Informatik III - Tutorium XIII & XV (SR -107)

Tut Nr. 6 – Üb5, Einführung in die Komplexitätstheorie

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
ITI Wagner

5. Dezember 2007

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

1 Auftakt

Inhaltsverzeichnis

1. Auftakt
2. Themen
   - Übungsblatt 5
   - Problemformulierungen
   - Kodierungsschemata für Sprachen und Probleme
   - Komplexitätsklasse $\mathcal{P}$
   - Komplexitätsklasse $\mathcal{NP}$

Inhaltsverzeichnis

1. Auftakt
2. Themen
   - Übungsblatt 5
   - Problemformulierungen
   - Kodierungsschemata für Sprachen und Probleme
   - Komplexitätsklasse $\mathcal{P}$
   - Komplexitätsklasse $\mathcal{NP}$
3. Abspann

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 13: Mittwochs 8:00 Uhr - Raum -107
Tutorium 15: Mittwochs 9:45 Uhr - Raum -107

Übungsblattabgabe Donnerstag.

Aufgabe 2

Zeige, dass die Sprache
$L_{\emptyset}^{T} = \{\langle M \rangle : M \text{ Turingmaschine}, L(M) = \emptyset\}$ nicht entscheidbar ist.

Aufgabe 2

Zeige, dass die Sprache
$L_\emptyset^T = \{\langle M \rangle : M \text{ Turingmaschine}, L(M) = \emptyset\}$ nicht entscheidbar ist.

Lösung:
Für $L = \emptyset$ ist nach Korollar 3.6 S.40 L entscheidbar, weil charakteristische Funktion $\chi_L = 0$ berechenbar ist.
Nach dem Satz von Rice 3.16 S.46 mit $S = \{\chi_L\}$ ist somit die Sprache $L_\emptyset^T$ nicht entscheidbar.

Aufgabe 3

Zeige: Eine Sprache L ist genau dann entscheidbar, wenn L und L^c semientscheidbar sind. (Benutze hierzu, dass eine 2-Band Turingmaschine durch eine 1-Band Turingmaschine simuliert werden kann).

Aufgabe 3

Zeige: Eine Sprache L ist genau dann entscheidbar, wenn L und L^c semientscheidbar sind. (Benutze hierzu, dass eine 2-Band Turingmaschine durch eine 1-Band Turingmaschine simuliert werden kann).

Sei $L = L(M_1)$ und $L^c = L(M_2)$.
Nun werden wir eine TM M konstruieren, die L erkennt:
Wir simulieren parallel auf dem Band 1 der TM M' die TM M_1 und auf dem Band 2 der TM M' die TM M_2.
Diese TM M' können wir nach 3.2 S.42 in eine 1-Band-TM M konvertieren.
Ist die Eingabe w in L enthalten, dann wird M_1 akzeptieren. Ist die Eingabe w in L^c enthalten, dann wird M_2 akzeptieren.
Unsere TM M soll in beiden genannten Fällen stoppen und w genau dann akzeptieren, wenn M_1 akzeptiert hat.

Aufgabe 4

Sei L eine semi-entscheidbare Sprache, die nicht entscheidbar ist. Ist die Sprache $L' = \{0w : w \in L\} \cup \{1w : w \notin L\}$ entscheidbar, semi-entscheidbar oder keins von beiden?
Hinweis: Nimm an, L' sei semi-entscheidbar, und folgere daraus, dass dann L entscheidbar ist. Man kann zusätzlich das Ergebnis von Aufgabe 3 benutzen.

Aufgabe 4

Sei L eine semi-entscheidbare Sprache, die nicht entscheidbar ist. Ist die Sprache $L' = \{0w : w \in L\} \cup \{1w : w \notin L\}$ entscheidbar, semi-entscheidbar oder keins von beiden?
Hinweis: Nimm an, L' sei semi-entscheidbar, und folgere daraus, dass dann L entscheidbar ist. Man kann zusätzlich das Ergebnis von Aufgabe 3 benutzen.

