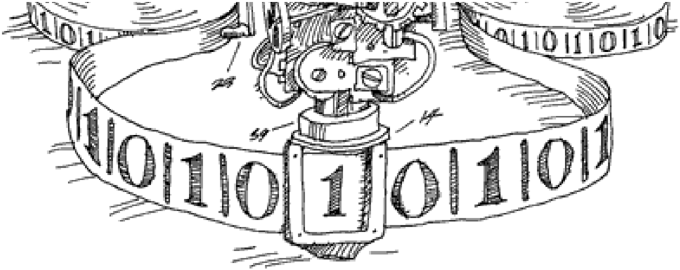
Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Tutorium 5

Institut für Theoretische Informatik



Aufgabe zur Entscheidbarkeit

Sei L eine nicht entscheidbare Sprache über dem Alphabet $\Sigma = \{0, 1\}$ und sei $(01)^k \notin L$ für jedes $k \in \mathbb{N}_{>0}$.
Zeige: Die Sprache

$$L' = \{w_1 \# w_2 \mid (w_1 \in L \wedge w_2 \notin L) \text{ oder } (w_1 \notin L \wedge w_2 \in L)\}$$

über dem Alphabet $\Sigma' = \{0, 1, \#\}$ ist nicht entscheidbar.

Aufgabe zur Entscheidbarkeit

Sei L eine nicht entscheidbare Sprache über dem Alphabet $\Sigma = \{0, 1\}$ und sei $(01)^k \notin L$ für jedes $k \in \mathbb{N}_{>0}$.
Zeige: Die Sprache

$$L' = \{w_1\#w_2 \mid (w_1 \in L \wedge w_2 \notin L) \text{ oder } (w_1 \notin L \wedge w_2 \in L)\}$$

über dem Alphabet $\Sigma' = \{0, 1, \#\}$ ist nicht entscheidbar.

Annahme: L' ist entscheidbar. Sei M' dann eine Turingmaschine, die L' entscheidet. M' entscheidet also zunächst, ob der erste Teil des Wortes in L liegt, und dann, ob der zweite Teil des Wortes nicht in L liegt.
Sei M eine Turingmaschine, die die erste Phase von M' simuliert. Dann entscheidet M die Sprache L, was ein Widerspruch zur Annahme ist.

Aufgabe zur Entscheidbarkeit

Karlsruher Institut für Technologie

Sei L eine nicht entscheidbare Sprache über dem Alphabet $\Sigma = \{0, 1\}$ und sei $(01)^k \notin L$ für jedes $k \in \mathbb{N}_{>0}$.
Zeige: Die Sprache

$$L' = \{w_1 \# w_2 \mid (w_1 \in L \land w_2 \notin L) \text{ oder } (w_1 \notin L \land w_2 \in L)\}$$

über dem Alphabet $\Sigma' = \{0, 1, \#\}$ ist nicht entscheidbar.

Annahme: L' ist entscheidbar. Sei M' dann eine Turingmaschine, die L' entscheidet. M' entscheidet also zunächst, ob der erste Teil des Wortes in L liegt, und dann, ob der zweite Teil des Wortes nicht in L liegt.
Sei M eine Turingmaschine, die die erste Phase von M' simuliert. Dann entscheidet M die Sprache L, was ein Widerspruch zur Annahme ist.

NOPE

Aufgabe zur Entscheidbarkeit

Sei L eine nicht entscheidbare Sprache über dem Alphabet $\Sigma = \{0, 1\}$ und sei $(01)^k \notin L$ für jedes $k \in \mathbb{N}_{>0}$.
Zeige: Die Sprache

$$L' = \{w_1 \# w_2 \mid (w_1 \in L \land w_2 \notin L) \text{ oder } (w_1 \notin L \land w_2 \in L)\}$$

über dem Alphabet $\Sigma' = \{0, 1, \#\}$ ist nicht entscheidbar.

Annahme: L' ist entscheidbar. Sei M' eine TM, die L' entscheidet. Für ein Wort $w \in \Sigma^*$ konstruiere das Wort $w' = w\#01$. Dann gilt:
$w' \in L' \Leftrightarrow w \in L$.
Damit kann man L wie folgt entscheiden, im Widerspruch zur Nichtentscheidbarkeit von L:
Simuliere die TM M' auf Eingabe w' und akzeptiere $w \in L$ gdw. M' $w' \in L'$ akzeptiert.

Probleme

Karlsruher Institut für Technologie

Definition (Skript)

Ein *Problem* Π ist gegeben durch:

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

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

Entscheidungsproblem

Definition

Ein Entscheidungsproblem ist ein Problem, bei dem die Lösung „Ja“ oder „Nein“ lautet. Entscheidungsprobleme lassen sich gut auf Sprachen abbilden.

Optimierungsproblem und Entscheidungsproblem

Beispiel

1. Die Aufgabe, aus einem gegebenen Blech möglichst viele sternförmige Weihnachtsplätzchen zu gewinnen, ist ein Optimierungsproblem.
2. Die Frage, ob aus diesem Blech mindestens k Plätzchen gewonnen werden können, ist ein zugehöriges Entscheidungsproblem.

