

Script generated by TTT

Title: seidl: Theoretische_Informatik (09.07.2012)

Date: Mon Jul 09 10:19:58 CEST 2012

Duration: 88:52 min

Pages: 79

Weitere Anwendungen von SAT:

Bounded Model Checking

Weitere Anwendungen von SAT:

Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für **alle** Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

Weitere Anwendungen von SAT:

Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für **alle** Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

Software Entscheide, ob ein Programm für **alle** Eingaben innerhalb von 10 Schritten ein bestimmtes Verhalten (nicht) hat.
Variablen müssen auf sehr kleine Wertebereiche eingeschränkt werden!

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in \text{NP}$ (meist trivial)

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in \text{NP}$ (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A .

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in \text{NP}$ (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A .

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in \text{NP}$ (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A .

Lemma 5.23

Ist A NP-vollständig, so ist $B \in \text{NP}$ ebenfalls NP-vollständig, falls $A \leq_p B$.

5.4 Weitere NP-vollständige Probleme

Wie zeigt man, dass ein (weiteres) Problem B NP-vollständig ist?

- Zeige $B \in \text{NP}$ (meist trivial)
- Zeige $A \leq_p B$ für ein NP-vollständiges Problem A .

Lemma 5.23

Ist A NP-vollständig, so ist $B \in \text{NP}$ ebenfalls NP-vollständig, falls $A \leq_p B$.

Beweis:

Folgt direkt aus der Transitivität von \leq_p :

B ist NP-hart, denn für $C \in \text{NP}$ gilt $C \leq_p A \leq_p B$. □

Warum will man wissen, dass ein Problem B NP-vollständig ist?

Um sicher zu sein, dass

- ein polynomieller Algorithmus für B ein Durchbruch wäre
- und daher wahrscheinlich nicht existiert.

Warum will man wissen, dass ein Problem B NP-vollständig ist?

Um sicher zu sein, dass

- ein polynomieller Algorithmus für B ein Durchbruch wäre

Warum will man wissen, dass ein Problem B NP-vollständig ist?

Um sicher zu sein, dass

- ein polynomieller Algorithmus für B ein Durchbruch wäre
- und daher wahrscheinlich nicht existiert.

Praktische Instanzen von B könnten trotzdem (zB mit SAT) „effizient“ lösbar sein.

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$
- Ein **Literal** ist eine (evtl. negierte) Variable.

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$
- Ein **Literal** ist eine (evtl. negierte) Variable.
- Eine Formel ist in **3KNF** gdw jede Klausel ≤ 3 Literale enthält.

$$x_1 \vee \neg x_2 \vee x_3$$

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$
- Ein **Literal** ist eine (evtl. negierte) Variable.
- Eine Formel ist in **3KNF** gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.

Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$
- Ein **Literal** ist eine (evtl. negierte) Variable.
- Eine Formel ist in **3KNF** gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.

Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

3KNF-SAT

Gegeben: Ein Formel in 3KNF

Problem: Ist die Formel erfüllbar?

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$
- Ein **Literal** ist eine (evtl. negierte) Variable.
- Eine Formel ist in **3KNF** gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.

Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

3KNF-SAT

Gegeben: Ein Formel in 3KNF

Definition 5.24

- Eine Formel ist in **Konjunktiver Normalform (KNF)** gdw sie eine Konjunktion von **Klauseln** ist: $K_1 \wedge \dots \wedge K_n$
- Eine **Klausel** ist eine Disjunktion von **Literalen**: $L_1 \vee \dots \vee L_m$
- Ein **Literal** ist eine (evtl. negierte) Variable.
- Eine Formel ist in **3KNF** gdw jede Klausel ≤ 3 Literale enthält.

Dh eine KNF ist ein Konjunktion von Disjunktionen von (evtl negierten) Variablen.

Bsp: $(x_9 \vee \neg x_2) \wedge (\neg x_7 \vee x_1 \vee x_6)$

3KNF-SAT

Gegeben: Ein Formel in 3KNF

Problem: Ist die Formel erfüllbar?

Offensichtlich gilt $3KNF-SAT \in NP$.