Lösung:
Weil L semi-entscheidbar ist, aber nicht entscheidbar, ist L^c nicht semi-entscheidbar.
Wäre jetzt L' semi-entscheidbar mit zugehöriger TM M', so könnten wir M' so zu einer TM M^c modifizieren, sodass M^c die Sprache L^c semi-entscheidet: Wir ändern jeden Input w zu $1w$ und simulieren dann M'. Also ist L' nicht semi-entscheidbar und damit auch nicht entscheidbar.

Aufgabe 5

a) Die Menge der semi-entscheidbaren Sprachen ist unter Vereinigung und Schnitt abgeschlossen.

Aufgabe 5

a) Die Menge der semi-entscheidbaren Sprachen ist unter Vereinigung und Schnitt abgeschlossen.

b) Die Menge der semi-entscheidbaren Sprachen ist unter Komplementbildung nicht abgeschlossen.

Aufgabe 5

a) Die Menge der semi-entscheidbaren Sprachen ist unter Vereinigung und Schnitt abgeschlossen.

b) Die Menge der semi-entscheidbaren Sprachen ist unter Komplementbildung nicht abgeschlossen.

c) Die Menge der entscheidbaren Sprachen ist unter Komplementbildung, Vereinigung und Schnitt abgeschlossen.

Aufgabe 5

a) Die Menge der semi-entscheidbaren Sprachen ist unter Vereinigung und Schnitt abgeschlossen.

b) Die Menge der semi-entscheidbaren Sprachen ist unter Komplementbildung nicht abgeschlossen.

c) Die Menge der entscheidbaren Sprachen ist unter Komplementbildung, Vereinigung und Schnitt abgeschlossen.

d) Warum ist Abgeschlossenheit bezüglich Komplementbildung bei semi-entscheidbaren Sprachen eine wünschenswerte Eigenschaft?

Aufgabe 5a

Seien L_1, L_2 semi-entscheidbar. M_1, M_2 TM, die L_1 bzw. L_2 akz.

Vereinigung:

Es seien Q_1, δ_1 bzw. Q_2, δ_2 die Zustände und Übergangsfunktionen von M_1 und M_2. Wir konstruieren eine 2-Band-TM M, die $L_1 \cup L_2$ semi-entscheidet.

M kopiert zuerst die Eingabe auf Band 2. Die Zustandsmenge von M ist $Q_1 \times Q_2$. Die Übergangsfunktion arbeitet komponentenweise auf den entsprechenden Bändern für die entsprechende simulierte TM. Die TM endet in einem akzeptierenden Zustand, wenn eine Komponente in einem akzeptierenden Zustand von M_1 bzw. M_2 ist.

Aufgabe 5a

Seien L_1, L_2 semi-entscheidbar. M_1, M_2 TM, die L_1 bzw. L_2 akz.

Vereinigung:

Es seien Q_1, δ_1 bzw. Q_2, δ_2 die Zustände und Übergangsfunktionen von M_1 und M_2. Wir konstruieren eine 2-Band-TM M, die $L_1 \cup L_2$ semi-entscheidet.

M kopiert zuerst die Eingabe auf Band 2. Die Zustandsmenge von M ist $Q_1 \times Q_2$. Die Übergangsfunktion arbeitet komponentenweise auf den entsprechenden Bändern für die entsprechende simulierte TM. Die TM endet in einem akzeptierenden Zustand, wenn eine Komponente in einem akzeptierenden Zustand von M_1 bzw. M_2 ist.

Schnitt:

Wie Vereinigung, allerdings wird nur akzeptiert, wenn sowohl M_1 als auch M_2 akzeptieren.

Aufgabe 5b

Gegenbeispiel:
Die Universelle Sprache L_u ist semi-entscheidbar, aber nicht entscheidbar.
Angenommen L_u^c ist semi-sntscheidbar, dann ist L_u entscheidbar.
Widerspruch!

Aufgabe 5c

Seien L, L_1, L_2 entscheidbar.

