSum}$ ist $\mathcal{NP}$-vollständig.

1. $\textsc{SubsetSum} \in \mathcal{NP}$ (Übung)
2. 3SAT $\leq_p$ $\textsc{SubsetSum}$

Idee der Reduktion $f(\phi(x_1, \dots, x_n)) = (S, t)$: Konstruieren

- für jedes x_i Elemente $y_i, z_i \in S$ für $x_i = 1$ bzw. $x_i = 0$,
- für jede Klausel K_j Variablen $g_j, h_j \in S$ für nicht erfüllte Literale.
- Definieren Tabelle T mit Zeilen y_i, z_i, g_j, h_j und Zeile t. Die Spalten bestehen aus x_i und K_j für $i \in [n], j \in [m]$.
- Einträge in einer Zeile werden als Dezimaldarstellung interpretiert.

Konstruktion der Reduktion f

Algorithmus M_f

EINGABE: $\phi(x_1, \ldots, x_n) = K_1 \wedge \ldots \wedge K_m$ mit $K_j = \ell_{j1} \vee \ell_{j2} \vee \ell_{j3}$

1. Erstelle Tabelle T mit Spalten für $x_1, \ldots, x_n$ und $K_1, \ldots, K_m$.
2. Erstelle $2n$ Variablenzeilen für x_i, $i = 1, \ldots, n$:
   - y_i: Einsen in Spalte x_i. Für alle Spalten K_j: Anzahl Literale x_i in K_j.
   - z_i: Einsen in Spalte x_i. Für alle Spalten K_j: Anzahl Literale $\neg x_i$ in K_j.
3. Erstelle $2m$ Klauselzeilen für K_j, $j = 1, \ldots, m$:
   - g_j, h_j: Einsen jeweils in Spalte K_j.
4. Erstelle Zeile t: Einsen in Spalten x_i, Dreien in Spalten K_j.
5. Fülle mit Nullen. Definiere $y_1, z_1, \ldots, y_n, z_n$, $g_1, h_1, \ldots, g_m, h_m, t$ mittels des Dezimalwerts der betreffenden Zeile.

AUSGABE: (M, t) mit $M = \{y_1, z_1, \ldots, y_n, z_n, g_1, h_1, \ldots, g_m, h_m\}$.

Laufzeit:

- Eingabelänge $|\phi| \geq \max\{m, n\} = \Omega(m + n)$
- $T(M_f) = \mathcal{O}((n + m)^2)$, d.h. polynomiell in der Eingabelänge.

Bsp für $\phi = (x_1 \vee x_2 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_2)$

- Definieren Tabelle T

	x_1	x_2	K_1	K_2
y_1	1	0	1	0
z_1	1	0	0	1
y_2	0	1	2	1
z_2	0	1	0	1
g_1	0	0	1	0
h_1	0	0	1	0
g_2	0	0	0	1
h_2	0	0	0	1
t	1	1	3	3

- Setze $y_1 = 1010, z_1 = 1001, \ldots, t = 1133$.
- Belegung $x_1, x_2 = 1$ erfüllt alle Literale in K_1 und Literal x_2 in K_2.
- Zahlen y_1, y_2 summieren sich mit g_2, h_2 für K_2 zu t.

Korrektheit: $\phi \in$ 3SAT $\Rightarrow f(\phi) \in$ SUBSETSUM

Sei $\phi \in$ 3SAT

- Dann besitzt ϕ eine erfüllende Belegung B.
- Nimm y_i in S auf, falls $x_i = 1$ in B. Sonst nimm z_i in S auf.
- Betrachten $t' = \sum_{s \in S} s$:
  - B ist konsistente Belegung: Obere n Dezimalstellen von t' sind 1.
  - B ist erfüllend: Untere m Dezimalstellen $t_1, \ldots, t_m$ sind aus $\{1, 2, 3\}$.
- Falls $t_j = 1$, nimm g_j und h_j in S auf. Falls $t_j = 2$, nimm g_j in S auf.
- Damit gilt $\sum_{s \in S} s = t$.
- D.h. $f(\phi) = (M, t) \in$ SUBSETSUM

Korrektheit $f(\phi) \in$ SUBSETSUM $\Rightarrow \phi \in$ 3SAT

Sei $f(\phi) \in$ SUBSETSUM

- Dann gibt es $S \subseteq M$ mit $\sum_{s \in S} s = t$, wobei $t = 1 \ldots 13 \ldots 3$.
- Die oberen n Dezimalstellen von t sind 1.
  - Damit enthält S für jedes i genau eines der Elemente y_i, z_i.
  - Sei B die Belegung mit $x_1 = 1$ für $y_i \in S$ und $x_1 = 0$ für $z_i \in S$.
- Die unteren m Dezimalstellen $t_1, \ldots, t_m$ von t sind 3.
  - D.h. t_j kann nicht allein als Summe von g_j und h_j dargestellt werden.
  - Für jedes t_j kommt mindestens ein Beitrag aus einer der Zeilen y_i bzw. z_i.
  - D.h. das Literal x_i bzw. $\neg x_i$ erfüllt die Klausel K_j.
- Damit ist B eine erfüllende Belegung für ϕ.
- D.h. $\phi \in$ 3SAT.