Überlegung

Wie kann man mit einer Turingmaschine, welche das Entscheidungsproblem löst, das Optimierungsproblem lösen?

Hinweis: Nimm an, dass es nur endlich viele mögliche Kekskonfigurationen auf einem Backblech gibt.

Optimierungsproblem und Entscheidungsproblem

Überlegung

Wie kann man mit einer Turingmaschine, welche das Entscheidungsproblem löst, das Optimierungsproblem lösen?

Strategie

1. Zunächst mit binärer Suche das maximale k herausfinden.
2. Die n-te Formplatzierung beliebig festlegen. Wenn das Problem für $k - n$ Plätzchen noch lösbar ist, war die Platzierung richtig, sonst nimm sie zurück.
3. Wiederhole Schritt 2, bis alle Formen platziert sind.

Entscheidungsproblem

Gegeben seien ein gerichteter Graph $G = (V, E)$ und zwei Knoten $v, w \in V$.

1. Betrachte die Menge der Wege von v nach w in G und formuliere ein naheliegendes Optimierungsproblem.
2. Formuliere ein zu deinem Optimierungsproblem gehörendes Optimalwertproblem.
3. Formuliere ein zu deinem Optimierungsproblem gehörendes Entscheidungsproblem.

Kodierungsschemata (VL)

Definition Kodierungsschema

Ein *Kodierungsschema* s ordnet jeder Instanz eines Problems eine Zeichenkette oder Kodierung über einem Alphabet Σ zu. Die Eingabelänge eines Problembeispiels ist die Anzahl der Symbole seiner Kodierung. Dabei gibt es natürliche verschiedene Kodierungsschemata für ein bestimmtes Problem.

Äquivalenz

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

$$|s_2(I)| \leq p_2(|s_1(I)|) \text{ und } |s_1(I)| \leq p_1(|s_2(I)|)$$

für alle Problembeispiele I von Π.

Kodierungsschemata

Das Entscheidungsproblem *PRIMES* besteht darin, zu entscheiden, ob es sich bei einer gegebenen natürlichen Zahl $p > 1$ um eine Primzahl handelt. Eine Probleminstanz von *PRIMES* wird also durch eine natürliche Zahl kodiert. Für $b \in \mathbb{N}$ bezeichne s_b das Kodierungsschema, welches Zahlen zur Basis b darstellt.
Zeige, dass die Kodierungsschemata s_a und s_b für $a, b > 1$ bezüglich *PRIMES* äquivalent sind.

Überlegung

Welches Kodierungsschema s_c wäre zu s_a, s_b nicht äquivalent? Warum nicht?

Kodierung von SHORTEST-PATH

Wiederholung: SHORTEST-PATH

Gegeben seien ein gerichteter Graph $G = (V, E)$ und zwei Knoten $v, w \in V$.
Das Entscheidungsproblem SHORTEST-PATH besteht darin, zu entscheiden, ob es einen Pfad der Länge $\leq k$ von v nach w gibt.

Überlegung

Wie sähe ein mögliches Kodierungsschema auf dem Alphabet $\Sigma = \{0, 1, \#\}$ aus?

Kodierung von SHORTEST-PATH

Wiederholung: SHORTEST-PATH

Gegeben seien ein gerichteter Graph $G = (V, E)$ und zwei Knoten $v, w \in V$.
Das Entscheidungsproblem SHORTEST-PATH besteht darin, zu entscheiden, ob es einen Pfad der Länge $\leq k$ von v nach w gibt.

Überlegung

Wie sähe ein mögliches Kodierungsschema auf dem Alphabet $\Sigma = \{0, 1, \#\}$ aus?

Mögliche Kodierung

$|V| \#k\#v\#w$ gefolgt von $\#i\#j$ für $1 \leq i, j \leq |V|$, falls $(i, j) \in E$.

Zeitkomplexität

Die Zeitkomplexität einer deterministischen Turingmaschine $\mathcal{M}$ ist definiert als:

$$T_{\mathcal{M}}(n) = max \left\{ \text{\# Berechnungsschritte bei Eingabe } x \,\middle|\, x \in \Sigma^*, |x| = n \right\}$$

Die Klasse $\mathcal{P}$

Die Klasse $\mathcal{P}$ ist definiert als die Menge aller Sprachen, die von einer (deterministischen) Turingmaschine entschieden werden können, deren Zeitkomplexität höchstens polynomiell ist.

Beispiel

Anmerkung: Da eine RAM-Maschine in Polyzeit von einer TM simuliert werden kann, kann man (auf einer abstrakteren Ebene) i. A. Komplexitäten von bekannten Algorithmen in Beweisen verwenden.

Die folgende Sprache ist in $\mathcal{P}$: Die gerichteten Graphen zusammen mit einem Start- und einem Endknoten, so dass der Endknoten vom Startknoten aus über die Kanten erreichbar ist.

