Einführung in die Theoretische Informatik

Christian Dalvit Manuel Eberl
Samuel Frontull **Cezary Kaliszyk** Daniel Ranalter

Wintersemester 2022/23

universität innsbruck

Zusammenfassung

Wintersemester 2022/23

Satz

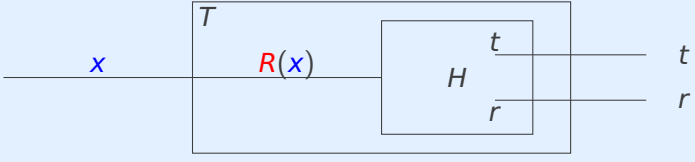
Jede partielle Funktion $f\colon \mathbb{N}^k \to \mathbb{N}$, die berechenbar auf einer RM ist, ist auf einer TM berechenbar und umgekehrt

Definition (Berechenbare Reduktion)

angenommen

- L, M Sprachen über Σ
- $L \leqslant_T M$ mit $R\colon \Sigma^* \to \Sigma^*$
- die Reduktion R wird von TM T berechnet, sodass gilt $x \in L \Leftrightarrow R(x) \in M$

Entscheidbarkeit von L, durch Entscheider H von M



Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, Konjunktive und Disjunktive Normalformen

Einführung in die Algebra

Algebraische Strukturen, Boolesche Algebra, Universelle Algebra

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, Kontextfreie Sprachen, Anwendungen von formalen Sprachen

Einführung in die Berechenbarkeitstheorie und Komplexitätstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen, Komplexitätstheorie

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Einführung in die Logik

Syntax & Semantik der Aussagenlogik, Formales Beweisen, Konjunktive und Disjunktive Normalformen

Einführung in die Algebra

Algebraische Strukturen, Boolesche Algebra, Universelle Algebra

Einführung in die Theorie der Formalen Sprachen

Grammatiken und Formale Sprachen, Chomsky-Hierarchie, Reguläre Sprachen, Kontextfreie Sprachen, Anwendungen von formalen Sprachen

Einführung in die Berechenbarkeitstheorie und Komplexitätstheorie

Algorithmisch unlösbare Probleme, Turing Maschinen, Registermaschinen, Komplexitätstheorie

Einführung in die Programmverifikation

Prinzipien der Analyse von Programmen, Verifikation nach Hoare

Satz

Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ entscheidbar; dann ist $\sim L$ entscheidbar

Satz

Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ entscheidbar; dann ist $\sim L$ entscheidbar

Beweis.

Da L entscheidbar ist, gibt es eine totale TM M mit $L = \mathrm{L}(M)$. Wir definieren eine TM M', wobei der akzeptierende und der verwerfende Zustand von M vertauscht werden. Weil M total ist, ist auch M' total. Somit akzeptiert M' ein Wort genau dann, wenn M es verwirft und es folgt $\sim L = \mathrm{L}(M')$, d.h. $\sim L$ ist entscheidbar. ■

Satz

Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ entscheidbar; dann ist $\sim L$ entscheidbar

Beweis.

Da L entscheidbar ist, gibt es eine totale TM M mit $L = \mathrm{L}(M)$. Wir definieren eine TM M', wobei der akzeptierende und der verwerfende Zustand von M vertauscht werden. Weil M total ist, ist auch M' total. Somit akzeptiert M' ein Wort genau dann, wenn M es verwirft und es folgt $\sim L = \mathrm{L}(M')$, d.h. $\sim L$ ist entscheidbar. ■

Satz

Jede entscheidbare Menge ist rekursiv aufzählbar. Andererseits ist nicht jede rekursiv aufzählbare Menge entscheidbar.

Satz

Sei Σ ein Alphabet und $L \subseteq \Sigma^*$ entscheidbar; dann ist $\sim L$ entscheidbar

Beweis.

Da L entscheidbar ist, gibt es eine totale TM M mit $L = \mathrm{L}(M)$. Wir definieren eine TM M', wobei der akzeptierende und der verwerfende Zustand von M vertauscht werden. Weil M total ist, ist auch M' total. Somit akzeptiert M' ein Wort genau dann, wenn M es verwirft und es folgt $\sim L = \mathrm{L}(M')$, d.h. $\sim L$ ist entscheidbar. ■

Satz

Jede entscheidbare Menge ist rekursiv aufzählbar. Andererseits ist nicht jede rekursiv aufzählbare Menge entscheidbar.

