

Vorlesung
Grundlagen der Theoretischen Informatik /
Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik



Sommersemester 2007

Dank

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

– *Bernhard Beckert, April 2007*

Frage

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Antwort

Es gibt mehrere Wege, ein schwerstes Problem zu definieren. Sie hängen davon ab, welchen **Begriff von Reduzierbarkeit** wir benutzen.

Für einen gegebenen Begriff von Reduzierbarkeit ist die Antwort: **Ja**. Solche Probleme werden **vollständig in der gegebenen Klasse** bezüglich des Begriffs der Reduzierbarkeit genannt.

Frage

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Antwort

Es gibt mehrere Wege, ein schwerstes Problem zu definieren. Sie hängen davon ab, welchen **Begriff von Reduzierbarkeit** wir benutzen.

Für einen gegebenen Begriff von Reduzierbarkeit ist die Antwort: **Ja**. Solche Probleme werden **vollständig in der gegebenen Klasse** bezüglich des Begriffs der Reduzierbarkeit genannt.

Definition 16.7 (Polynomial-Zeit-Reduzibilität)

Seien L_1, L_2 Sprachen.

Polynomial-Zeit: L_2 ist Polynomial-Zeit reduzibel auf L_1 , bezeichnet mit $L_2 \preceq_{\text{pol}} L_1$, wenn es eine **Polynomial-Zeit beschränkte DTM** gibt, die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so daß

$$w \in L_2 \text{ gdw } f(w) \in L_1$$

Definition 16.7 (Polynomial-Zeit-Reduzibilität)

Seien L_1, L_2 Sprachen.

Polynomial-Zeit: L_2 ist Polynomial-Zeit reduzibel auf L_1 , bezeichnet mit $L_2 \preceq_{\text{pol}} L_1$, wenn es eine **Polynomial-Zeit beschränkte DTM** gibt, die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so daß

$$w \in L_2 \text{ gdw } f(w) \in L_1$$

Lemma 16.8 (Polynomial-Zeit-Reduktionen)

① Sei L_2 Polynomial-Zeit-reduzibel auf L_1 . Dann gilt

L_2 ist in **NP** wenn L_1 in **NP** ist

L_2 ist in **P** wenn L_1 in **P** ist

② Die Komposition zweier Polynomial-Zeit-Reduktionen ist wieder eine Polynomial-Zeit-Reduktionen.

Lemma 16.8 (Polynomial-Zeit-Reduktionen)

1 Sei L_2 Polynomial-Zeit-reduzibel auf L_1 . Dann gilt

L_2 ist in **NP** wenn L_1 in **NP** ist

L_2 ist in **P** wenn L_1 in **P** ist

2 Die Komposition zweier Polynomial-Zeit-Reduktionen ist wieder eine Polynomial-Zeit-Reduktionen.

Lemma 16.8 (Polynomial-Zeit-Reduktionen)

① Sei L_2 Polynomial-Zeit-reduzibel auf L_1 . Dann gilt

L_2 ist in **NP** wenn L_1 in **NP** ist

L_2 ist in **P** wenn L_1 in **P** ist

② Die Komposition zweier Polynomial-Zeit-Reduktionen ist wieder eine Polynomial-Zeit-Reduktionen.

Theorem 16.9 (Charakterisierung von NP)

Eine Sprache L ist in **NP** genau dann wenn es eine Sprache L' in **P** und ein $k \geq 0$ gibt, so dass für alle $w \in \Sigma$ gilt:

$$w \in L \text{ gdw. es gibt ein } c : \langle w, c \rangle \in L' \text{ und } |c| < |w|^k.$$

c wird **Zeuge** (witness oder Zertifikat/certificate) von w in L genannt. Eine DTM, die die Sprache L' akzeptiert, wird **Prüfer** (verifier) von L genannt.

Wichtig:

Ein Entscheidungsproblem ist in **NP** genau dann wenn **jede Ja-Instanz ein kurzes Zertifikat** hat (d.h. seine Länge polynomial in der Länge der Eingabe ist), welche in polynomial-Zeit verifiziert werden kann.