Satz 5.25

3KNF-SAT ist NP-vollständig.

Satz 5.25

3KNF-SAT ist NP-vollständig.

Beweis:

Wir zeigen $\text{SAT} \leq_p \text{3KNF-SAT}$ mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar} \Leftrightarrow F' \text{ ist erfüllbar}$$

NB F und F' sind **erfüllbarkeitsäquivalent**, aber nicht notwendigerweise auch äquivalent.

Satz 5.25

3KNF-SAT ist NP-vollständig.

Beweis:

Wir zeigen $\text{SAT} \leq_p \text{3KNF-SAT}$

Satz 5.25

3KNF-SAT ist NP-vollständig.

Beweis:

Wir zeigen $\text{SAT} \leq_p \text{3KNF-SAT}$ mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar} \Leftrightarrow F' \text{ ist erfüllbar}$$

NB F und F' sind **erfüllbarkeitsäquivalent**, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in **Negations-Normalform (NNF)** durch fortgesetzte Anwendung der de Morganschen Gesetze

Satz 5.25

3KNF-SAT ist NP-vollständig.

Beweis:

Wir zeigen $\text{SAT} \leq_p \text{3KNF-SAT}$ mit einer polynomiellen Reduktion $F \mapsto F'$ so dass F' in 3KNF ist und

$$F \text{ ist erfüllbar} \Leftrightarrow F' \text{ ist erfüllbar}$$

NB F und F' sind erfüllbarkeitsäquivalent, aber nicht notwendigerweise auch äquivalent.

1. Transformiere F in Negations-Normalform (NNF) durch fortgesetzte Anwendung der de Morganschen Gesetze

$$\neg(A \wedge B) = \neg A \vee \neg B$$

$$\neg(A \vee B) = \neg A \wedge \neg B$$

$$\neg\neg A = A$$

von links nach rechts

Beweis (Forts.):

Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Beweis (Forts.):

Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Beweis (Forts.):

Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen

$$y_i = \begin{array}{c} \circ_i \\ / \quad \backslash \\ l_i \quad r_i \end{array}$$

Beweis (Forts.):

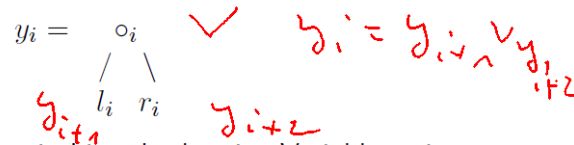
Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen



wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.

Beweis (Forts.):

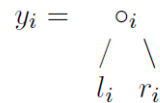
Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen



wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.

Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0)) \wedge (y_1 \leftrightarrow (l_1 \circ_1 r_1))$$

Beweis (Forts.):

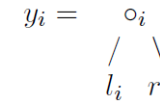
Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen



wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.

Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0))$$

Beweis (Forts.):

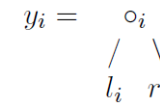
Für F_1 gilt: \neg nur noch direkt vor Variablen.

2. Betrachte F_1 als Baum, wobei die Literale Blätter sind.

Ordne jedem inneren Knoten eine neue Variable $\in \{y_0, y_1, \dots\}$ zu.

Ordne dabei der Wurzel von F_1 die Variable y_0 zu.

3. Betrachte die y_i als Abkürzung für die Teilbäume, an deren Wurzeln sie stehen



wobei $\circ_i \in \{\wedge, \vee\}$ und l_i, r_i ein Literal oder eine Variable y_j ist.

Beschreibe F_1 Knoten für Knoten:

$$y_0 \wedge (y_0 \leftrightarrow (l_0 \circ_0 r_0)) \wedge (y_1 \leftrightarrow (l_1 \circ_1 r_1)) \dots =: F_2$$

Beweis (Forts.):

F_1 erf. $\implies F_2$ erf.: y_i bekommt Wert seines Teilbaums.