Komplement:

Sei M eine stets haltende TM, die genau die Wörter aus L akzeptiert. Wir konstruieren eine TM M^* aus M, indem wir einen Zustand q^* hinzufügen, in den gewechselt wird, falls in M in einem nicht akzeptierenden Zustand gestoppt wird. Der einzige akzeptierende Zustand in M^* soll q^* sein. Damit akzeptiert M^* genau L^c.

Aufgabe 5c

Vereinigung:
Es seien M_1 und M_2 stets haltende Turingmaschinen, die L_1 bzw. L_2 akzeptieren. Wir konstruieren eine stets haltende TM M, die $L_1 \cup L_2$ akzeptiert. M arbeitet auf 2 Bändern und schreibt zunächst eine Kopie der Eingabe auf Band 2. Dann wird M_1 auf Band 1 simuliert. Falls M_1 die Eingabe akzeptiert, akzeptiert auch M. Ansonsten wird M_2 auf Band 2 simuliert. Falls nun M_2 die Eingabe akzeptiert, akzeptiert auch M.

Aufgabe 5c

Vereinigung:
Es seien M_1 und M_2 stets haltende Turingmaschinen, die L_1 bzw. L_2 akzeptieren. Wir konstruieren eine stets haltende TM M, die $L_1 \cup L_2$ akzeptiert. M arbeitet auf 2 Bändern und schreibt zunächst eine Kopie der Eingabe auf Band 2. Dann wird M_1 auf Band 1 simuliert. Falls M_1 die Eingabe akzeptiert, akzeptiert auch M. Ansonsten wird M_2 auf Band 2 simuliert. Falls nun M_2 die Eingabe akzeptiert, akzeptiert auch M.
Schnitt:
Mit DeMorgan folgt: $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$

Aufgabe 5d

Weil die Sprache dann entscheidbar ist.

Definition: Optimierungsproblem

Gesucht ist eine optimale Lösung für eine Probleminstanz.
Bsp.: die kürzeste Rundreise

Definition: Optimierungsproblem

Gesucht ist eine optimale Lösung für eine Probleminstanz.
Bsp.: die kürzeste Rundreise

Definition: Optimalwertproblem

Gesucht ist der optimale Wert für eine Probleminstanz.
Bsp.: die Länge der kürzesten Rundreise

Definition: Optimierungsproblem

Gesucht ist eine optimale Lösung für eine Probleminstanz.
Bsp.: die kürzeste Rundreise

Definition: Optimalwertproblem

Gesucht ist der optimale Wert für eine Probleminstanz.
Bsp.: die Länge der kürzesten Rundreise

Definition: Entscheidungsproblem

Gibt es eine Problemlösung besser als eine bestimmte Schranke?
Antwort: Ja oder Nein.

Optimierungs- vs. Entscheidungsproblem

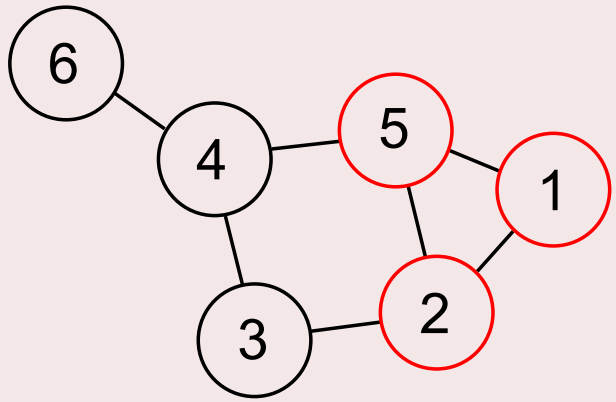
Offensichtlich kann man mit dem Entscheidungsproblem auch das Optimierungsproblem lösen, indem man die Schranke je nach Antwort variiert (Keine Lösung $\rightarrow$ höhere Schranke, Lösung gefunden $\rightarrow$ Schranke senken, um das Optimum zu finden.

Definition: CLIQUE

In einem ungerichteten Graphen $G = (V, E)$ beildet die Knotenmenge $V' \subseteq V$ eine Clique, wenn für alle $v, v' \in V'$ gilt $\{v, v'\} \in E$.

