SS 2005

Einführung in die Informatik IV

Ernst W. Mayr

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2005SS/info4/index.html.de

15. Juli 2005

Lemma 219 (blaue Regel)

Sei $G = (V, E)$, $(W, V \setminus W)$ ein Schnitt von G (d.h. $\emptyset \neq W \neq V$). Sei ferner e eine leichteste Kante $\in W \times (V \setminus W) \cap E$. Dann gibt es einen minimalen Spannbaum von G, der e enthält. Ist e die einzige leichteste Kante in dem Schnitt, dann ist e in jedem minimalen Spannbaum von G enthalten.

Beweis:

Sei $(W, V \setminus W)$ ein Schnitt in G und e leichteste Kante in diesem Schnitt. Sei ferner M ein minimaler Spannbaum von G, der e nicht enthält. Durch Hinzufügen von e zu M entsteht (genau) ein Kreis, der, da M ein Spannbaum ist, mindestens eine weitere Kante f aus dem Schnitt $(W, V \setminus W)$ enthält. Entfernt man nun f, so entsteht wieder ein Spannbaum M', mit dem Gewicht

$$w(M') = w(M) - w(f) + w(e)\,.$$

Falls e einzige leichteste Kante im Schnitt ist, wäre $w(M') < w(M)$ im Widerspruch zur Tatsache, dass M minimaler Spannbaum von G ist. □

Beweis:
Sei $(W, V \setminus W)$ ein Schnitt in G und e leichteste Kante in diesem Schnitt. Sei ferner M ein minimaler Spannbaum von G, der e nicht enthält. Durch Hinzufügen von e zu M entsteht (genau) ein Kreis, der, da M ein Spannbaum ist, mindestens eine weitere Kante f aus dem Schnitt $(W, V \setminus W)$ enthält. Entfernt man nun f, so entsteht wieder ein Spannbaum M', mit dem Gewicht

$$w(M') = w(M) - w(f) + w(e)\,.$$

Falls e einzige leichteste Kante im Schnitt ist, wäre $w(M') < w(M)$ im Widerspruch zur Tatsache, dass M minimaler Spannbaum von G ist. □

Der Algorithmus $\textsc{Prim}(V, E, w)$:

$W := \{s\}$ für ein beliebiges $s \in V$; $T := \emptyset$

initialisiere Priority-Queue-Struktur R für V, Schlüssel $\rho(v)$ von v gleich

$$\rho(v) := \begin{cases} 0 & \text{falls } v = s \\ d(s,v) & \text{falls } v \in \Gamma(s) \\ \infty & \text{sonst} \end{cases}$$

$$\text{pred}[v] := \begin{cases} s & \text{falls } v \in \Gamma(s) \\ \text{nil} & \text{sonst} \end{cases}$$

while $W \neq V$ **do**
  $x :=$ ExtractMin(R)
  $W := W \cup \{x\}$; $T := T \cup \{x, \text{pred}[x]\}$
  for all $v \in \Gamma(x) \cap (V \setminus W)$ **do**
    if $\rho(v) > w(x,v)$ **then**
      DecreaseKey$(R, v, w(x,v))$; pred$[v] := x$
return T

Der Algorithmus $\textsc{Prim}(V, E, w)$:

$W := \{s\}$ für ein beliebiges $s \in V$; $T := \emptyset$

initialisiere Priority-Queue-Struktur R für V, Schlüssel $\rho(v)$ von v gleich

$$\rho(v) := \begin{cases} 0 & \text{falls } v = s \\ d(s,v) & \text{falls } v \in \Gamma(s) \\ \infty & \text{sonst} \end{cases}$$

$$\text{pred}[v] := \begin{cases} s & \text{falls } v \in \Gamma(s) \\ \text{nil} & \text{sonst} \end{cases}$$

while $W \neq V$ **do**

  $x :=$ ExtractMin(R)

  $W := W \cup \{x\}$; $T := T \cup \{x, \text{pred}[x]\}$

  for all $v \in \Gamma(x) \cap (V \setminus W)$ **do**

    if $\rho(v) > w(x,v)$ **then**

      DecreaseKey$(R, v, w(x,v))$; pred$[v] := x$

return T

Der Algorithmus $\textsc{Prim}(V, E, w)$:

$W := \{s\}$ für ein beliebiges $s \in V$; $T := \emptyset$

initialisiere Priority-Queue-Struktur R für V, Schlüssel $\rho(v)$ von v gleich

$$\rho(v) := \begin{cases} 0 & \text{falls } v = s \\ d(s,v) & \text{falls } v \in \Gamma(s) \\ \infty & \text{sonst} \end{cases}$$

$$\text{pred}[v] := \begin{cases} s & \text{falls } v \in \Gamma(s) \\ \text{nil} & \text{sonst} \end{cases}$$

while $W \neq V$ **do**

  $x :=$ ExtractMin(R)

  $W := W \cup \{x\}$; $T := T \cup \{x, \text{pred}[x]\}$

  for all $v \in \Gamma(x) \cap (V \setminus W)$ **do**

    if $\rho(v) > w(x,v)$ **then**

      DecreaseKey$(R, v, w(x,v))$; pred$[v] := x$

return T

Der Algorithmus $\textsc{Prim}(V, E, w)$:

$W := \{s\}$ für ein beliebiges $s \in V$; $T := \emptyset$

initialisiere Priority-Queue-Struktur R für V, Schlüssel $\rho(v)$ von v gleich

$$\rho(v) := \begin{cases} 0 & \text{falls } v = s \\ d(s,v) & \text{falls } v \in \Gamma(s) \\ \infty & \text{sonst} \end{cases}$$

$$\text{pred}[v] := \begin{cases} s & \text{falls } v \in \Gamma(s) \\ \text{nil} & \text{sonst} \end{cases}$$

while $W \neq V$ **do**

  $x :=$ ExtractMin(R)

  $W := W \cup \{x\}$; $T := T \cup \{x, \text{pred}[x]\}$

  for all $v \in \Gamma(x) \cap (V \setminus W)$ **do**

    if $\rho(v) > w(x,v)$ **then**

      DecreaseKey$(R, v, w(x,v))$; pred$[v] := x$

return T

Satz 220

Prim's Algorithmus bestimmt einen minimalen Spannbaum des zusammenhängenden Graphen $G = (V, E)$ in Zeit $O(m + n \log n)$ (bei Verwendung von Fibonacci-Heaps).

Beweis:

Betrachte in jeder Iteration den Schnitt $(W, V \setminus W)$. Die Korrektheit des Algorithmus folgt damit aus dem vorigen Lemma.

Der Algorithmus benötigt i.W. $\leq m$ DecreaseKey-Operationen und $n - 1$ ExtractMin-Operationen einer Priority-Queue. Damit ergibt sich die behauptete Zeitkomplexität. □

Satz 220

Prim's Algorithmus bestimmt einen minimalen Spannbaum des zusammenhängenden Graphen $G = (V, E)$ in Zeit $O(m + n \log n)$ (bei Verwendung von Fibonacci-Heaps).

Beweis:

Betrachte in jeder Iteration den Schnitt $(W, V \setminus W)$. Die Korrektheit des Algorithmus folgt damit aus dem vorigen Lemma.

Der Algorithmus benötigt i.W. $\leq m$ DecreaseKey-Operationen und $n - 1$ ExtractMin-Operationen einer Priority-Queue. Damit ergibt sich die behauptete Zeitkomplexität. □

Satz 220

Prim's Algorithmus bestimmt einen minimalen Spannbaum des zusammenhängenden Graphen $G = (V, E)$ in Zeit $O(m + n \log n)$ (bei Verwendung von Fibonacci-Heaps).

Beweis:

Betrachte in jeder Iteration den Schnitt $(W, V \setminus W)$. Die Korrektheit des Algorithmus folgt damit aus dem vorigen Lemma.

Der Algorithmus benötigt i.W. $\leq m$ DecreaseKey-Operationen und $n - 1$ ExtractMin-Operationen einer Priority-Queue. Damit ergibt sich die behauptete Zeitkomplexität. □

Kapitel IV Komplexitätstheorie

1. Definitionen

Definition 221

Sei M eine deterministische Turingmaschine und $\Sigma = \{0, 1\}$.

1. $\text{TIME}_M(x) :=$ Anzahl der Schritte, die M bei Eingabe $x \in \Sigma^*$ durchführt
2. $\text{DTIME}(f) :=$ Menge aller Sprachen, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $\text{TIME}_M(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{P} = \bigcup_{p \text{ Polynom}} \text{DTIME}(p)$$

$\mathcal{P}$ ist die Menge der polynomiell lösbaren Probleme und wird allgemein mit der Klasse der effizient lösbaren Probleme gleichgesetzt.

Kapitel IV Komplexitätstheorie

1. Definitionen

Definition 221

Sei M eine deterministische Turingmaschine und $\Sigma = \{0, 1\}$.

1. $\text{TIME}_M(x) :=$ Anzahl der Schritte, die M bei Eingabe $x \in \Sigma^*$ durchführt
2. $\text{DTIME}(f) :=$ Menge aller Sprachen, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $\text{TIME}_M(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{P} = \bigcup_{p \text{ Polynom}} \text{DTIME}(p)$$

$\mathcal{P}$ ist die Menge der polynomiell lösbaren Probleme und wird allgemein mit der Klasse der effizient lösbaren Probleme gleichgesetzt.