F_2 erf. $\implies F_1$ erf.: klar

4. Transformiere jede Äquivalenz in 3KNF:

$$(a \leftrightarrow (b \vee c)) \mapsto (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c)$$

$$(a \leftrightarrow (b \wedge c)) \mapsto (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

$$\begin{aligned} & a \vee \neg(b \wedge c) \\ \equiv & a \vee \neg b \vee \neg c \end{aligned}$$

Beweis (Forts.):

F_1 erf. $\implies F_2$ erf.: y_i bekommt Wert seines Teilbaums.

F_2 erf. $\implies F_1$ erf.: klar

4. Transformiere jede Äquivalenz in 3KNF:

$$(a \leftrightarrow (b \vee c)) \mapsto (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c)$$

$$(a \leftrightarrow (b \wedge c)) \mapsto (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Ergebnis ist F' .

Jeder Schritt ist in polynomieller Zeit in $|F|$ berechenbar.

Bei Transformation in NNF nimmt mit jedem Schritt

Summe der $|G|$ für alle Teilformeln $\neg G$ von F

ab.

Beweis (Forts.):

F_1 erf. $\implies F_2$ erf.: y_i bekommt Wert seines Teilbaums.

F_2 erf. $\implies F_1$ erf.: klar

4. Transformiere jede Äquivalenz in 3KNF:

$$(a \leftrightarrow (b \vee c)) \mapsto (a \vee \neg b) \wedge (a \vee \neg c) \wedge (\neg a \vee b \vee c)$$

$$(a \leftrightarrow (b \wedge c)) \mapsto (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Ergebnis ist F' .

Jeder Schritt ist in polynomieller Zeit in $|F|$ berechenbar.

Da jede Formel in 3KNF auch in KNF ist:

Korollar 5.26

KNF-SAT ist NP-vollständig.

Da jede Formel in 3KNF auch in KNF ist:

[Korollar 5.26](#)

KNF-SAT ist NP-vollständig.

Kann man wie folgt die NP-Vollständigkeit von KNF-SAT zeigen?



MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Da jede Formel in 3KNF auch in KNF ist:

[Korollar 5.26](#)

KNF-SAT ist NP-vollständig.

Kann man wie folgt die NP-Vollständigkeit von KNF-SAT zeigen?

Man zeigt $\text{SAT} \leq_p \text{KNF-SAT}$
indem man jede Formel in KNF transformiert.

[Satz 5.27](#)

$2\text{KNF-SAT} \in P$

Ohne Beweis

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{aligned} T_1 &= \{1, 2\} & T_2 &= \{1, 3\} \\ T_3 &= \{3, 4\} & T_4 &= \{3, 5\} \\ M &= \{1, 2, 3, 4, 5\} & k &= 3 \end{aligned}$$

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{aligned} T_1 &= \{1, 2\} & T_2 &= \{1, 3\} \\ T_3 &= \{3, 4\} & T_4 &= \{3, 5\} \\ M &= \{1, 2, 3, 4, 5\} & k &= 3 \end{aligned}$$

Lösung:

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{aligned} T_1 &= \{1, 2\} & T_2 &= \{1, 3\} \\ T_3 &= \{3, 4\} & T_4 &= \{3, 5\} \\ M &= \{1, 2, 3, 4, 5\} & k &= 3 \end{aligned}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{aligned} T_1 &= \{1, 2\} & T_2 &= \{1, 3\} \\ T_3 &= \{3, 4\} & T_4 &= \{3, 5\} \\ M &= \{1, 2, 3, 4, 5\} & k &= 3 \end{aligned}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{ll} T_1 = \{1, 2\} & T_2 = \{1, 3\} \\ T_3 = \{3, 4\} & T_4 = \{3, 5\} \\ M = \{1, 2, 3, 4, 5\} & k = 3 \end{array}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

T_i Menge der Teile, die Zulieferer i anbietet

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{ll} T_1 = \{1, 2\} & T_2 = \{1, 3\} \\ T_3 = \{3, 4\} & T_4 = \{3, 5\} \\ M = \{1, 2, 3, 4, 5\} & k = 3 \end{array}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

T_i Menge der Teile, die Zulieferer i anbietet

Kann die Firma ihre Bedürfnisse mit k Zulieferern abdecken?