Beweis.

Der erste Teil des Satzes ist eine Konsequenz der Definitionen; der zweite Teil folgt daraus, dass HP zwar rek. aufz. ist, $\sim$ HP aber nicht. ($\rightarrow$ *DS*, 3. Semester) ■

Satz

Wenn L und $\sim L$ rekursiv aufzählbar sind, dann ist L entscheidbar.

Satz

Wenn L und $\sim L$ rekursiv aufzählbar sind, dann ist L entscheidbar.

Beweis.

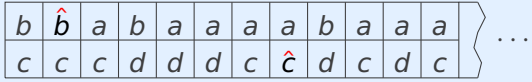
- $\exists$ TM M_1, M_2 mit $L = \mathrm{L}(M_1)$ und $\sim(L) = \mathrm{L}(M_2)$

Satz

Wenn L und $\sim L$ rekursiv aufzählbar sind, dann ist L entscheidbar.

Beweis.

- $\exists$ TM M_1, M_2 mit $L = \mathrm{L}(M_1)$ und $\sim(L) = \mathrm{L}(M_2)$
- definiere TM M', sodass das Band zwei Hälften hat

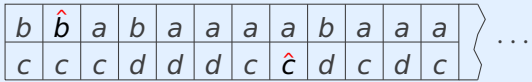


Satz

Wenn L und $\sim L$ rekursiv aufzählbar sind, dann ist L entscheidbar.

Beweis.

- $\exists$ TM M_1, M_2 mit $L = \mathrm{L}(M_1)$ und $\sim(L) = \mathrm{L}(M_2)$
- definiere TM M', sodass das Band zwei Hälften hat



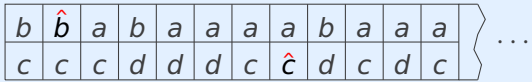
- M_1 wird auf der oberen und M_2 auf der unteren Hälfte simuliert

Satz

Wenn L und $\sim L$ rekursiv aufzählbar sind, dann ist L entscheidbar.

Beweis.

- $\exists$ TM M_1, M_2 mit $L = \mathrm{L}(M_1)$ und $\sim(L) = \mathrm{L}(M_2)$
- definiere TM M', sodass das Band zwei Hälften hat



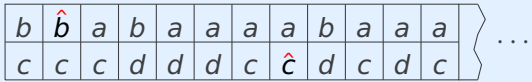
- M_1 wird auf der oberen und M_2 auf der unteren Hälfte simuliert
- wenn M_1 x akzeptiert, M' akzeptiert x
- wenn M_2 x akzeptiert, M' verwirft x

Satz

Wenn L und $\sim L$ rekursiv aufzählbar sind, dann ist L entscheidbar.

Beweis.

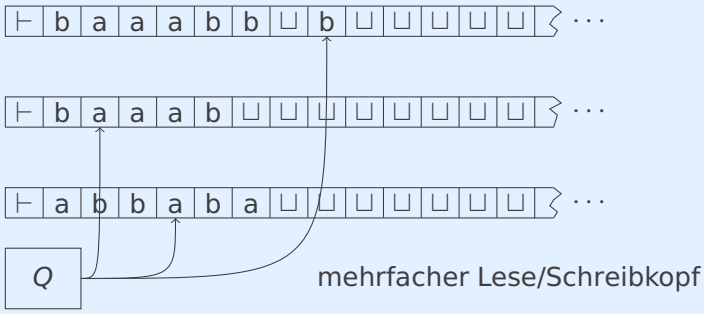
- $\exists$ TM M_1, M_2 mit $L = \mathrm{L}(M_1)$ und $\sim(L) = \mathrm{L}(M_2)$
- definiere TM M', sodass das Band zwei Hälften hat



- M_1 wird auf der oberen und M_2 auf der unteren Hälfte simuliert
- wenn M_1 x akzeptiert, M' akzeptiert x
- wenn M_2 x akzeptiert, M' verwirft x

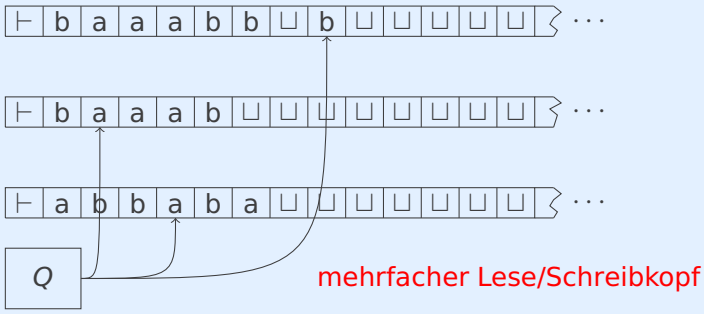
Definition (informell)