Kapitel IV Komplexitätstheorie

1. Definitionen

Definition 221

Sei M eine deterministische Turingmaschine und $\Sigma = \{0, 1\}$.

1. $\text{TIME}_M(x) :=$ Anzahl der Schritte, die M bei Eingabe $x \in \Sigma^*$ durchführt
2. $\text{DTIME}(f) :=$ Menge aller Sprachen, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $\text{TIME}_M(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{P} = \bigcup_{p \text{ Polynom}} \text{DTIME}(p)$$

$\mathcal{P}$ ist die Menge der polynomiell lösbaren Probleme und wird allgemein mit der Klasse der effizient lösbaren Probleme gleichgesetzt.

Kapitel IV Komplexitätstheorie

1. Definitionen

Definition 221

Sei M eine deterministische Turingmaschine und $\Sigma = \{0, 1\}$.

1. $\text{TIME}_M(x) :=$ Anzahl der Schritte, die M bei Eingabe $x \in \Sigma^*$ durchführt
2. $\text{DTIME}(f) :=$ Menge aller Sprachen, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $\text{TIME}_M(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{P} = \bigcup_{p \text{ Polynom}} \text{DTIME}(p)$$

$\mathcal{P}$ ist die Menge der polynomiell lösbaren Probleme und wird allgemein mit der Klasse der effizient lösbaren Probleme gleichgesetzt.

Kapitel IV Komplexitätstheorie

1. Definitionen

Definition 221

Sei M eine deterministische Turingmaschine und $\Sigma = \{0, 1\}$.

1. $\text{TIME}_M(x) :=$ Anzahl der Schritte, die M bei Eingabe $x \in \Sigma^*$ durchführt
2. $\text{DTIME}(f) :=$ Menge aller Sprachen, für die es eine deterministische Mehrband-Turingmaschine M gibt mit $\text{TIME}_M(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{P} = \bigcup_{p \text{ Polynom}} \text{DTIME}(p)$$

$\mathcal{P}$ ist die Menge der polynomiell lösbaren Probleme und wird allgemein mit der Klasse der effizient lösbaren Probleme gleichgesetzt.

Beispiel 222 (Probleme in $\mathcal{P}$)

- 2-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq 2$?

  2-COLORING $\in \mathcal{P}$, da $\chi(G) \leq 2$ gdw G bipartit.

- Shortest-Distance: Gegeben ein gewichteter Digraph $G = (V, E, d)$, $s, t \in V$ und eine Zahl D, ist die Entfernung von s nach t in $G \leq D$?

  Die Frage kann z.B. mit Dijkstra's Algorithmus in polynomieller Zeit entschieden werden.

Beispiel 222 (Probleme in $\mathcal{P}$)

- 2-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq 2$?
  2-COLORING $\in \mathcal{P}$, da $\chi(G) \leq 2$ gdw G bipartit.
- Shortest-Distance: Gegeben ein gewichteter Digraph $G = (V, E, d)$, $s, t \in V$ und eine Zahl D, ist die Entfernung von s nach t in $G \leq D$?

  Die Frage kann z.B. mit Dijkstra's Algorithmus in polynomieller Zeit entschieden werden.

Beispiel 222 (Probleme in $\mathcal{P}$)

- 2-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq 2$?
  2-COLORING $\in \mathcal{P}$, da $\chi(G) \leq 2$ gdw G bipartit.
- Shortest-Distance: Gegeben ein gewichteter Digraph $G = (V, E, d)$, $s, t \in V$ und eine Zahl D, ist die Entfernung von s nach t in $G \leq D$?

  Die Frage kann z.B. mit Dijkstra's Algorithmus in polynomieller Zeit entschieden werden.

Beispiel 222 (Probleme in $\mathcal{P}$)

- 2-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq 2$?
  2-COLORING $\in \mathcal{P}$, da $\chi(G) \leq 2$ gdw G bipartit.
- Shortest-Distance: Gegeben ein gewichteter Digraph $G = (V, E, d)$, $s, t \in V$ und eine Zahl D, ist die Entfernung von s nach t in $G \leq D$?

  Die Frage kann z.B. mit Dijkstra's Algorithmus in polynomieller Zeit entschieden werden.

Beispiel 222 (Probleme in $\mathcal{P}$)

- 2-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq 2$?

  2-COLORING $\in \mathcal{P}$, da $\chi(G) \leq 2$ gdw G bipartit.
- Shortest-Distance: Gegeben ein gewichteter Digraph $G = (V, E, d)$, $s, t \in V$ und eine Zahl D, ist die Entfernung von s nach t in $G \leq D$?

  Die Frage kann z.B. mit Dijkstra's Algorithmus in polynomieller Zeit entschieden werden.

Definition 223

Sei N eine nichtdeterministische Turingmaschine. Dann

1. $$\mathsf{TIME}_N(x) := \begin{cases} \text{minimale Anzahl der} \\ \text{Schritte, die } N \text{ benötigt,} & \text{falls } x \in L(N) \\ \text{um } x \in \Sigma^* \text{ zu akzeptieren} \\ \\ 0 & \text{sonst} \end{cases}$$
2. $\mathsf{NTIME}(f) :=$ Menge aller Sprachen, für die es eine nichtdeterministische Mehrband-Turingmaschine N gibt mit $\mathsf{TIME}_N(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{NP} = \bigcup_{p \text{ Polynom}} \mathsf{NTIME}(p)$$

Definition 223

Sei N eine nichtdeterministische Turingmaschine. Dann

1. $$\mathrm{TIME}_N(x) := \begin{cases} \text{minimale Anzahl der} \\ \text{Schritte, die } N \text{ benötigt,} & \text{falls } x \in L(N) \\ \text{um } x \in \Sigma^* \text{ zu akzeptieren} \\ \\ 0 & \text{sonst} \end{cases}$$
2. $\mathrm{NTIME}(f) :=$ Menge aller Sprachen, für die es eine nichtdeterministische Mehrband-Turingmaschine N gibt mit $\mathrm{TIME}_N(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{NP} = \bigcup_{p \text{ Polynom}} \mathrm{NTIME}(p)$$

Definition 223

Sei N eine nichtdeterministische Turingmaschine. Dann

1. $$\mathrm{TIME}_N(x) := \begin{cases} \text{minimale Anzahl der} \\ \text{Schritte, die } N \text{ benötigt,} & \text{falls } x \in L(N) \\ \text{um } x \in \Sigma^* \text{ zu akzeptieren} \\ \\ 0 & \text{sonst} \end{cases}$$
2. $\mathrm{NTIME}(f) :=$ Menge aller Sprachen, für die es eine nichtdeterministische Mehrband-Turingmaschine N gibt mit $\mathrm{TIME}_N(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{NP} = \bigcup_{p \text{ Polynom}} \mathrm{NTIME}(p)$$

Definition 223

Sei N eine nichtdeterministische Turingmaschine. Dann

1. $$\mathrm{TIME}_N(x) := \begin{cases} \text{minimale Anzahl der} \\ \text{Schritte, die } N \text{ benötigt,} & \text{falls } x \in L(N) \\ \text{um } x \in \Sigma^* \text{ zu akzeptieren} \\ \\ 0 & \text{sonst} \end{cases}$$
2. $\mathrm{NTIME}(f) :=$ Menge aller Sprachen, für die es eine nichtdeterministische Mehrband-Turingmaschine N gibt mit $\mathrm{TIME}_N(x) \leq f(|x|)$ für alle $x \in \Sigma^*$
3. $$\mathcal{NP} = \bigcup_{p \text{ Polynom}} \mathrm{NTIME}(p)$$

Man beachte, da wir bei nichtdeterministischen Maschinen i.W. an akzeptierenden Berechnungen interessiert sind, die etwas ungewöhnliche Festlegung für $x \notin L(N)$.

Eine alternative Definition für $\mathcal{NP}$ ist

Definition 224

$\mathcal{NP}$ ist die Klasse der Probleme L, für die es ein Prädikat $P \in \mathcal{P}$ und ein Polynom p gibt, so dass

$$(\forall x \in \Sigma^*)\,[x \in L \Leftrightarrow (\exists y \in \Sigma^*)[|y| \leq p(|x|) \wedge P(x,y)]]$$

Man beachte, da wir bei nichtdeterministischen Maschinen i.W. an akzeptierenden Berechnungen interessiert sind, die etwas ungewöhnliche Festlegung für $x \notin L(N)$.

Eine alternative Definition für $\mathcal{NP}$ ist

Definition 224

$\mathcal{NP}$ ist die Klasse der Probleme L, für die es ein Prädikat $P \in \mathcal{P}$ und ein Polynom p gibt, so dass

$$(\forall x \in \Sigma^*)\,[x \in L \Leftrightarrow (\exists y \in \Sigma^*)[|y| \leq p(|x|) \wedge P(x,y)]]$$

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)

- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!

- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)

- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!

- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)
- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!
- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)
- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!
- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)
- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!
- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)
- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!
- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Beispiel 225 (Probleme in $\mathcal{NP}$)

- k-COLORING: Gegeben ein Graph $G = (V, E)$, ist $\chi(G) \leq k$?

  k-COLORING ist in $\mathcal{NP}$, da eine nichtdet. TM eine gültige k-Färbung raten und dann verifizieren kann (in der alternativen Definition wäre y die Beschreibung einer solchen k-Färbung!)
- COLORING: Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$, ist $\chi(G) \leq k$?

  Begründung i.W. wie oben; hier ist jedoch k Teil der Probleminstanz, also der Eingabe!
- SAT: Gegeben eine boolesche Formel F, hat F eine erfüllende Belegung?

  Eine nichtdet. TM kann einfach eine erfüllende Belegung, falls eine solche existiert, raten und verifizieren.

Eine der wichtigsten offenen Fragen der Informatik (und nicht nur der Theoretischen Informatik!) ist

$$\mathcal{P} =^{?} \mathcal{NP}$$

Eine der wichtigsten offenen Fragen der Theoretischen Informatik ist

Wie könnte man $\mathcal{P} \neq \mathcal{NP}$ beweisen?

Eine der wichtigsten offenen Fragen der Informatik (und nicht nur der Theoretischen Informatik!) ist

$$\mathcal{P} =^{?} \mathcal{NP}$$

Eine der wichtigsten offenen Fragen der Theoretischen Informatik ist

Wie könnte man $\mathcal{P} \neq \mathcal{NP}$ beweisen?

2. $\mathcal{NP}$-Vollständigkeit

Definition 226

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A polynomiell reduzierbar auf B (polynomiell many-one-reduzierbar; i.Z. $A \preceq_p B$ oder $A \preceq_p^{\text{m-1}} B$ oder auch $A \leq_p B$), falls es eine polynomiell berechenbare totale Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass gilt

$$(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$$

Bemerkungen:

1. Die Relation $\preceq_p$ ist transitiv, d.h.

$$A \preceq_p B \wedge B \preceq_p C \Rightarrow A \preceq_p C$$

2. $$A \preceq_p B \wedge B \in \mathcal{P} \Rightarrow A \in \mathcal{P}$$

3. $$A \preceq_p B \wedge B \in \mathcal{NP} \Rightarrow A \in \mathcal{NP}$$

2. $\mathcal{NP}$-Vollständigkeit

Definition 226

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A polynomiell reduzierbar auf B (polynomiell many-one-reduzierbar; i.Z. $A \preceq_p B$ oder $A \preceq_p^{\text{m-1}} B$ oder auch $A \leq_p B$), falls es eine polynomiell berechenbare totale Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass gilt

$$(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$$

Bemerkungen:

1. Die Relation $\preceq_p$ ist transitiv, d.h.

$$A \preceq_p B \wedge B \preceq_p C \Rightarrow A \preceq_p C$$

2. $$A \preceq_p B \wedge B \in \mathcal{P} \Rightarrow A \in \mathcal{P}$$

3. $$A \preceq_p B \wedge B \in \mathcal{NP} \Rightarrow A \in \mathcal{NP}$$

2. $\mathcal{NP}$-Vollständigkeit

Definition 226

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A polynomiell reduzierbar auf B (polynomiell many-one-reduzierbar; i.Z. $A \preceq_p B$ oder $A \preceq_p^{\text{m-1}} B$ oder auch $A \leq_p B$), falls es eine polynomiell berechenbare totale Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass gilt

$$(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$$

Bemerkungen:

1. Die Relation $\preceq_p$ ist transitiv, d.h.

$$A \preceq_p B \wedge B \preceq_p C \Rightarrow A \preceq_p C$$

2. $$A \preceq_p B \wedge B \in \mathcal{P} \Rightarrow A \in \mathcal{P}$$

3. $$A \preceq_p B \wedge B \in \mathcal{NP} \Rightarrow A \in \mathcal{NP}$$

2. $\mathcal{NP}$-Vollständigkeit

Definition 226

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A polynomiell reduzierbar auf B (polynomiell many-one-reduzierbar; i.Z. $A \preceq_p B$ oder $A \preceq_p^{\text{m-1}} B$ oder auch $A \leq_p B$), falls es eine polynomiell berechenbare totale Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass gilt

$$(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$$

Bemerkungen:

1. Die Relation $\preceq_p$ ist transitiv, d.h.

$$A \preceq_p B \wedge B \preceq_p C \Rightarrow A \preceq_p C$$

2.

$$A \preceq_p B \wedge B \in \mathcal{P} \Rightarrow A \in \mathcal{P}$$

3.

$$A \preceq_p B \wedge B \in \mathcal{NP} \Rightarrow A \in \mathcal{NP}$$

2. $\mathcal{NP}$-Vollständigkeit

Definition 226

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A polynomiell reduzierbar auf B (polynomiell many-one-reduzierbar; i.Z. $A \preceq_p B$ oder $A \preceq_p^{\text{m-1}} B$ oder auch $A \leq_p B$), falls es eine polynomiell berechenbare totale Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass gilt

$$(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$$

Bemerkungen:

1. Die Relation $\preceq_p$ ist transitiv, d.h.

$$A \preceq_p B \wedge B \preceq_p C \Rightarrow A \preceq_p C$$

2. $$A \preceq_p B \wedge B \in \mathcal{P} \Rightarrow A \in \mathcal{P}$$

3. $$A \preceq_p B \wedge B \in \mathcal{NP} \Rightarrow A \in \mathcal{NP}$$

Definition 227

1. Eine Sprache A heißt $\mathcal{NP}$-schwer (oder $\mathcal{NP}$-hart, engl. $\mathcal{NP}$-hard), falls für alle Sprachen B aus $\mathcal{NP}$ gilt:

$$B \preceq_p A$$

2. Eine Sprache A heißt $\mathcal{NP}$-vollständig (engl. $\mathcal{NP}$-complete), falls $A \in \mathcal{NP}$ und A $\mathcal{NP}$-schwer ist.

Intuitiv sind die $\mathcal{NP}$-vollständigen Probleme die "schwersten" Probleme in $\mathcal{NP}$.

$\mathcal{NP}$-harte Probleme sind "mindestens so schwer" wie alle Probleme in $\mathcal{NP}$, müssen aber selbst nicht in $\mathcal{NP}$ sein, können also noch viel höhere Komplexität haben.

Definition 227

1. Eine Sprache A heißt $\mathcal{NP}$-schwer (oder $\mathcal{NP}$-hart, engl. $\mathcal{NP}$-hard), falls für alle Sprachen B aus $\mathcal{NP}$ gilt:

$$B \preceq_p A$$

2. Eine Sprache A heißt $\mathcal{NP}$-vollständig (engl. $\mathcal{NP}$-complete), falls $A \in \mathcal{NP}$ und A $\mathcal{NP}$-schwer ist.

Intuitiv sind die $\mathcal{NP}$-vollständigen Probleme die "schwersten" Probleme in $\mathcal{NP}$.

$\mathcal{NP}$-harte Probleme sind "mindestens so schwer" wie alle Probleme in $\mathcal{NP}$, müssen aber selbst nicht in $\mathcal{NP}$ sein, können also noch viel höhere Komplexität haben.

Definition 227

1. Eine Sprache A heißt $\mathcal{NP}$-schwer (oder $\mathcal{NP}$-hart, engl. $\mathcal{NP}$-hard), falls für alle Sprachen B aus $\mathcal{NP}$ gilt:

$$B \preceq_p A$$

2. Eine Sprache A heißt $\mathcal{NP}$-vollständig (engl. $\mathcal{NP}$-complete), falls $A \in \mathcal{NP}$ und A $\mathcal{NP}$-schwer ist.

Intuitiv sind die $\mathcal{NP}$-vollständigen Probleme die "schwersten" Probleme in $\mathcal{NP}$.

$\mathcal{NP}$-harte Probleme sind "mindestens so schwer" wie alle Probleme in $\mathcal{NP}$, müssen aber selbst nicht in $\mathcal{NP}$ sein, können also noch viel höhere Komplexität haben.

Definition 227

1. Eine Sprache A heißt $\mathcal{NP}$-schwer (oder $\mathcal{NP}$-hart, engl. $\mathcal{NP}$-hard), falls für alle Sprachen B aus $\mathcal{NP}$ gilt:

$$B \preceq_p A$$

2. Eine Sprache A heißt $\mathcal{NP}$-vollständig (engl. $\mathcal{NP}$-complete), falls $A \in \mathcal{NP}$ und A $\mathcal{NP}$-schwer ist.

Intuitiv sind die $\mathcal{NP}$-vollständigen Probleme die "schwersten" Probleme in $\mathcal{NP}$.

$\mathcal{NP}$-harte Probleme sind "mindestens so schwer" wie alle Probleme in $\mathcal{NP}$, müssen aber selbst nicht in $\mathcal{NP}$ sein, können also noch viel höhere Komplexität haben.

Definition 227

1. Eine Sprache A heißt $\mathcal{NP}$-schwer (oder $\mathcal{NP}$-hart, engl. $\mathcal{NP}$-hard), falls für alle Sprachen B aus $\mathcal{NP}$ gilt:

$$B \preceq_p A$$

2. Eine Sprache A heißt $\mathcal{NP}$-vollständig (engl. $\mathcal{NP}$-complete), falls $A \in \mathcal{NP}$ und A $\mathcal{NP}$-schwer ist.