Fakt 5.29

$MÜ \in NP$.

MENGENÜBERDECKUNG (MÜ)

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M und eine Zahl $k \leq n$.

Problem: Gibt es $i_1, \dots, i_k \in \{1, \dots, n\}$ mit $M = T_{i_1} \cup \dots \cup T_{i_k}$?

Beispiel 5.28

$$\begin{array}{ll} T_1 = \{1, 2\} & T_2 = \{1, 3\} \\ T_3 = \{3, 4\} & T_4 = \{3, 5\} \\ M = \{1, 2, 3, 4, 5\} & k = 3 \end{array}$$

Lösung:

Anwendung: Zulieferer

M Menge der Teile, die eine Firma einkaufen muss

T_i Menge der Teile, die Zulieferer i anbietet

Kann die Firma ihre Bedürfnisse mit k Zulieferern abdecken?

Satz 5.30

$MÜ$ ist NP-vollständig.

Beweis:

Wir zeigen $\text{KNF-SAT} \leq_p \text{MÜ}$.

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen $\text{KNF-SAT} \leq_p \text{MÜ}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ in KNF, mit Variablen x_1, \dots, x_k .

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen $\text{KNF-SAT} \leq_p \text{MÜ}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ in KNF, mit Variablen x_1, \dots, x_k .

$$M := \{1, \dots, m, m+1, \dots, m+k\}$$

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen $\text{KNF-SAT} \leq_p \text{MÜ}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ in KNF, mit Variablen x_1, \dots, x_k .

$$M := \{1, \dots, m, m+1, \dots, m+k\}$$

$$T_i := \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\}$$

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen $\text{KNF-SAT} \leq_p \text{MÜ}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ in KNF, mit Variablen x_1, \dots, x_k .

$$M := \{1, \dots, m, m+1, \dots, m+k\}$$

$$T_i := \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\}$$

$$T'_i := \{j \mid x_i \text{ kommt in } K_j \text{ negativ vor}\} \cup \{m+i\}$$

Satz 5.30

MÜ ist NP-vollständig.

Beweis:

Wir zeigen $\text{KNF-SAT} \leq_p \text{MÜ}$.

Sei $F = K_1 \wedge \dots \wedge K_m$ in KNF, mit Variablen x_1, \dots, x_k .

$$M := \{1, \dots, m, m+1, \dots, m+k\}$$

$$T_i := \{j \mid x_i \text{ kommt in } K_j \text{ positiv vor}\} \cup \{m+i\}$$

$$T'_i := \{j \mid x_i \text{ kommt in } K_j \text{ negativ vor}\} \cup \{m+i\}$$

F ist erfüllbar gdw

M wird durch k der Teilmengen $T_1, \dots, T_k, T'_1, \dots, T'_k$ überdeckt

□

Das Minimierungsproblem

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k , so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

Finde kleinstes k durch binäre Suche im Intervall $[1, n]$
mit $O(\log n)$ Aufrufen von MÜ.

Das Minimierungsproblem

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k , so dass M von k der Teilmengen überdeckt wird.

Das Minimierungsproblem

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k , so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

Finde kleinstes k durch binäre Suche im Intervall $[1, n]$
mit $O(\log n)$ Aufrufen von MÜ.

Kann man MÜ in Zeit $O(f(n))$ entscheiden,
dann kann man das kleinste k in Zeit $O(f(n) \cdot \log n)$ berechnen.

Das Minimierungsproblem

Gegeben: Teilmengen $T_1, \dots, T_n \subseteq M$ einer endlichen Menge M

Problem: Finde das kleinste k , so dass M von k der Teilmengen überdeckt wird.

kann auf das Entscheidungsproblem reduziert werden:

Finde kleinstes k durch binäre Suche im Intervall $[1, n]$
mit $O(\log n)$ Aufrufen von MÜ.

Kann man MÜ in Zeit $O(f(n))$ entscheiden,
dann kann man das kleinste k in Zeit $O(f(n) \cdot \log n)$ berechnen.