Definition: CLIQUE

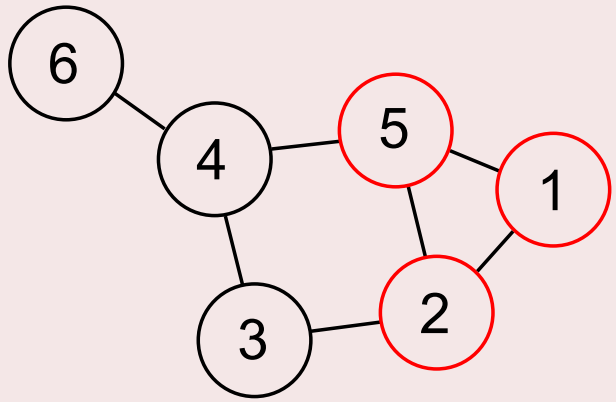
In einem ungerichteten Graphen $G = (V, E)$ beildet die Knotenmenge $V' \subseteq V$ eine Clique, wenn für alle $v, v' \in V'$ gilt $\{v, v'\} \in E$.



Beispiel:

Definition: CLIQUE

In einem ungerichteten Graphen $G = (V, E)$ beildet die Knotenmenge $V' \subseteq V$ eine Clique, wenn für alle $v, v' \in V'$ gilt $\{v, v'\} \in E$.



Beispiel:

Aufgabe 1

Formulieren Sie **CLIQUE** als Opimierungs-, Optimalwert- und Entscheidungsproblem.

Problem / Problembeispiel

Ein **Problem** Π ist gegeben durch:

1. eine allgemeine Beschreibung aller vorkommenden Parameter
2. eine genaue Beschreibung der Eigenschaften, die die Lösung haben soll.

Problem / Problembeispiel

Ein **Problem** Π ist gegeben durch:

1. eine allgemeine Beschreibung aller vorkommenden Parameter
2. eine genaue Beschreibung der Eigenschaften, die die Lösung haben soll.

Nicht zwingend: wie wird das Problem gelöst?

Problem / Problembeispiel

Ein **Problem** Π ist gegeben durch:

1. eine allgemeine Beschreibung aller vorkommenden Parameter
2. eine genaue Beschreibung der Eigenschaften, die die Lösung haben soll.

Nicht zwingend: wie wird das Problem gelöst?

Ein **Problembeispiel** I **(Instanz)** von Π erhalten wir, indem wir die Parameter von Π festlegen.

Definition: Kodierungsschemata

Ein **Kodierungsschema** s ordnet jedem **Problembeispiel** I eines **Problems** Π eine Zeichenkette oder *Kodierung* über dem Alphabet Σ zu. Die **Inputlänge** $|s(I)|$ eines Problembeispiels ist die Anzahl der Symbole seiner Kodierung. Dabei gibt es natürlich verschiedene Kodierungsschemata für ein bestimmtes Problem.

Definition: Kodierungsschemata

Ein **Kodierungsschema** s ordnet jedem **Problembeispiel** I eines **Problems** Π eine Zeichenkette oder *Kodierung* über dem Alphabet Σ zu. Die **Inputlänge** $|s(I)|$ eines Problembeispiels ist die Anzahl der Symbole seiner Kodierung. Dabei gibt es natürlich verschiedene Kodierungsschemata für ein bestimmtes Problem.

Kodierungsschema $\simeq$ Festlegung der Art der Parameter für ein Problem

Definition: Äquivalenz von Kodierungsschemata

Zwei Kodierungsschemata s_1 , s_2 heißen **äquivalent** bezüglich eines Problems Π, falls es Polynome p_1 , p_2 gibt, so dass gilt:

$$(|s_1(I)| = n \Rightarrow |s_2(I)| \leq p_2(n))$$

Definition: Äquivalenz von Kodierungsschemata

Zwei Kodierungsschemata s_1 , s_2 heißen **äquivalent** bezüglich eines Problems Π, falls es Polynome p_1 , p_2 gibt, so dass gilt:

$$(|s_1(I)| = n \Rightarrow |s_2(I)| \leq p_2(n))$$

und

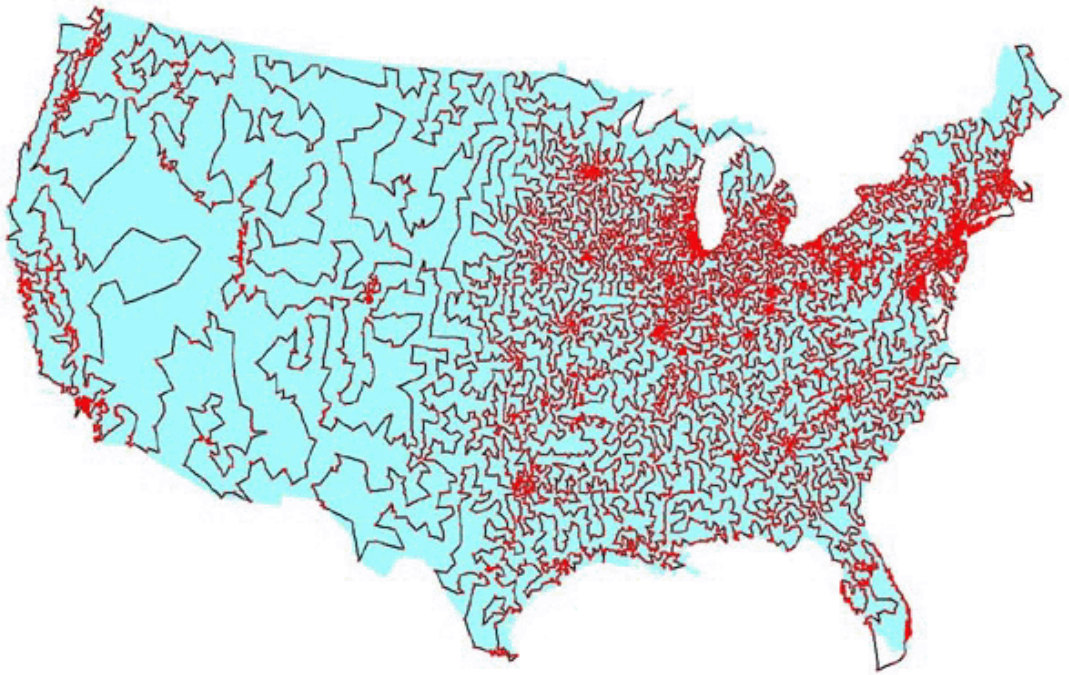
$$(|s_2(I)| = m \Rightarrow |s_1(I)| \leq p_1(m))$$

für alle Problembeispiele I von Π.

Definition: TRAVELING SALESMAN PROBLEM (TSP)

Gegeben sind n Orte und die Kosten $c(i,j) \in \mathcal{N}$, um von i nach j zu reisen. Eine Rundreise (Hamiltonkreis, eine Tour) ist durch eine Permutation π auf $\{1,...,n\}$ gegeben, ihre Kosten betragen $c(\pi(1),\pi(2))+c(\pi(2),\pi(3))+...+c(\pi(n-1),\pi(n))+c(\pi(n),\pi(1))$. Berechne die billigste Rundreise.

Beispiel:



Rundreise durch die USA mit Besuch von jedem Ort, mit mehr als 500 Einwohnern.

Aufgabe 2

Geben Sie ein Kodierungsschema für TSP explizit an.

Definition: $\mathcal{P}$

Die Klasse $\mathcal{P}$ ist die Menge aller Sprachen L (Probleme), für die eine **deterministische** Turing-Maschine existiert, deren Zeitkomplexitätsfunktion $T_{\mathcal{M}}$ polynomial ist, d.h. es existiert ein Polynom p mit

$$T_{\mathcal{M}}(n) \leq p(n).$$

Definition: $\mathcal{P}$

Die Klasse $\mathcal{P}$ ist die Menge aller Sprachen L (Probleme), für die eine **deterministische** Turing-Maschine existiert, deren Zeitkomplexitätsfunktion $T_{\mathcal{M}}$ polynomial ist, d.h. es existiert ein Polynom p mit

$$T_{\mathcal{M}}(n) \leq p(n).$$

Bemerkung: $T_{\mathcal{M}}(n)$ ist der "worst case" an Übergängen bis $\mathcal{M}$ bei Eingabe eines x der Länge n einen Endzustand erreicht.