Theorem 16.9 (Charakterisierung von NP)

Eine Sprache L ist in **NP** genau dann wenn es eine Sprache L' in **P** und ein $k \geq 0$ gibt, so dass für alle $w \in \Sigma$ gilt:

$$w \in L \text{ gdw. es gibt ein } c : \langle w, c \rangle \in L' \text{ und } |c| < |w|^k.$$

c wird **Zeuge** (witness oder Zertifikat/certificate) von w in L genannt. Eine DTM, die die Sprache L' akzeptiert, wird **Prüfer** (verifier) von L genannt.

Wichtig:

Ein Entscheidungsproblem ist in **NP** genau dann wenn **jede Ja-Instanz ein kurzes Zertifikat** hat (d.h. seine Länge polynomial in der Länge der Eingabe ist), welche in polynomial-Zeit verifiziert werden kann.

Teil VI

- 1 **Die Struktur von PSPACE**
- 2 Vollständige und harte Probleme
- 3 Beispiele

Teil VI



1 Die Struktur von PSPACE

2 Vollständige und harte Probleme

3 Beispiele

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - in NP ist ($L \in \mathbf{NP}$), und
 - NP-hart ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in \mathbf{NP} ist ($L \in \mathbf{NP}$), und
 - 2 NP-hart ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in **NP** ist ($L \in \mathbf{NP}$), und
 - 2 NP-hart ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in \mathbf{NP} ist ($L \in \mathbf{NP}$), und
 - 2 **NP-hart** ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in \mathbf{NP} ist ($L \in \mathbf{NP}$), und
 - 2 **NP-hart** ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in \mathbf{NP} ist ($L \in \mathbf{NP}$), und
 - 2 **NP-hart** ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
 - 1 in \mathbf{PSPACE} ist ($L \in \mathbf{PSPACE}$) und
 - 2 **PSPACE-hart** ist

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in \mathbf{NP} ist ($L \in \mathbf{NP}$), und
 - 2 **NP-hart** ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
 - 1 in \mathbf{PSPACE} ist ($L \in \mathbf{PSPACE}$) und
 - 2 **PSPACE-hart** ist

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in **NP** ist ($L \in \mathbf{NP}$), und
 - 2 **NP-hart** ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
 - 1 in **PSPACE** ist ($L \in \mathbf{PSPACE}$) und
 - 2 **PSPACE-hart** ist

Vollständige und harte Probleme

Definition 17.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
 - 1 in \mathbf{NP} ist ($L \in \mathbf{NP}$), und
 - 2 **NP-hart** ist

Definition 17.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
 - 1 in \mathbf{PSPACE} ist ($L \in \mathbf{PSPACE}$) und
 - 2 **PSPACE-hart** ist

Bemerkenswert

- Wenn gezeigt werden kann, daß auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist **P = NP**.
- Wenn **P** \neq **NP** gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Eine Million Euro für den, der das „P = NP“-Problem löst!
(Millenium Probleme)

Bemerkenswert

- Wenn gezeigt werden kann, daß auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist **P = NP**.
- Wenn **P \neq NP** gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Eine Million Euro für den, der das „P = NP“-Problem löst!
(Millenium Probleme)

Bemerkenswert

- Wenn gezeigt werden kann, daß auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist **P = NP**.
- Wenn **P \neq NP** gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Eine Million Euro für den, der das „P = NP“-Problem löst!
(Millenium Probleme)

Bemerkenswert

- Wenn gezeigt werden kann, daß auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist **P = NP**.
- Wenn **P \neq NP** gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

Eine Million Euro für den, der das „P = NP“-Problem löst!
(Millenium Probleme)

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

Finde bekanntermaßen **NP**-vollständige Sprache L' und reduzieren sie auf L :

$$L' \preceq L$$

Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq L$ dann auch auf L .

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

Vollständige und harte Probleme

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

Finde bekanntermaßen **NP**-vollständige Sprache L' und reduzieren sie auf L :

$$L' \preceq L$$

Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq L$ dann auch auf L .

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

Finde bekanntermaßen **NP**-vollständige Sprache L' und reduzieren sie auf L :

$$L' \preceq L$$

Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq L$ dann auch auf L .

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

Finde bekanntermaßen **NP**-vollständige Sprache L' und reduzieren sie auf L :

$$L' \preceq L$$

Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq L$ dann auch auf L .