Intuitiv sind die $\mathcal{NP}$-vollständigen Probleme die "schwersten" Probleme in $\mathcal{NP}$.

$\mathcal{NP}$-harte Probleme sind "mindestens so schwer" wie alle Probleme in $\mathcal{NP}$, müssen aber selbst nicht in $\mathcal{NP}$ sein, können also noch viel höhere Komplexität haben.

Satz 228

SAT *ist $\mathcal{NP}$-vollständig.*

Satz 228

SAT *ist $\mathcal{NP}$-vollständig.*

Beweis:

Offensichtlich ist SAT $\in \mathcal{NP}$.

Es bleibt zu zeigen, dass SAT $\mathcal{NP}$-hart ist, d.h. $\forall L \in \mathcal{NP}$ $L \preceq_p SAT$.

Wir tun dies, indem wir die Berechnung einer nichtdet. TM so in einer Formel kodieren, dass diese genau dann erfüllbar ist, wenn die Maschine in polynomiell vielen Schritten akzeptiert.

Satz 228

SAT *ist $\mathcal{NP}$-vollständig.*

Beweis:

Offensichtlich ist SAT $\in \mathcal{NP}$.

Es bleibt zu zeigen, dass SAT $\mathcal{NP}$-hart ist, d.h. $\forall L \in \mathcal{NP}$ $L \preceq_p SAT$.

Wir tun dies, indem wir die Berechnung einer nichtdet. TM so in einer Formel kodieren, dass diese genau dann erfüllbar ist, wenn die Maschine in polynomiell vielen Schritten akzeptiert.

Satz 228

SAT *ist $\mathcal{NP}$-vollständig.*

Beweis:

Offensichtlich ist SAT $\in \mathcal{NP}$.

Es bleibt zu zeigen, dass SAT $\mathcal{NP}$-hart ist, d.h. $\forall L \in \mathcal{NP}$ $L \preceq_p SAT$.

Wir tun dies, indem wir die Berechnung einer nichtdet. TM so in einer Formel kodieren, dass diese genau dann erfüllbar ist, wenn die Maschine in polynomiell vielen Schritten akzeptiert.

Beweis:

Für jedes $L \in \mathcal{NP}$ gibt es eine Maschine M und ein Polynom p, so dass $(\forall x \in \Sigma^*)\,[\mathsf{TIME}_M(x) \leq p(|x|)]$.

Für die Reduktion zeigen wir jetzt, wie man daraus eine Formel F von polynomieller Größe q konstruiert mit

$$x = x_1 \cdots x_n \in L \quad \text{gdw} \quad F \text{ ist erfüllbar}$$

Die Formel F besteht aus mehreren Teilen:

$$F = R \wedge A \wedge U \wedge E$$

Beweis:

Für jedes $L \in \mathcal{NP}$ gibt es eine Maschine M und ein Polynom p, so dass $(\forall x \in \Sigma^*)\,[\mathsf{TIME}_M(x) \leq p(|x|)]$.

Für die Reduktion zeigen wir jetzt, wie man daraus eine Formel F von polynomieller Größe q konstruiert mit

$$x = x_1 \cdots x_n \in L \quad \text{gdw} \quad F \text{ ist erfüllbar}$$

Die Formel F besteht aus mehreren Teilen:

$$F = R \wedge A \wedge U \wedge E$$

Beweis:

Für jedes $L \in \mathcal{NP}$ gibt es eine Maschine M und ein Polynom p, so dass $(\forall x \in \Sigma^*)\,[\mathsf{TIME}_M(x) \leq p(|x|)]$.

Für die Reduktion zeigen wir jetzt, wie man daraus eine Formel F von polynomieller Größe q konstruiert mit

$$x = x_1 \cdots x_n \in L \quad \text{gdw} \quad F \text{ ist erfüllbar}$$

Die Formel F besteht aus mehreren Teilen:

$$F = R \wedge A \wedge U \wedge E$$

Beweis:

Wir führen folgende Variablen ein:

- $B^s_{i,t}$ – die i-te Zelle des Bandes enthält zum Zeitpunkt t das Symbol s,
- Z^q_t – die Turingmaschine befindet sich zum Zeitpunkt t in Zustand q,
- P^i_t – der Schreib-/Lesekopf der Turingmaschine befindet sich zum Zeitpunkt t an Position i.

Für $|x| = n$ sei $\hat{t} = p(n)$. Dann ist $1 \leq t \leq \hat{t}$, $-\hat{t} \leq i \leq \hat{t}$, $q \in Q$ und $s \in \Sigma$.

Insgesamt haben wir also nur $O(\hat{t}^2) = O(p(n)^2)$ viele Variablen.

Beweis:

Wir führen folgende Variablen ein:

- $B^s_{i,t}$ – die i-te Zelle des Bandes enthält zum Zeitpunkt t das Symbol s,
- Z^q_t – die Turingmaschine befindet sich zum Zeitpunkt t in Zustand q,
- P^i_t – der Schreib-/Lesekopf der Turingmaschine befindet sich zum Zeitpunkt t an Position i.

Für $|x| = n$ sei $\hat{t} = p(n)$. Dann ist $1 \leq t \leq \hat{t}$, $-\hat{t} \leq i \leq \hat{t}$, $q \in Q$ und $s \in \Sigma$.

Insgesamt haben wir also nur $O(\hat{t}^2) = O(p(n)^2)$ viele Variablen.

Beweis:

Wir führen folgende Variablen ein:

- $B^s_{i,t}$ – die i-te Zelle des Bandes enthält zum Zeitpunkt t das Symbol s,
- Z^q_t – die Turingmaschine befindet sich zum Zeitpunkt t in Zustand q,
- P^i_t – der Schreib-/Lesekopf der Turingmaschine befindet sich zum Zeitpunkt t an Position i.

Für $|x| = n$ sei $\hat{t} = p(n)$. Dann ist $1 \leq t \leq \hat{t}$, $-\hat{t} \leq i \leq \hat{t}$, $q \in Q$ und $s \in \Sigma$.

Insgesamt haben wir also nur $O(\hat{t}^2) = O(p(n)^2)$ viele Variablen.

Beweis:

Wir führen folgende Variablen ein:

- $B^s_{i,t}$ – die i-te Zelle des Bandes enthält zum Zeitpunkt t das Symbol s,
- Z^q_t – die Turingmaschine befindet sich zum Zeitpunkt t in Zustand q,
- P^i_t – der Schreib-/Lesekopf der Turingmaschine befindet sich zum Zeitpunkt t an Position i.

Für $|x| = n$ sei $\hat{t} = p(n)$. Dann ist $1 \leq t \leq \hat{t}$, $-\hat{t} \leq i \leq \hat{t}$, $q \in Q$ und $s \in \Sigma$.

Insgesamt haben wir also nur $O(\hat{t}^2) = O(p(n)^2)$ viele Variablen.

Beweis:

Wir führen folgende Variablen ein:

- $B_{i,t}^{s}$ – die i-te Zelle des Bandes enthält zum Zeitpunkt t das Symbol s,
- Z_{t}^{q} – die Turingmaschine befindet sich zum Zeitpunkt t in Zustand q,
- P_{t}^{i} – der Schreib-/Lesekopf der Turingmaschine befindet sich zum Zeitpunkt t an Position i.

Für $|x| = n$ sei $\hat{t} = p(n)$. Dann ist $1 \leq t \leq \hat{t}$, $-\hat{t} \leq i \leq \hat{t}$, $q \in Q$ und $s \in \Sigma$.

Insgesamt haben wir also nur $O(\hat{t}^2) = O(p(n)^2)$ viele Variablen.

Beweis:

Wir führen folgende Variablen ein:

- $B_{i,t}^{s}$ – die i-te Zelle des Bandes enthält zum Zeitpunkt t das Symbol s,
- Z_{t}^{q} – die Turingmaschine befindet sich zum Zeitpunkt t in Zustand q,
- P_{t}^{i} – der Schreib-/Lesekopf der Turingmaschine befindet sich zum Zeitpunkt t an Position i.

Für $|x| = n$ sei $\hat{t} = p(n)$. Dann ist $1 \leq t \leq \hat{t}$, $-\hat{t} \leq i \leq \hat{t}$, $q \in Q$ und $s \in \Sigma$.

Insgesamt haben wir also nur $O(\hat{t}^2) = O(p(n)^2)$ viele Variablen.