Aufgabe 3

Zeige, dass $\mathcal{P}$ unter Vereinigung abgeschlossen ist:
$A, B \in \mathcal{P} \Rightarrow \mathcal{A} \cup \mathcal{B} \in \mathcal{P}$.

Aufgabe 3

Zeige, dass $\mathcal{P}$ unter Vereinigung abgeschlossen ist:
$A, B \in \mathcal{P} \Rightarrow \mathcal{A} \cup \mathcal{B} \in \mathcal{P}$.

Lösung:
Da $A, B \in \mathcal{P}$ gibt es zwei Turing-Maschinen M_A und M_B, die A bzw. B in polynomieller Zeit entscheiden. Wir konstruieren eine Turing-Maschine M, die angesetzt auf w zuerst M_A auf w ansetzt und dann M_B. M erreiche genau dann einen Endzustand, wenn dies M_A oder M_B tut. Die Laufzeit von M ist im Wesentlichen die Summe der polynomiellen Laufzeiten von M_A und M_B. Da die Summe von zwei Polynomen ebenfalls ein Polynom ist, entscheidet M in polynomieller Zeit $A \cup B$. Daraus folgt $A \cup B \in \mathcal{P}$.

Definition: $\mathcal{NP}$

Die Klasse $\mathcal{NP}$ ist die Menge aller Sprachen L, für die es eine **nichtdeterministische** Turing-Maschine gibt, deren Zeitkomplexitätsfunktion $T_{\mathcal{M}}$ **polynomial** beschränkt ist.

($\mathcal{NP}$ steht für **nichtdeterministisch polynomial**).

Definition: $\mathcal{NP}$

Die Klasse $\mathcal{NP}$ ist die Menge aller Sprachen L, für die es eine **nichtdeterministische** Turing-Maschine gibt, deren Zeitkomplexitätsfunktion $T_{\mathcal{M}}$ **polynomial** beschränkt ist.

($\mathcal{NP}$ steht für **nichtdeterministisch polynomial**).

Bemerkung: $T_{\mathcal{M}}(n)$ ist der "worst case" an Zeit, die $\mathcal{M}$ bei Eingabe eines x der Länge n benötigt, x zu akzeptieren.

Nichtdeterministische Turing-Maschine

Nichtdeterministische Turing-Maschine

Eine nichtdeterministische Turing-Maschine (**NTM**) ist genauso definiert wie eine DTM, mit Erweiterungen:

1. ε-Übergänge in δ erlaubt

Nichtdeterministische Turing-Maschine

Eine nichtdeterministische Turing-Maschine (**NTM**) ist genauso definiert wie eine DTM, mit Erweiterungen:

1. ε-Übergänge in δ erlaubt
2. Wahlmöglichkeit, welches Zeichen geschrieben werden soll

Nichtdeterministische Turing-Maschine

Eine nichtdeterministische Turing-Maschine (**NTM**) ist genauso definiert wie eine DTM, mit Erweiterungen:

1. ε-Übergänge in δ erlaubt
2. Wahlmöglichkeit, welches Zeichen geschrieben werden soll
3. " Orakelmodul"

Das Orakel

Was macht ein Orakel?

Das Orakel

Was macht ein Orakel?

1. ein Orakel ist eine nichtdeterministische Turing-Maschine

Das Orakel

Was macht ein Orakel?

1. ein Orakel ist eine nichtdeterministische Turing-Maschine
2. schreibt ein Symbol auf das Band und geht nach links, oder

Das Orakel

Was macht ein Orakel?

1. ein Orakel ist eine nichtdeterministische Turing-Maschine
2. schreibt ein Symbol auf das Band und geht nach links, oder
3. hält an

Das Orakel

Was macht ein Orakel?