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

P, PSPACE abgeschlossen unter Komplement

Alle Komplexitätsklassen, die mittels **deterministischer Turing-Maschinen** definiert sind, sind **abgeschlossen unter Komplement-Bildung**

Denn:

Wenn eine Sprache L dazu gehört, dann auch ihr Komplement (einfach die alte Maschine ausführen und die Ausgabe invertieren).

P, PSPACE abgeschlossen unter Komplement

Alle Komplexitätsklassen, die mittels **deterministischer Turing-Maschinen** definiert sind, sind **abgeschlossen unter Komplement-Bildung**

Denn:

Wenn eine Sprache L dazu gehört, dann auch ihr Komplement (einfach die alte Maschine ausführen und die Ausgabe invertieren).

Abgeschlossenheit von NP unter Komplement

Frage:

Ist **NP** abgeschlossen unter Komplementbildung?

Antwort:

Keiner weiß es!

Abgeschlossenheit von NP unter Komplement

Frage:

Ist **NP** abgeschlossen unter Komplementbildung?

Antwort:

Keiner weiß es!

Definition 17.3 (co-NP)

co-NP ist die Klasse der Sprachen deren Komplemente in **NP** liegen:

$$\mathbf{co-NP} = \{L \mid \bar{L} \in \mathbf{NP}\}$$

Die folgenden Beziehungen sind momentan noch unbekannt

- 1 $P \not\equiv NP$.
- 2 $NP \not\equiv \text{co-NP}$.
- 3 $P \not\equiv \text{PSPACE}$.
- 4 $NP \not\equiv \text{PSPACE}$.

Die folgenden Beziehungen sind momentan noch unbekannt

- 1 **$P \neq NP$.**
- 2 $NP \neq \text{co-NP}$.
- 3 $P \neq \text{PSPACE}$.
- 4 $NP \neq \text{PSPACE}$.

Die folgenden Beziehungen sind momentan noch unbekannt

- 1 **$P \neq NP$.**
- 2 **$NP \neq \text{co-NP}$.**
- 3 $P \neq \text{PSPACE}$.
- 4 $NP \neq \text{PSPACE}$.

Die folgenden Beziehungen sind momentan noch unbekannt

- 1 $P \neq NP$.
- 2 $NP \neq \text{co-NP}$.
- 3 $P \neq \text{PSPACE}$.
- 4 $NP \neq \text{PSPACE}$.

Die folgenden Beziehungen sind momentan noch unbekannt

- 1 $P \not\equiv NP$.
- 2 $NP \not\equiv \text{co-NP}$.
- 3 $P \not\equiv \text{PSPACE}$.
- 4 $NP \not\equiv \text{PSPACE}$.

Teil VI

- 1 Die Struktur von PSPACE
- 2 Vollständige und harte Probleme**
- 3 Beispiele

Teil VI



- 1 Die Struktur von PSPACE
- 2 Vollständige und harte Probleme
- 3 Beispiele**

Beispiel 18.1 (NP vollständige Probleme)

- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Gibt es in einem Graphen eine Clique der Größe k ? (**Clique of size k**)
- Ist ein Graph mit drei Farben zu färben? (**3-colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x ? (**Subset Sum**)

Beispiel 18.1 (NP vollständige Probleme)

- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Gibt es in einem Graphen eine Clique der Größe k ? (**Clique of size k**)
- Ist ein Graph mit drei Farben zu färben? (**3-colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x ? (**Subset Sum**)

Beispiel 18.1 (NP vollständige Probleme)

- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Gibt es in einem Graphen eine Clique der Größe k ? (**Clique of size k**)
- Ist ein Graph mit drei Farben zu färben? (**3-colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x ? (**Subset Sum**)

Beispiel 18.1 (NP vollständige Probleme)

- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Gibt es in einem Graphen eine Clique der Größe k ? (**Clique of size k**)
- Ist ein Graph mit drei Farben zu färben? (**3-colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x ? (**Subset Sum**)

Beispiel 18.1 (NP vollständige Probleme)

- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Gibt es in einem Graphen eine Clique der Größe k ? (**Clique of size k**)
- Ist ein Graph mit drei Farben zu färben? (**3-colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x ? (**Subset Sum**)

Definition 18.2 (Hamilton Circle)

Hamilton-Kreis:

Weg entlang der Kanten in einem Graphen, der jeden Knoten genau einmal besucht

$L_{\text{Ham}_{\text{undir}}}$: Die Sprache, die aus allen ungerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