Beweis:

Die Teilformel R gibt die Randbedingungen, dass

1. die Turingmaschine zu jedem Zeitpunkt t auf genau einem Feld i steht

$$\bigwedge_{1\le t\le \hat{t}} \left(\left(\bigvee_{-\hat{t}\le i\le \hat{t}} P_t^i \right) \wedge \left(\bigwedge_{i\neq j} P_t^i \Rightarrow \neg P_t^j \right) \right) \ ,$$

2. die Turingmaschine zu jedem Zeitpunkt t in genau einem Zustand q ist

$$\bigwedge_{1\le t\le \hat{t}} \left(\left(\bigvee_{q\in Q} Z_t^q \right) \wedge \left(\bigwedge_{q\neq q'} Z_t^q \Rightarrow \neg Z_t^{q'} \right) \right) \text{ und}$$

3. zu jedem Zeitpunkt t an jeder Bandposition i genau ein Symbol s steht

$$\bigwedge_{\substack{1\le t\le \hat{t} \\ -\hat{t}\le i\le \hat{t}}} \left(\left(\bigvee_{s\in\Sigma} B_{i,t}^s \right) \wedge \left(\bigwedge_{s\neq s'} B_{i,t}^s \Rightarrow \neg B_{i,t}^{s'} \right) \right)$$

Beweis:

Die Teilformel R gibt die Randbedingungen, dass

1. die Turingmaschine zu jedem Zeitpunkt t auf genau einem Feld i steht

$$\bigwedge_{1\leq t\leq \hat{t}} \left(\left(\bigvee_{-\hat{t}\leq i\leq \hat{t}} P_t^i \right) \wedge \left(\bigwedge_{i\neq j} P_t^i \Rightarrow \neg P_t^j \right) \right) \quad ,$$

2. die Turingmaschine zu jedem Zeitpunkt t in genau einem Zustand q ist

$$\bigwedge_{1\leq t\leq \hat{t}} \left(\left(\bigvee_{q\in Q} Z_t^q \right) \wedge \left(\bigwedge_{q\neq q'} Z_t^q \Rightarrow \neg Z_t^{q'} \right) \right) \quad \text{und}$$

3. zu jedem Zeitpunkt t an jeder Bandposition i genau ein Symbol s steht

$$\bigwedge_{\substack{1\leq t\leq \hat{t} \\ -\hat{t}\leq i\leq \hat{t}}} \left(\left(\bigvee_{s\in\Sigma} B_{i,t}^s \right) \wedge \left(\bigwedge_{s\neq s'} B_{i,t}^s \Rightarrow \neg B_{i,t}^{s'} \right) \right)$$

Beweis:

Die Teilformel R gibt die Randbedingungen, dass

1. die Turingmaschine zu jedem Zeitpunkt t auf genau einem Feld i steht

$$\bigwedge_{1\leq t\leq \hat{t}} \left(\left(\bigvee_{-\hat{t}\leq i\leq \hat{t}} P_t^i \right) \wedge \left(\bigwedge_{i\neq j} P_t^i \Rightarrow \neg P_t^j \right) \right) \quad ,$$

2. die Turingmaschine zu jedem Zeitpunkt t in genau einem Zustand q ist

$$\bigwedge_{1\leq t\leq \hat{t}} \left(\left(\bigvee_{q\in Q} Z_t^q \right) \wedge \left(\bigwedge_{q\neq q'} Z_t^q \Rightarrow \neg Z_t^{q'} \right) \right) \quad \text{und}$$

3. zu jedem Zeitpunkt t an jeder Bandposition i genau ein Symbol s steht

$$\bigwedge_{\substack{1\leq t\leq \hat{t} \\ -\hat{t}\leq i\leq \hat{t}}} \left(\left(\bigvee_{s\in\Sigma} B_{i,t}^s \right) \wedge \left(\bigwedge_{s\neq s'} B_{i,t}^s \Rightarrow \neg B_{i,t}^{s'} \right) \right)$$

Beweis:

Die Teilformel R gibt die Randbedingungen, dass

1. die Turingmaschine zu jedem Zeitpunkt t auf genau einem Feld i steht

$$\bigwedge_{1\leq t\leq \hat{t}} \left(\left(\bigvee_{-\hat{t}\leq i\leq \hat{t}} P_t^i \right) \wedge \left(\bigwedge_{i\neq j} P_t^i \Rightarrow \neg P_t^j \right) \right) \quad ,$$

2. die Turingmaschine zu jedem Zeitpunkt t in genau einem Zustand q ist

$$\bigwedge_{1\leq t\leq \hat{t}} \left(\left(\bigvee_{q\in Q} Z_t^q \right) \wedge \left(\bigwedge_{q\neq q'} Z_t^q \Rightarrow \neg Z_t^{q'} \right) \right) \quad \text{und}$$

3. zu jedem Zeitpunkt t an jeder Bandposition i genau ein Symbol s steht

$$\bigwedge_{\substack{1\leq t\leq \hat{t} \\ -\hat{t}\leq i\leq \hat{t}}} \left(\left(\bigvee_{s\in\Sigma} B_{i,t}^s \right) \wedge \left(\bigwedge_{s\neq s'} B_{i,t}^s \Rightarrow \neg B_{i,t}^{s'} \right) \right)$$

Beweis:

Die Teilformel A gibt die Anfangsbedingung, dass zum Zeitpunkt 1 die Turingmaschine im Startzustand q_0 an Position 1 des Bandes ist und auf dem Band die Eingabe $x_1 \cdots x_n$ steht:

$$Z_1^{q_0} \wedge P_1^1 \wedge \bigwedge_{1 \leq i \leq n} B_{i,1}^{x_i} \wedge \bigwedge_{\substack{-\hat{t} \leq i < 1 \vee \\ n < i \leq \hat{t}}} B_{i,1}^{\square},$$

wobei $\square$ das Leerzeichen ist.

Beweis:

Die Teilformel U stellt sicher, dass die Übergänge zwischen zwei Zeitpunkten t und $t+1$ entsprechend der Übergangsrelation $\delta : Q \times \Sigma \to 2^{Q \times \Sigma \times \{-1,0,1\}}$ der Turingmaschine sind. Die Teilformel U ergibt sich aus der Konjunktion von

$$\left(\neg P_t^i \wedge B_{i,t}^s \Rightarrow B_{i,t+1}^s\right)$$

und

$$P_t^i \wedge \bigvee_{(q',s',d)\in\delta(q,s)} \left(B_{i,t}^s \wedge Z_t^q \Rightarrow B_{i,t+1}^{s'} \wedge Z_{t+1}^{q'} \wedge P_{t+1}^{i+d}\right)$$

für alle $1 \leq t < \hat{t}$ und alle $-\hat{t} \leq i \leq \hat{t}$.

Beweis:

Die Teilformel U stellt sicher, dass die Übergänge zwischen zwei Zeitpunkten t und $t+1$ entsprechend der Übergangsrelation $\delta : Q \times \Sigma \to 2^{Q \times \Sigma \times \{-1,0,1\}}$ der Turingmaschine sind. Die Teilformel U ergibt sich aus der Konjunktion von

$$\left(\neg P_t^i \wedge B_{i,t}^s \Rightarrow B_{i,t+1}^s\right)$$

und

$$P_t^i \wedge \bigvee_{(q',s',d)\in\delta(q,s)} \left(B_{i,t}^s \wedge Z_t^q \Rightarrow B_{i,t+1}^{s'} \wedge Z_{t+1}^{q'} \wedge P_{t+1}^{i+d}\right)$$

für alle $1 \leq t < \hat{t}$ und alle $-\hat{t} \leq i \leq \hat{t}$.

Beweis:

Die Teilformel U stellt sicher, dass die Übergänge zwischen zwei Zeitpunkten t und $t+1$ entsprechend der Übergangsrelation $\delta : Q \times \Sigma \to 2^{Q \times \Sigma \times \{-1,0,1\}}$ der Turingmaschine sind. Die Teilformel U ergibt sich aus der Konjunktion von

$$\left(\neg P_t^i \wedge B_{i,t}^s \Rightarrow B_{i,t+1}^s\right)$$

und

$$P_t^i \wedge \bigvee_{(q',s',d)\in\delta(q,s)} \left(B_{i,t}^s \wedge Z_t^q \Rightarrow B_{i,t+1}^{s'} \wedge Z_{t+1}^{q'} \wedge P_{t+1}^{i+d}\right)$$

für alle $1 \leq t < \hat{t}$ und alle $-\hat{t} \leq i \leq \hat{t}$.

Beweis:

Die Teilformel U stellt sicher, dass die Übergänge zwischen zwei Zeitpunkten t und $t+1$ entsprechend der Übergangsrelation $\delta : Q \times \Sigma \to 2^{Q \times \Sigma \times \{-1,0,1\}}$ der Turingmaschine sind. Die Teilformel U ergibt sich aus der Konjunktion von

$$\left(\neg P_t^i \wedge B_{i,t}^s \Rightarrow B_{i,t+1}^s\right)$$

und

$$P_t^i \wedge \bigvee_{(q',s',d)\in\delta(q,s)} \left(B_{i,t}^s \wedge Z_t^q \Rightarrow B_{i,t+1}^{s'} \wedge Z_{t+1}^{q'} \wedge P_{t+1}^{i+d}\right)$$

für alle $1 \leq t < \hat{t}$ und alle $-\hat{t} \leq i \leq \hat{t}$.

Beweis:

Die Teilformel E modelliert schließlich die Bedingung, dass die Turingmaschine einen akzeptierenden Zustand erreicht.
O.B.d.A. nehmen wir an, dass M nach Erreichen eines Endzustandes in diesem verbleibt, d.h. für alle $q \in E$, $s \in \Sigma$ gilt $(q, s, 0) \in \delta(q, s)$. Dann ist E geben durch:

$$\bigvee_{\substack{q \in E \\ 1 \leq t < \hat{t}}} Z_t^q$$

Daher ergibt sich:

$$x \in L \text{ gdw } F \text{ ist erfüllbar.}$$

Beachte: Für $n = |x|$ hat F die Größe $O(p(n)^3)$.