1. ein Orakel ist eine nichtdeterministische Turing-Maschine
2. schreibt ein Symbol auf das Band und geht nach links, oder
3. hält an

Danach steht eine mögliche Lösung unseres Problems auf dem Band. **Aber**: sie kann komplett falsch sein!

Was kann man mit einem Orakel machen?

Was kann man mit einem Orakel machen?

1. zeigen, dass ein gegebenes Problem in $\mathcal{NP}$ liegt:

Was kann man mit einem Orakel machen?

1. zeigen, dass ein gegebenes Problem in $\mathcal{NP}$ liegt:
2. Aber wie?

Was kann man mit einem Orakel machen?

1. zeigen, dass ein gegebenes Problem in $\mathcal{NP}$ liegt:
2. Aber wie?
    1. Das Orakel eine Lösung “erraten” lassen,

Was kann man mit einem Orakel machen?

1. zeigen, dass ein gegebenes Problem in $\mathcal{NP}$ liegt:
2. Aber wie?
   1. Das Orakel eine Lösung “erraten” lassen,
   2. Gegebenenfalls prüfen, ob die Kodierung der Ausgabe korrekt ist,

Was kann man mit einem Orakel machen?

1. zeigen, dass ein gegebenes Problem in $\mathcal{NP}$ liegt:
2. Aber wie?
   1. Das Orakel eine Lösung "erraten" lassen,
   2. Gegebenenfalls prüfen, ob die Kodierung der Ausgabe korrekt ist,
   3. **polynomial** überprüfen, ob die Lösung korrekt ist.

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \propto L$

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \propto L$

Definition: polynomiale Transformation

Eine polynomiale Transformation einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

1. es existiert eine **polynomial deterministische** Turing-Maschine, die f berechnet
2. für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

SAT

Das Erfüllbarkeitsproblem **SAT** (satisfiability) ist das "erste" $\mathcal{NP}$-vollständige Problem (Steven Cook hat es 1971 bewiesen).

Gegeben: Menge U von Variablen, Menge C von Klauseln über U

Frage: Exisitert eine Wahrheitsbelegung von U, so dass C erfüllt wird, d.h. dass alle Klausel aus C den Wahrheitswert *wahr* annehmen?

Aufgabe 4

Geben Sie eine nicht-triviale Instanz von SAT an und überprüfen Sie, ob es eine Wahrheitsbelegung gibt.

Aufgabe 4

Geben Sie eine nicht-triviale Instanz von SAT an und überprüfen Sie, ob es eine Wahrheitsbelegung gibt.

Wer keine nicht-triviale Instanz im Kopf hat:

$$U = \{a, b, c, d, e, f\}$$

$$C = \{\bar{a} \vee \bar{b} \vee c \vee d \vee \bar{e} \vee f, \bar{a} \vee b \vee \bar{c} \vee \bar{d} \vee e \vee \bar{f}, \bar{a} \vee b \vee \bar{c} \vee d \vee \bar{e} \vee f,$$
$$a \vee b \vee \bar{c} \vee \bar{d} \vee e \vee \bar{f}, a \vee \bar{b} \vee c \vee d \vee \bar{e} \vee f, \bar{a} \vee b \vee \bar{c} \vee d \vee e \vee f,$$
$$a \vee b \vee c \vee d \vee e \vee f\}$$

Aufgabe 5

Wie könnte eine NTM, die das Entscheidungsproblem von SAT löst, arbeiten? Schätzen Sie grob den Arbeitsaufwand ab.

Aufgabe 6

Überlegen Sie, wie eine DTM, die das Entscheidungsproblem von SAT löst, arbeiten könnte. Schätzen Sie grob den Arbeitsaufwand ab.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Problembeschreibungen

Reflexion

Was haben wir heute gelernt?

- Problembeschreibungen
- Unterschied zwischen $\mathcal{P}$ und $\mathcal{NP}$

Noch Fragen?

Vorschau

Vorschau

- $\mathcal{NP}$-Vollständigkeits-Beweise

Bis zum nächsten Mal