Grober Beweis: Tiefensuche ist in $\mathcal{O}(|V| + |E|)$

Sind folgende Sprachen in $\mathcal{P}$?

- Die Wörter der deutschen Sprache
- Die ungeraden Zahlen
- Eine reguläre Sprache L
- Die Instanzen von PKP, die eine Lösung haben

Laufzeitbetrachtung

In dieser Aufgabe bezeichne s_1 die unäre Kodierung und s_k, $k > 1$, die k-äre Kodierung. Das Entscheidungsproblem *PRIMES* fragt, ob eine gegebene Zahl $q \in \mathbb{N}$ eine Primzahl ist.

1. Betrachte $L_1 := L[\mathit{PRIMES}, s_1]$. Ein naiver Algorithmus für *PRIMES* könnte alle Zahlen $2, 3, \ldots, p - 1$ daraufhin überprüfen, ob sie die gegebene Zahl p teilen. Beschreibe kurz in Worten die Arbeitsweise einer **deterministischen** Turing-Maschine, die diesen Algorithmus implementiert und damit L_1 entscheidet. Gib die Laufzeit deiner Turing-Maschine asymptotisch an. Ist L_1 in $\mathcal{P}$? Begründe gegebenenfalls, warum L_1 in $\mathcal{P}$ ist.
2. Erst 2002 konnte gezeigt werden, dass das Problem *PRIMES* in der Klasse $\mathcal{P}$ liegt. Weshalb folgt $\mathit{PRIMES} \in \mathcal{P}$ nicht aus 1.?

Nichtdeterministische Turingmaschinen

Karlsruher Institut für Technologie

Version aus der Vorlesung

- Erweiterung der deterministischen Turingmaschine.
- Neu: *Orakelmodul* (Black Box)
- Ablauf einer Berechnung:
  - Zuerst: Orakelmodul schreibt Zeichen auf das Band links von der Eingabe.
  - Dann: Normale deterministische Berechnung
- NTM akzeptiert $\Leftrightarrow$ Es gibt eine Ausgabe des Orakelmoduls, sodass der deterministische Anteil der Maschine das Wort akzeptiert.

Äquivalente Definition

Analog zu nichtdeterministischen endlichen Automaten:

- Normale Turingmaschine mit nicht eindeutiger Übergangsfunktion
- NTM akzeptiert $\Leftrightarrow$ Es existiert ein akzeptierender Berechnungspfad

Nichtdeterministische Turingmaschinen

Karlsruher Institut für Technologie

Version aus der Vorlesung

- Erweiterung der deterministischen Turingmaschine.
- Neu: *Orakelmodul* (Black Box)
- Ablauf einer Berechnung:
  - Zuerst: Orakelmodul schreibt Zeichen auf das Band links von der Eingabe.
  - Dann: Normale deterministische Berechnung
- NTM akzeptiert $\Leftrightarrow$ Es gibt eine Ausgabe des Orakelmoduls, sodass der deterministische Anteil der Maschine das Wort akzeptiert.

Äquivalente Definition

Analog zu nichtdeterministischen endlichen Automaten:

- Normale Turingmaschine mit nicht eindeutiger Übergangsfunktion
- NTM akzeptiert $\Leftrightarrow$ Es existiert ein akzeptierender Berechnungspfad

Nichtdeterministische Turingmaschinen: Zeitkomplexität

Die Zeitkomplexität einer nichtdeterministischen Turingmaschine $\mathcal{M}$ ist definiert als:

$$T_{\mathcal{M}}(n) = max \left\{ \textbf{min. \# Berechnungsschritte bei Eingabe } x \,\middle|\, x \in \Sigma^*, |x| = n \right\}$$

Polyreduktionen

Definition (Vorlesung)

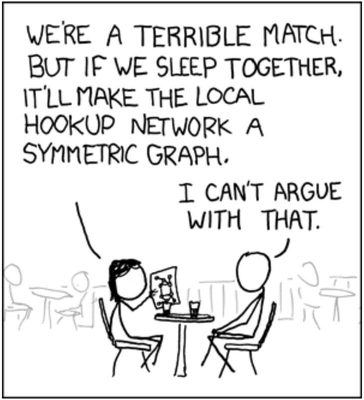
Eine polynomielle Transformation einer Sprache L_1 in eine Sprache L_2 ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

1. Es existiert eine deterministische Turing-Maschine, die in polynomieller Zeit f berechnet
2. Für alle x gilt: $x \in L_1 \iff f(x) \in L_2$

Wir schreiben dann: $L_1 \propto L_2$ (L_1 ist polynomiell transformierbar in (reduzierbar auf) L_2).

Anmerkung: In welche Richtung das $\propto$ Symbol zeigt, kann man sich folgerndermaßen merken: L_2 ist das „schwerere" Problem. Kann man L_2 entscheiden, so kann man mit polynomiellen Aufwand auch L_1 entscheiden.

Bis zum nächsten Mal!



Check it out; I've had sex with someone who's had sex with someone who's written a paper with Paul Erdős!

Lizenzen