Beweis:

Die Teilformel E modelliert schließlich die Bedingung, dass die Turingmaschine einen akzeptierenden Zustand erreicht.
O.B.d.A. nehmen wir an, dass M nach Erreichen eines Endzustandes in diesem verbleibt, d.h. für alle $q \in E$, $s \in \Sigma$ gilt $(q, s, 0) \in \delta(q, s)$. Dann ist E geben durch:

$$\bigvee_{\substack{q \in E \\ 1 \leq t < \hat{t}}} Z_t^q$$

Daher ergibt sich:

$$x \in L \text{ gdw } F \text{ ist erfüllbar.}$$

Beachte: Für $n = |x|$ hat F die Größe $O(p(n)^3)$.

Beweis:

Die Teilformel E modelliert schließlich die Bedingung, dass die Turingmaschine einen akzeptierenden Zustand erreicht.
O.B.d.A. nehmen wir an, dass M nach Erreichen eines Endzustandes in diesem verbleibt, d.h. für alle $q \in E$, $s \in \Sigma$ gilt $(q, s, 0) \in \delta(q, s)$. Dann ist E geben durch:

$$\bigvee_{\substack{q \in E \\ 1 \leq t < \hat{t}}} Z_t^q$$

Daher ergibt sich:

$$x \in L \text{ gdw } F \text{ ist erfüllbar.}$$

Beachte: Für $n = |x|$ hat F die Größe $O(p(n)^3)$. □

Definition 229

3SAT ist die Menge der booleschen Formeln in konjunktiver Normalform, die in jeder Klausel höchstens drei Literale enthalten und die erfüllbar sind.

Satz 230

3SAT *ist $\mathcal{NP}$-vollständig.*

Definition 229

3SAT ist die Menge der booleschen Formeln in konjunktiver Normalform, die in jeder Klausel höchstens drei Literale enthalten und die erfüllbar sind.

Satz 230

3SAT *ist $\mathcal{NP}$-vollständig.*

Beweis:

Offensichtlich ist 3SAT $\in \mathcal{NP}$.

Es bleibt zu zeigen, dass 3SAT $\mathcal{NP}$-schwer ist.

Wir tun dies, indem wir SAT polynomiell auf 3SAT reduzieren.

Beweis:
Offensichtlich ist 3SAT $\in \mathcal{NP}$.

Es bleibt zu zeigen, dass 3SAT $\mathcal{NP}$-schwer ist.

Wir tun dies, indem wir SAT polynomiell auf 3SAT reduzieren.

Beweis:
Offensichtlich ist 3SAT $\in \mathcal{NP}$.

Es bleibt zu zeigen, dass 3SAT $\mathcal{NP}$-schwer ist.

Wir tun dies, indem wir SAT polynomiell auf 3SAT reduzieren.

Beweis:

Sei F eine beliebige boolesche Formel in konjunktiver Normalform. Wir ersetzen jede Klausel

$$(x_1 \vee x_2 \vee \ldots \vee x_k)$$

(x_i bezeichnet hier ein beliebiges Literal) durch

$$(x_1 \vee x_2 \vee z_2) \wedge (\overline{z_2} \vee x_3 \vee z_3) \wedge \ldots \wedge (\overline{z_{k-2}} \vee x_{k-1} \vee x_k)\,,$$

wobei $z_2, \ldots, z_{k-2}$ neue Variable sind.

Beweis:

Sei F eine beliebige boolesche Formel in konjunktiver Normalform.
Wir ersetzen jede Klausel

$$(x_1 \vee x_2 \vee \ldots \vee x_k)$$

(x_i bezeichnet hier ein beliebiges Literal) durch

$$(x_1 \vee x_2 \vee z_2) \wedge (\overline{z_2} \vee x_3 \vee z_3) \wedge \ldots \wedge (\overline{z_{k-2}} \vee x_{k-1} \vee x_k)\,,$$

wobei $z_2, \ldots, z_{k-2}$ neue Variable sind.
Es gibt eine Belegung für die z_i, so dass alle Klauseln erfüllt sind, gdw mindestens eines der Literale x_j wahr ist.

Beweis:

Sei F eine beliebige boolesche Formel in konjunktiver Normalform. Wir ersetzen jede Klausel

$$(x_1 \vee x_2 \vee \ldots \vee x_k)$$

(x_i bezeichnet hier ein beliebiges Literal) durch

$$(x_1 \vee x_2 \vee z_2) \wedge (\overline{z_2} \vee x_3 \vee z_3) \wedge \ldots \wedge (\overline{z_{k-2}} \vee x_{k-1} \vee x_k)\,,$$

wobei $z_2, \ldots, z_{k-2}$ neue Variable sind.
Es gibt eine Belegung für die z_i, so dass alle Klauseln erfüllt sind, gdw mindestens eines der Literale x_j wahr ist.
Die Größe der konstruierten Formel ist polynomiell in der Größe der Ausgangsklausel. Daraus folgt, dass die obige Umformung eine $\preceq_p$-Reduktion ist. □

Satz 231

3-COLORING *ist $\mathcal{NP}$-vollständig.*

Beweis:

Es ist wiederum klar, dass 3-COLORING $\in \mathcal{NP}$.

Beweis:

Es ist wiederum klar, dass 3-COLORING $\in \mathcal{NP}$.

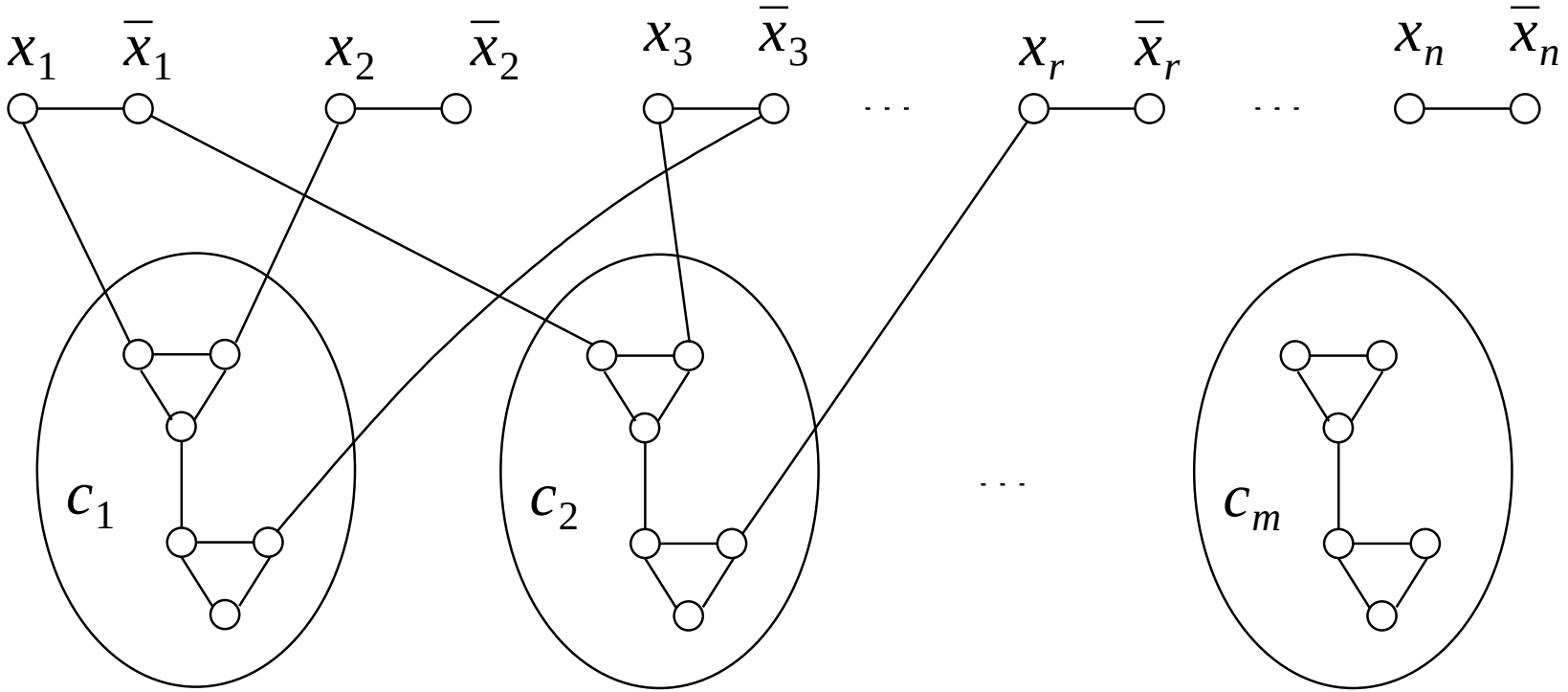
Um zu zeigen, dass 3-COLORING $\mathcal{NP}$-schwer ist, reduzieren wir 3SAT auf 3-COLORING.