Erweiterung um mehrere Bänder und Lese/Schreibköpfe:



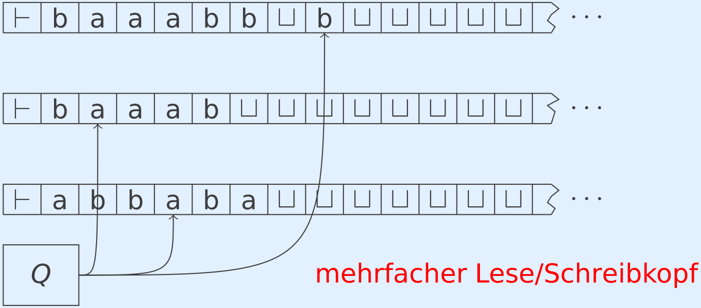
Definition (informell)

Erweiterung um mehrere Bänder und Lese/Schreibköpfe:



Definition (informell)

Erweiterung um mehrere Bänder und Lese/Schreibköpfe:



Definition

$$\delta\colon Q \times \Gamma^3 \to Q \times \Gamma^3 \times \{\mathsf{L}, \mathsf{R}\}^3$$

Satz

Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Satz

Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

Satz

Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

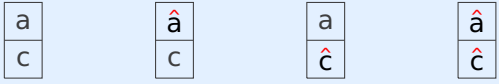
- wir simulieren die Bänder übereinander, oBdA sei $k = 2$

Satz

Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

- wir simulieren die Bänder übereinander, oBdA sei $k = 2$
- wir erweitern das Alphabet von M'

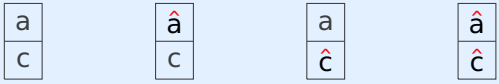


Satz

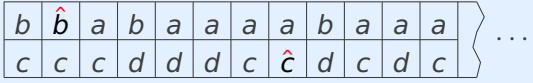
Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

- wir simulieren die Bänder übereinander, oBdA sei $k = 2$
- wir erweitern das Alphabet von M'



- Band von M' kann folgende Gestalt haben:

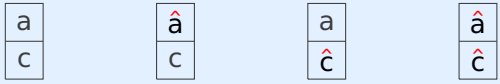


Satz

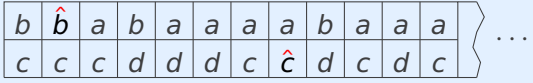
Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

- wir simulieren die Bänder übereinander, oBdA sei $k = 2$
- wir erweitern das Alphabet von M'



- Band von M' kann folgende Gestalt haben:



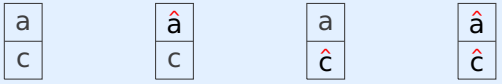
- alle Bänder von M sind nun repräsentiert und die Leseköpfe werden durch die Zusatzmarkierung ˆ ausgedrückt

Satz

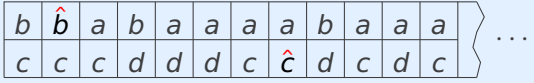
Sei M eine k-bändige TM. Dann existiert eine (einbändige) TM M', sodass $\mathrm{L}(M) = \mathrm{L}(M')$

Beweisskizze.

- wir simulieren die Bänder übereinander, oBdA sei $k = 2$
- wir erweitern das Alphabet von M'

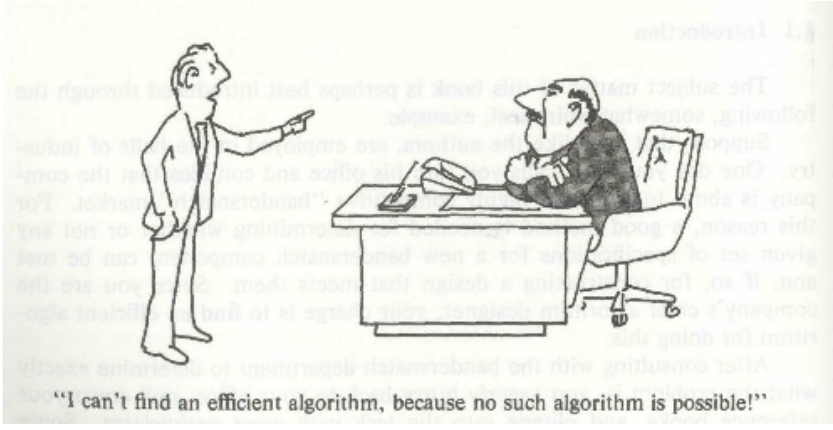


- Band von M' kann folgende Gestalt haben:



- alle Bänder von M sind nun repräsentiert und die Leseköpfe werden durch die Zusatzmarkierung $\hat{}$ ausgedrückt

Berechenbarkeitstheorie, graphisch



Einführung in die Komplexitätstheorie

Wintersemester 2022/23

Definition

Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Ressourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Definition

Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Ressourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Ressourcen

- Speicherplatz
- Rechenzeit
- ...

Definition

Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Ressourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Ressourcen

- Speicherplatz
- Rechenzeit
- ...

Problem & Algorithmus

Wir unterscheiden zwischen

- der Komplexität eines Algorithmus
- der Komplexität eines Problems

Definition

Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Ressourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Ressourcen

- Speicherplatz
- Rechenzeit
- ...

Problem & Algorithmus

Wir unterscheiden zwischen

- der Komplexität eines Algorithmus

  Algorithmus von Quine: $2^{c \cdot n}$

  (wobei n die maximale binäre Länge der Eingabe)

- der Komplexität eines Problems

Definition

Komplexitätstheorie analysiert Algorithmen und Probleme:

Welche Ressourcen benötigt ein bestimmter Algorithmus oder ein Problem?

Ressourcen

- Speicherplatz
- Rechenzeit
- ...

Problem & Algorithmus

Wir unterscheiden zwischen

- der Komplexität eines Algorithmus — Algorithmus von Quine: $2^{c \cdot n}$ (wobei n die maximale binäre Länge der Eingabe)
- der Komplexität eines Problems — SAT ist in NP

Laufzeitkomplexität

Definition

sei M eine totale TM

- die Laufzeitkomplexität von M ist Funktion $T : \mathbb{N} \to \mathbb{N}$, wobei T wie folgt definiert

$$T(n) := \max\{m \mid M \text{ hält bei Eingabe } x, |x| = n, \text{ nach } m \text{ Schritten}\}$$

Laufzeitkomplexität

Definition

sei M eine totale TM

- die Laufzeitkomplexität von M ist Funktion $T\colon \mathbb{N} \to \mathbb{N}$, wobei T wie folgt definiert

$$T(n) := \max\{m \mid M \text{ hält bei Eingabe } x, |x| = n, \text{ nach } m \text{ Schritten}\}$$

- $T(n)$ bezeichnet die Laufzeit von M, wenn n die Länge der Eingabe
- M heißt T-Zeit-Turingmaschine

Laufzeitkomplexität

Definition

sei M eine totale TM

- die Laufzeitkomplexität von M ist Funktion $T\colon \mathbb{N} \to \mathbb{N}$, wobei T wie folgt definiert

$$T(n) := \max\{m \mid M \text{ hält bei Eingabe } x,\ |x| = n, \text{ nach } m \text{ Schritten}\}$$

- $T(n)$ bezeichnet die Laufzeit von M, wenn n die Länge der Eingabe
- M heißt T-Zeit-Turingmaschine

Definition

sei $T\colon \mathbb{N} \to \mathbb{N}$ eine numerische Funktion

$$\mathrm{DTIME}(T) := \{\mathrm{L}(M) \mid M \text{ ist eine mehrbändige TM mit Laufzeit (ungefähr) in } T\ \}$$

NB: Formal gilt $\exists c \in \mathbb{R}^+\ \exists m\ \forall n \geqslant m$: Laufzeit von M bei Eingabe $x\ \leqslant c \cdot T(n)$, wobei $|x| = n$.

Die Klasse P und NP

Definition

$$\mathsf{P} := \bigcup_{k \geqslant 1} \mathrm{DTIME}(n^k)$$

P ist die Menge aller formalen Sprachen, die sich von einer deterministischen Turingmachine in polynomieller Zeit entscheiden lässt.

Die Klasse P und NP

Definition

$$\mathrm{P} := \bigcup_{k \geqslant 1} \mathrm{DTIME}(n^k)$$

P ist die Menge aller formalen Sprachen, die sich von einer deterministischen Turingmachine in polynomieller Zeit entscheiden lässt.