$L_{\text{Ham}_{\text{dir}}}$: Die Sprache, die aus allen gerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

Definition 18.2 (Hamilton Circle)

Hamilton-Kreis:

Weg entlang der Kanten in einem Graphen, der jeden Knoten genau einmal besucht

$L_{\text{Ham}_{\text{undir}}}$: Die Sprache, die aus allen ungerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

$L_{\text{Ham}_{\text{dir}}}$: Die Sprache, die aus allen gerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

Definition 18.2 (Hamilton Circle)

Hamilton-Kreis:

Weg entlang der Kanten in einem Graphen, der jeden Knoten genau einmal besucht

$L_{\text{Ham}_{\text{undir}}}$: Die Sprache, die aus allen ungerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

$L_{\text{Ham}_{\text{dir}}}$: Die Sprache, die aus allen gerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

Definition 18.3 (Maximale Clique: L_{Clique_k})

Eine **Clique** in einem Graphen ist ein **vollständiger Teilgraph von G** .

Für $k \in \mathbb{N}$:

L_{Clique_k} Die Sprache, die aus allen ungerichteten Graphen besteht, die eine Clique der Größe k enthalten.

Definition 18.3 (Maximale Clique: L_{Clique_k})

Eine **Clique** in einem Graphen ist ein **vollständiger Teilgraph von G** .

Für $k \in \mathbb{N}$:

L_{Clique_k} Die Sprache, die aus allen ungerichteten Graphen besteht, die eine Clique der Größe k enthalten.

Definition 18.4 (k -colorability: $L_{\text{Color}_{\leq k}}$)

Ein (ungerichteter) Graph heißt **k -färbbar**, falls jeder Knoten mit einer von k Farben so gefärbt werden kann, daß benachbarte Knoten verschiedene Farben haben.

Für $k \in \mathbb{N}$:

$L_{\text{Color}_{\leq k}}$ Die Sprache, die aus allen ungerichteten, mit höchstens k Farben färbbaren Graphen besteht.

Definition 18.4 (k -colorability: $L_{\text{Color}_{\leq k}}$)

Ein (ungerichteter) Graph heißt **k -färbbar**, falls jeder Knoten mit einer von k Farben so gefärbt werden kann, daß benachbarte Knoten verschiedene Farben haben.

Für $k \in \mathbb{N}$:

$L_{\text{Color}_{\leq k}}$ Die Sprache, die aus allen ungerichteten, mit höchstens k Farben färbbaren Graphen besteht.

Definition 18.5 (SAT, k -CNF, k -DNF)

DNF: Eine Formel ist in **disjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \wedge \dots \wedge l_{1n_1}) \vee \dots \vee (l_{m1} \wedge \dots \wedge l_{mn_m})$$

CNF: Eine Formel ist in **konjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \vee \dots \vee l_{1n_1}) \wedge \dots \wedge (l_{m1} \vee \dots \vee l_{mn_m})$$

Definition 18.5 (SAT, k -CNF, k -DNF)

DNF: Eine Formel ist in **disjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \wedge \dots \wedge l_{1n_1}) \vee \dots \vee (l_{m1} \wedge \dots \wedge l_{mn_m})$$

CNF: Eine Formel ist in **konjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \vee \dots \vee l_{1n_1}) \wedge \dots \wedge (l_{m1} \vee \dots \vee l_{mn_m})$$

Definition (Fortsetzung)

k -DNF: Eine Formel ist in k -DNF wenn sie in DNF ist und jede ihrer Konjunktionen genau k Literale hat.

k -CNF: Eine Formel ist in k -CNF wenn sie in CNF ist und jede ihrer Disjunktion genau k Literale hat.

Definition (Fortsetzung)

k -DNF: Eine Formel ist in k -DNF wenn sie in DNF ist und jede ihrer Konjunktionen genau k Literale hat.

k -CNF: Eine Formel ist in k -CNF wenn sie in CNF ist und jede ihrer Disjunktion genau k Literale hat.