Beweis:
Es ist wiederum klar, dass 3-COLORING $\in \mathcal{NP}$.

Um zu zeigen, dass 3-COLORING $\mathcal{NP}$-schwer ist, reduzieren wir 3SAT auf 3-COLORING.
Sei also eine boolesche Formel mit den Literalen $x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}$ und den Klauseln $c_1, \ldots, c_m$ gegeben.

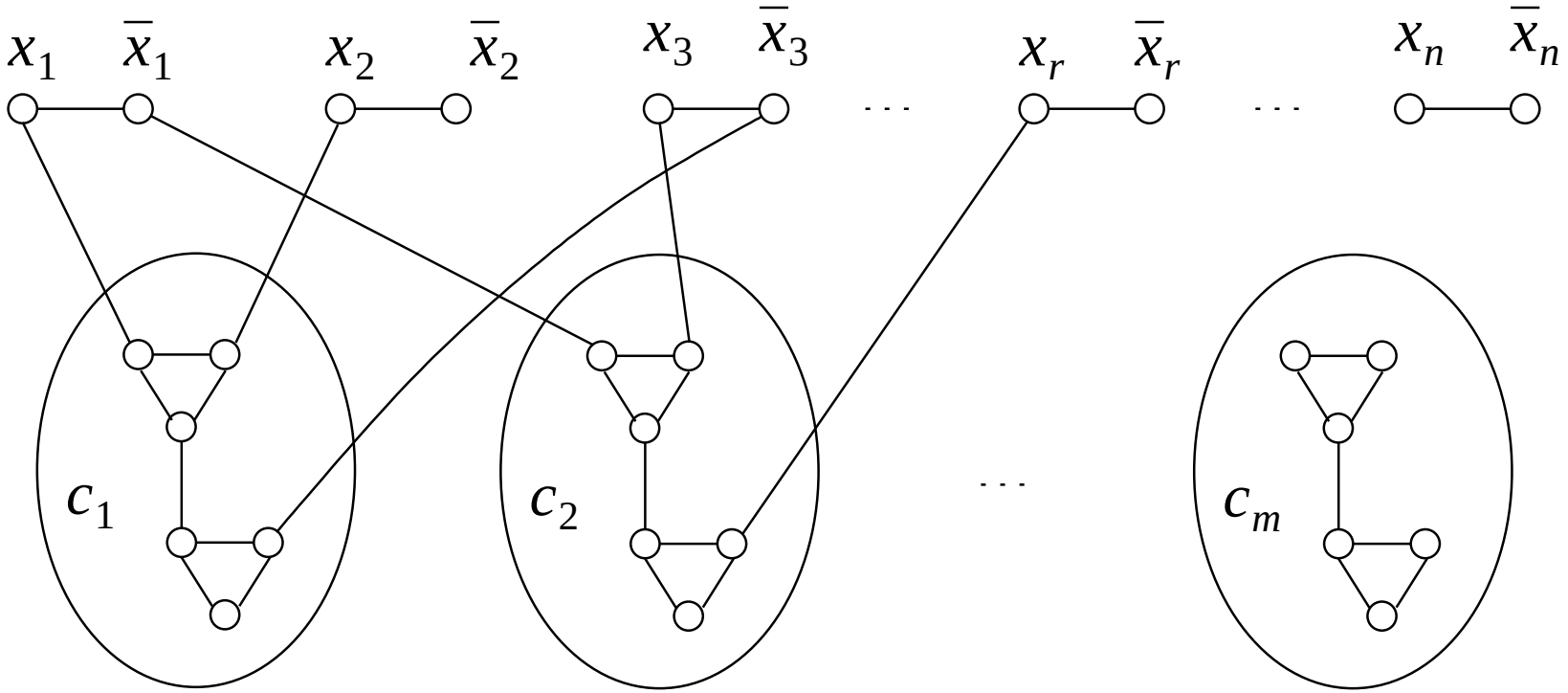
Beweis:

Wir konstruieren dazu folgenden Graphen:



Beweis:

Wir konstruieren dazu folgenden Graphen:



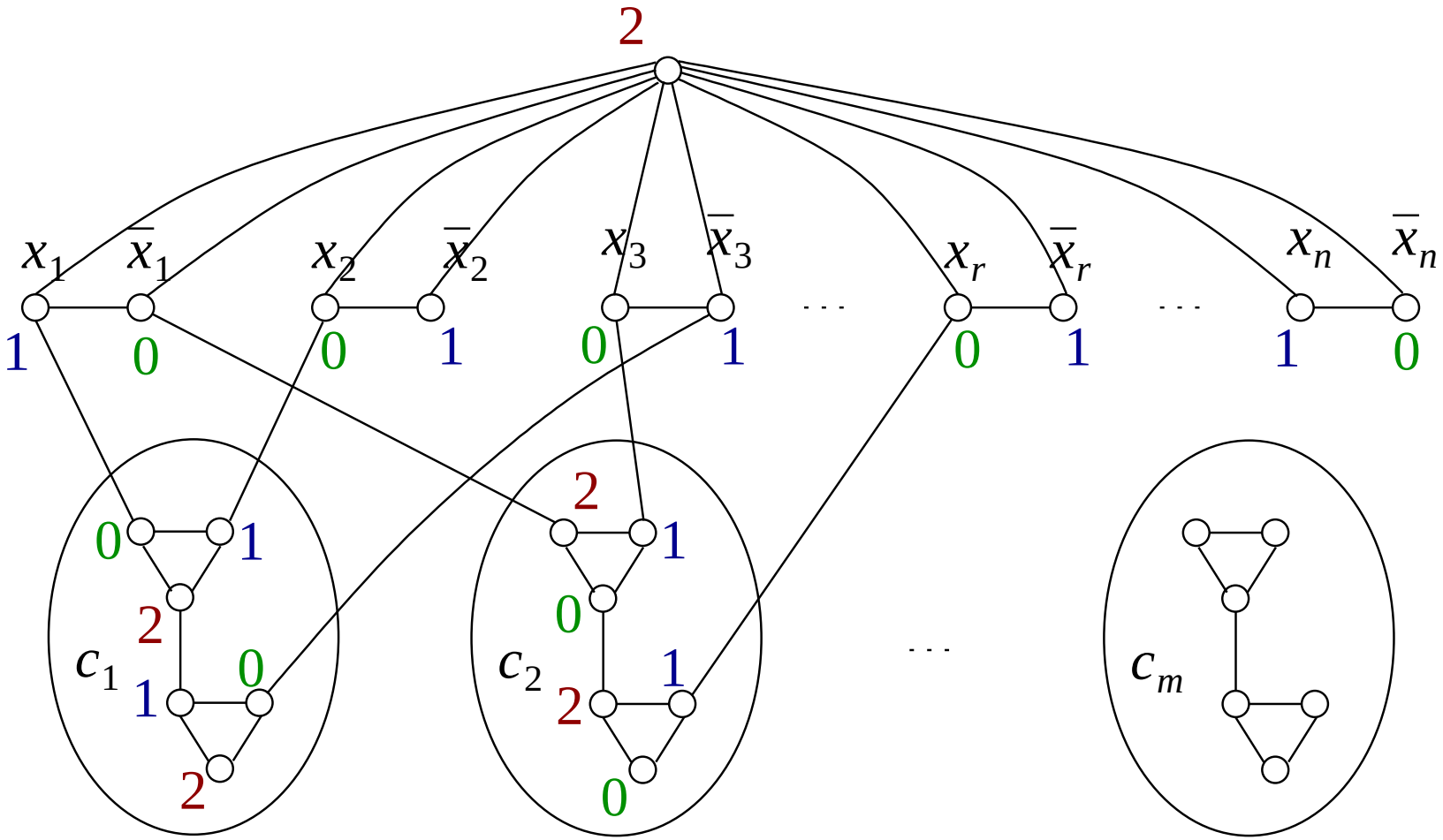
Die ersten beiden Klauseln sind hier

$$c_1 = x_1 \vee x_2 \vee \bar{x_3}$$

$$c_2 = \bar{x_1} \vee x_3 \vee x_r$$

Beweis:

Um zu erzwingen, dass alle Variablen nur mit Farben $\in \{0, 1\}$ gefärbt werden, verbinden wir alle "Literal"-Knoten mit einem zusätzlichen Knoten, der (per Vereinbarung) die Farbe 2 erhält.



Beweis:

Wie am Beispiel der Klausel c_2 ersichtlich, muss der unterste Knoten (der *Ausgang*) der Klausel die Farbe 0 erhalten, falls alle Literale der Klausel die Farbe 0 haben.

Falls nicht alle Literale einer Klausel die Farbe 0 haben, kann der Ausgang der Klausel, wie am Beispiel von c_1 ersichtlich, mit der Farbe 2 gefärbt werden.

Wir führen nun noch einen weiteren Hilfsknoten ein, der (per Vereinbarung) mit der Farbe 0 gefärbt wird (ansonsten werden die Farben wie angegeben umbenannt). Damit ergibt sich:

Beweis:

Wie am Beispiel der Klausel c_2 ersichtlich, muss der unterste Knoten (der *Ausgang*) der Klausel die Farbe 0 erhalten, falls alle Literale der Klausel die Farbe 0 haben.