$f(n)$ polynomiell $\implies f(n) \cdot \log n$ polynomiell
 $f(n)$ exponentiell $\implies f(n) \cdot \log n$ exponentiell

Die Berechnung einer Lösung

Gegeben: Teilmengen $\vec{T} := T_1, \dots, T_n \subseteq M$ einer endlichen Menge M , und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

```
if  $(\vec{T}, M, k) \notin \text{MÜ}$  then output("nicht lösbar")
else
   $\vec{U} := \emptyset$ 
```

Die Berechnung einer Lösung

Gegeben: Teilmengen $\vec{T} := T_1, \dots, T_n \subseteq M$ einer endlichen Menge M , und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

Die Berechnung einer Lösung

Gegeben: Teilmengen $\vec{T} := T_1, \dots, T_n \subseteq M$ einer endlichen Menge M , und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

```
if  $(\vec{T}, M, k) \notin \text{MÜ}$  then output("nicht lösbar")
else
   $\vec{U} := \emptyset$ 
  for  $i := 1$  to  $n$  do
    if  $(\vec{T} - T_i, M, k) \in \text{MÜ}$ 
```

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \dots, T_n \subseteq M$ einer endlichen Menge M , und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin \text{MÜ}$ **then** output("nicht lösbar")

else

$\vec{U} := \emptyset$

for $i := 1$ **to** n **do**

if $(\vec{T} - T_i, M, k) \in \text{MÜ}$

then $\vec{T} := \vec{T} - T_i$

Die Berechnung einer **Lösung**

Gegeben: Teilmengen $\vec{T} := T_1, \dots, T_n \subseteq M$ einer endlichen Menge M , und eine Zahl $k \leq n$.

Problem: Finde eine Überdeckung von M durch k der Teilmengen.

kann auf das Entscheidungsproblem reduziert werden:

if $(\vec{T}, M, k) \notin \text{MÜ}$ **then** output("nicht lösbar")

else

$\vec{U} := \emptyset$

for $i := 1$ **to** n **do**

if $(\vec{T} - T_i, M, k) \in \text{MÜ}$

then $\vec{T} := \vec{T} - T_i$

else $\vec{U} := \vec{U} \cup \{T_i\}$

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k ?

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k ?

Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$

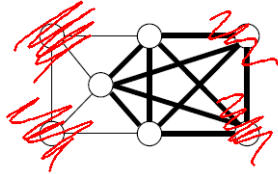
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k ?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:

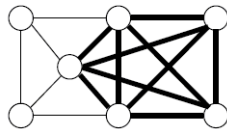


CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k ?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:



Anwendung von Clique-Berechnung:

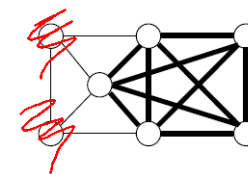
Knoten = Prozess
Kante = Zwei Prozesse sind parallel ausführbar

CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k ?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:

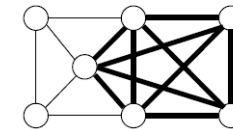


CLIQUE

Gegeben: Ungerichteter Graph $G = (V, E)$ und Zahl $k \in \mathbb{N}$.

Problem: Besitzt G eine „Clique“ der Größe mindestens k ?
Dh eine Teilmenge $V' \subseteq V$ mit $|V'| \geq k$
und alle $u, v \in V'$ mit $u \neq v$ sind benachbart.

Beispiel mit 5-Clique:



Anwendung von Clique-Berechnung:

Knoten = Prozess
Kante = Zwei Prozesse sind parallel ausführbar

\implies Clique ist Gruppe von parallelisierbaren Prozessen

Fakt 5.31

CLIQUE \in NP

Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1, 1), (1, 2), (1, 3), \dots, (k, 1), (k, 2), (k, 3)\}$$

Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

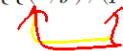
Eine Formel F in 3KNF-SAT (oE: *genau* 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1, 1), (1, 2), (1, 3), \dots, (k, 1), (k, 2), (k, 3)\}$$

$$E = \{(i, j), (p, q) \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\}$$



Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

Eine Formel F in 3KNF-SAT (oE: genau 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1, 1), (1, 2), (1, 3), \dots, (k, 1), (k, 2), (k, 3)\}$$

$$E = \{(i, j), (p, q) \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\}$$

F ist erfüllbar durch eine Belegung σ

\iff

Es gibt in jeder Klausel i ein Literal z_{ij} mit $\sigma(z_{ij}) = 1$

\iff

Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

Eine Formel F in 3KNF-SAT (oE: genau 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1, 1), (1, 2), (1, 3), \dots, (k, 1), (k, 2), (k, 3)\}$$

$$E = \{(i, j), (p, q) \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\}$$

F ist erfüllbar durch eine Belegung σ

\iff

Es gibt in jeder Klausel i ein Literal z_{ij} mit $\sigma(z_{ij}) = 1$

\iff

Die Literale $z_{1j_1}, \dots, z_{kj_k}$ sind paarweise nicht komplementär

\iff

Die Knoten $(1, j_1), \dots, (k, j_k)$ sind paarweise benachbart

$\iff G$ hat eine Clique der Größe k . □

Satz 5.32

CLIQUE ist NP-vollständig.

Beweis: 3KNF-SAT \leq_p CLIQUE:

Eine Formel F in 3KNF-SAT (oE: genau 3 Literale/Klausel)

$$F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{k1} \vee z_{k2} \vee z_{k3})$$

wird auf einen Graphen abgebildet:

$$V = \{(1, 1), (1, 2), (1, 3), \dots, (k, 1), (k, 2), (k, 3)\}$$

$$E = \{(i, j), (p, q) \mid i \neq p \text{ und } z_{ij} \neq \neg z_{pq}\}$$

F ist erfüllbar durch eine Belegung σ

\iff

Es gibt in jeder Klausel i ein Literal z_{ij} mit $\sigma(z_{ij}) = 1$

\iff

Die Literale $z_{1j_1}, \dots, z_{kj_k}$ sind paarweise nicht komplementär

\iff

RUCKSACK

Gegeben: Zahlen $a_1, \dots, a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

RUCKSACK

Gegeben: Zahlen $a_1, \dots, a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \dots, n\}$ mit $\sum_{i \in R} a_i = b$?

RUCKSACK

Gegeben: Zahlen $a_1, \dots, a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \dots, n\}$ mit $\sum_{i \in R} a_i = b$?

Satz 5.33

RUCKSACK ist NP-vollständig.

Beweis:

3KNF-SAT \leq_p RUCKSACK :

RUCKSACK

Gegeben: Zahlen $a_1, \dots, a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \dots, n\}$ mit $\sum_{i \in R} a_i = b$?

Satz 5.33

RUCKSACK ist NP-vollständig.

RUCKSACK

Gegeben: Zahlen $a_1, \dots, a_n \in \mathbb{N}$ und $b \in \mathbb{N}$.

Problem: Gibt es $R \subseteq \{1, \dots, n\}$ mit $\sum_{i \in R} a_i = b$?

Satz 5.33

RUCKSACK ist NP-vollständig.

Beweis:

3KNF-SAT \leq_p RUCKSACK :

Sei $F = (z_{11} \vee z_{12} \vee z_{13}) \wedge \dots \wedge (z_{m1} \vee z_{m2} \vee z_{m3})$, wobei

$$z_{ij} \in \{x_1, \dots, x_n\} \cup \{\neg x_1, \neg x_2, \dots, \neg x_n\}$$

D.h. m = Anzahl der Klauseln und n Anzahl der vorkommenden Variablen.

Wir geben jetzt Zahlen $\vec{a} = a_1, \dots, a_k$ und b an. b ist (etwa im Dezimalsystem)

$$b = \underbrace{44 \dots 44}_m \underbrace{11 \dots 11}_n$$

Handwritten notes in red:
 $\begin{matrix} 9 & \dots & 2 & 4 \\ 2 & 8 & \dots & 9 & 5 \\ 1 & 0 & \dots & 0 & \\ 2 & 0 & \dots & 0 & \end{matrix}$

□