NP-vollständige Probleme

Theorem 18.6 (NP-vollständige Probleme)

Die folgenden Probleme liegen in **NP** und sind **NP-vollständig**:

- L_{sat}
- CNF
- k - CNF für $k \geq 3$

Theorem 18.7 (Probleme in P)

Die folgenden Probleme liegen in **P**:

- DNF
- k - DNF für alle k
- 2 - CNF

Theorem 18.6 (NP-vollständige Probleme)

Die folgenden Probleme liegen in **NP** und sind **NP-vollständig**:

- L_{sat}
- CNF
- k - CNF für $k \geq 3$

Theorem 18.7 (Probleme in P)

Die folgenden Probleme liegen in **P**:

- DNF
- k - DNF für alle k
- 2 - CNF

Einige Beispiel-Reduktionen

- $L_{\text{CNF-SAT}} \preceq_{\text{pol}} L_{\text{Clique}_{\leq k}}$,
- $L_{\text{Ham}_{\text{dir}}} \preceq_{\text{pol}} L_{\text{Ham}_{\text{undir}}}$,
- $L_{\text{Ham}_{\text{undir}}} \preceq_{\text{pol}} L_{\text{Ham}_{\text{cost} \leq k}}, L_{\text{Clique}_k}$,
- $L_{\text{SAT}} \preceq_{\text{pol}} L_{\text{3-CNF}}$,
- $L_{\text{SAT}} \preceq_{\text{pol}} L_{\text{CNF-SAT}}$,
- $L_{\text{3-CNF}} \preceq_{\text{pol}} L_{\text{Color}_{\leq k}}$.

Beispiel 18.8 (CNF-SAT \preceq_{pol} Clique $_{\leq k}$)

Gegeben eine Instanz von CNF

(eine Konjunktion von Klauseln $C_1 \wedge C_2 \wedge \dots \wedge C_k$)

Wir konstruieren daraus einen Graphen:

Knoten: die Paare (x, i) , so dass x ein Literal ist, dass in der Klausel C_i vorkommt.

Kanten: Es gibt eine Kante zwischen (x, i) und (y, j) falls:

- (1) $i \neq j$, und
- (2) x, y sind nicht komplementär.

Es gilt dann:

Die CNF-Formel ist erfüllbar genau dann,
wenn der zugeordnete Graph eine Clique der Größe k hat.

NP-vollständige Probleme

Beispiel 18.8 (CNF-SAT \preceq_{pol} Clique $_{\leq k}$)

Gegeben eine Instanz von CNF

(eine Konjunktion von Klauseln $C_1 \wedge C_2 \wedge \dots \wedge C_k$)

Wir konstruieren daraus einen Graphen:

Knoten: die Paare (x, i) , so dass x ein Literal ist, dass in der Klausel C_i vorkommt.

Kanten: Es gibt eine Kante zwischen (x, i) und (y, j) falls:

- (1) $i \neq j$, und
- (2) x, y sind nicht komplementär.

Es gilt dann:

Die CNF-Formel ist erfüllbar genau dann,
wenn der zugeordnete Graph eine Clique der Größe k hat.

NP-vollständige Probleme

Beispiel 18.8 (CNF-SAT \preceq_{pol} Clique $_{\leq k}$)

Gegeben eine Instanz von CNF

(eine Konjunktion von Klauseln $C_1 \wedge C_2 \wedge \dots \wedge C_k$)

Wir konstruieren daraus einen Graphen:

Knoten: die Paare (x, i) , so dass x ein Literal ist, dass in der Klausel C_i vorkommt.

Kanten: Es gibt eine Kante zwischen (x, i) und (y, j) falls:

- (1) $i \neq j$, und
- (2) x, y sind nicht komplementär.

Es gilt dann:

Die CNF-Formel ist erfüllbar genau dann,
wenn der zugeordnete Graph eine Clique der Größe k hat.

Beispiel 18.8 (CNF-SAT \preceq_{pol} Clique $_{\leq k}$)

Gegeben eine Instanz von CNF

(eine Konjunktion von Klauseln $C_1 \wedge C_2 \wedge \dots \wedge C_k$)

Wir konstruieren daraus einen Graphen:

Knoten: die Paare (x, i) , so dass x ein Literal ist, dass in der Klausel C_i vorkommt.

Kanten: Es gibt eine Kante zwischen (x, i) und (y, j) falls:

- (1) $i \neq j$, und
- (2) x, y sind nicht komplementär.

Es gilt dann:

**Die CNF-Formel ist erfüllbar genau dann,
wenn der zugeordnete Graph eine Clique der Größe k hat.**