Falls nicht alle Literale einer Klausel die Farbe 0 haben, kann der Ausgang der Klausel, wie am Beispiel von c_1 ersichtlich, mit der Farbe 2 gefärbt werden.

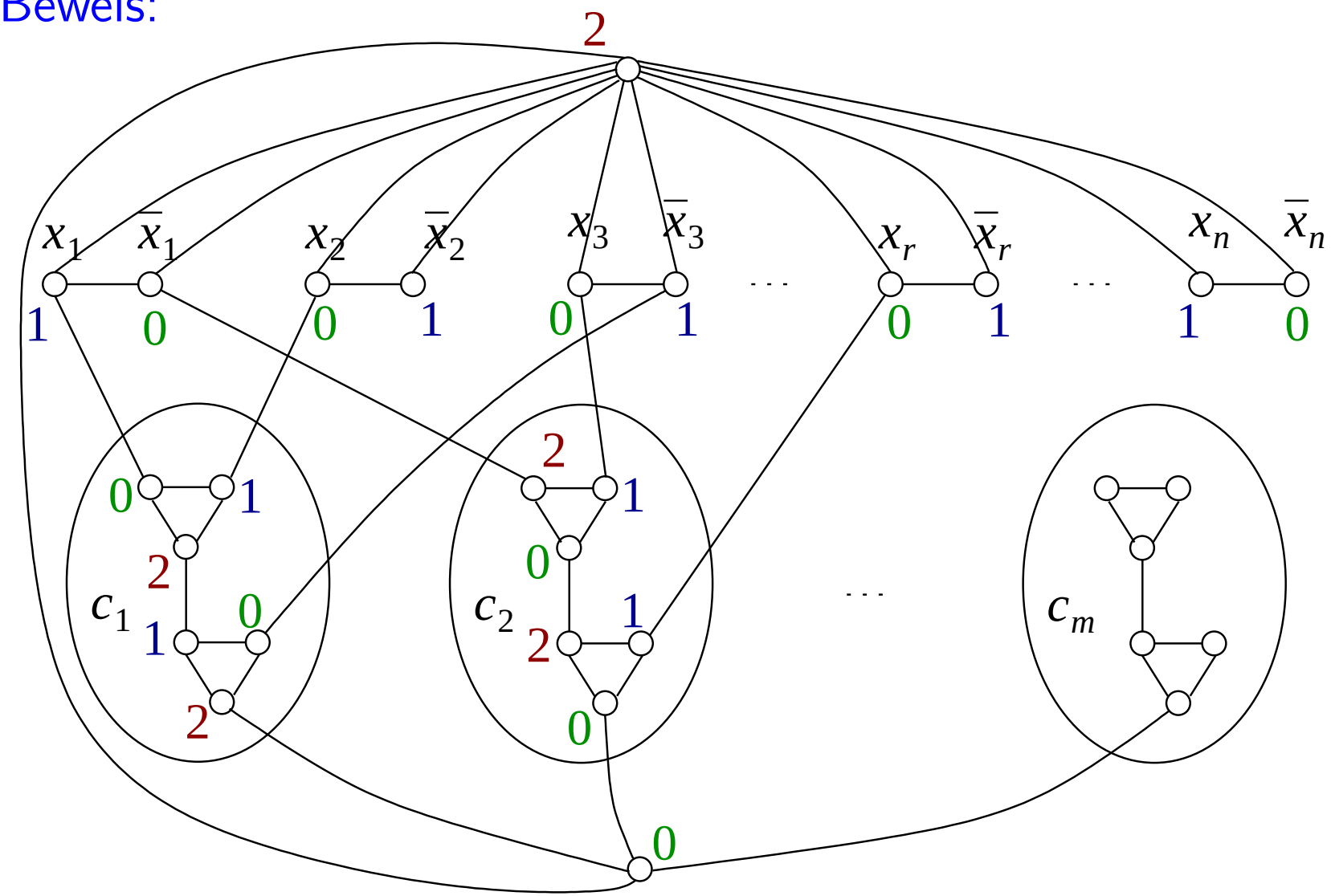
Wir führen nun noch einen weiteren Hilfsknoten ein, der (per Vereinbarung) mit der Farbe 0 gefärbt wird (ansonsten werden die Farben wie angegeben umbenannt). Damit ergibt sich:

Beweis:
Wie am Beispiel der Klausel c_2 ersichtlich, muss der unterste Knoten (der *Ausgang*) der Klausel die Farbe 0 erhalten, falls alle Literale der Klausel die Farbe 0 haben.

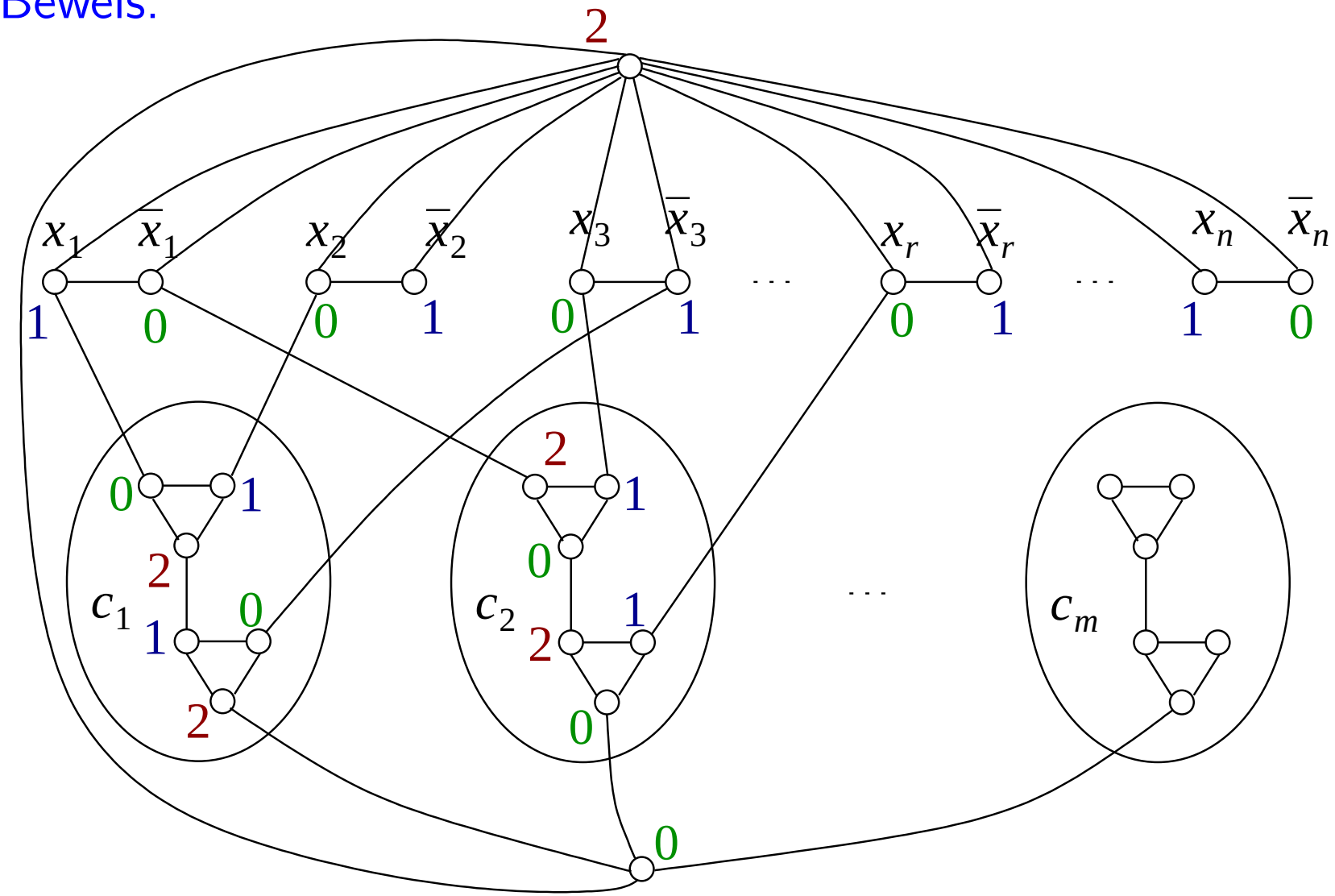
Falls nicht alle Literale einer Klausel die Farbe 0 haben, kann der Ausgang der Klausel, wie am Beispiel von c_1 ersichtlich, mit der Farbe 2 gefärbt werden.

Wir führen nun noch einen weiteren Hilfsknoten ein, der (per Vereinbarung) mit der Farbe 0 gefärbt wird (ansonsten werden die Farben wie angegeben umbenannt). Damit ergibt sich:

Beweis:


2
x_1
$\bar{x}_1$
x_2
$\bar{x}_2$
x_3
$\bar{x}_3$
x_r
$\bar{x}_r$
x_n
$\bar{x}_n$
1 0 0 1 0 1 0 1 1 0
c_1
c_2
c_m
0 1 2 0 1 2
2 1 0 1 2 0
0

Beweis:



und es gilt: F erfüllbar $\Longleftrightarrow$ G mit 3 Farben färbbar. □