Beispiel

Betrachte SAT als Sprache:

$$\mathrm{SAT} = \{F \mid F \text{ Formel mit erfüllbarer Belegung v}\}$$

es gilt $\mathrm{SAT} \in \mathrm{DTIME}(2^n)$, aber es ist nicht bekannt ob $\mathrm{SAT} \in \mathrm{P}$

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass

$$L = \{x \in \Sigma^* \mid \exists\, c, \text{ sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists\, c, \text{ sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein polytime Verifikator ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $|x| = n$

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists\, c, \text{sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein polytime Verifikator ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $|x| = n$
- Wort c wird Zertifikat genannt

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists\, c, \text{sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein polytime Verifikator ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $|x| = n$
- Wort c wird Zertifikat genannt

Definition

NP ist die Klasse der Sprachen, die einen polytime Verifikator haben

Definition

- ein Verifikator einer Sprache $L \subseteq \Sigma^*$, ist ein Algorithmus V sodass
$$L = \{x \in \Sigma^* \mid \exists\, c, \text{sodass } V \text{ akzeptiert Eingabe } (x, c)\}$$
- ein polytime Verifikator ist ein Verifikator mit (ungefährer) Laufzeit n^k wobei $|x| = n$
- Wort c wird Zertifikat genannt

Definition

NP ist die Klasse der Sprachen, die einen polytime Verifikator haben

Beispiel

- Es gilt SAT $\in$ NP.
- Als Zertifikat wählen wir die (erfüllende) Belegung v für F. Für jede Belegung v kann leicht (in polynomieller Zeit) nachgewiesen werden, ob $\mathrm{v}(F) = \mathrm{T}$.

Reduktionen (in polynomieller Zeit)

Definition

1. $\exists$ k-Band TM M mit Eingabealphabet Σ
2. M läuft in polynomieller Zeit
3. bei Eingabe $x \in \Sigma^*$, schreibt M, $R(x)$ auf das (erste) Band

Reduktionen (in polynomieller Zeit)

Definition

1. $\exists$ k-Band TM M mit Eingabealphabet Σ
2. M läuft in polynomieller Zeit
3. bei Eingabe $x \in \Sigma^*$, schreibt M, $R(x)$ auf das (erste) Band

dann heißt $R\colon \Sigma^* \to \Delta^*$ in polynomieller Zeit berechenbar

Reduktionen (in polynomieller Zeit)

Definition

1. $\exists$ k-Band TM M mit Eingabealphabet Σ
2. M läuft in polynomieller Zeit
3. bei Eingabe $x \in \Sigma^*$, schreibt M, $R(x)$ auf das (erste) Band

dann heißt $R\colon \Sigma^* \to \Delta^*$ in polynomieller Zeit berechenbar

Definition

1. $\exists\, R\colon \Sigma^* \to \Delta^*$
2. R berechenbar in polynomieller Zeit
3. für $L \subseteq \Sigma^*$, $M \subseteq \Delta^*$ gilt $x \in L \Leftrightarrow R(x) \in M$

Reduktionen (in polynomieller Zeit)

Definition

1. $\exists$ k-Band TM M mit Eingabealphabet Σ
2. M läuft in polynomieller Zeit
3. bei Eingabe $x \in \Sigma^*$, schreibt M, $R(x)$ auf das (erste) Band

dann heißt $R\colon \Sigma^* \to \Delta^*$ in polynomieller Zeit berechenbar

Definition

1. $\exists\, R\colon \Sigma^* \to \Delta^*$
2. R berechenbar in polynomieller Zeit
3. für $L \subseteq \Sigma^*$, $M \subseteq \Delta^*$ gilt $x \in L \Leftrightarrow R(x) \in M$

dann ist L in polynomieller Zeit auf M reduzierbar; kurz: $L \leqslant^{\mathrm{p}} M$

Beispiel (Wiederholung)

Seien

$$L = \{x \in \{a, b\}^* \mid |x| \text{ ist gerade}\}$$
$$M = \{x \in \{a, b\}^* \mid x \text{ ist ein Palindrom gerader Länge}\}$$

dann gilt $L \leqslant^{\mathsf{p}} M$

Beispiel (Wiederholung)

Seien

$$L = \{x \in \{a,b\}^* \mid |x| \text{ ist gerade}\}$$
$$M = \{x \in \{a,b\}^* \mid x \text{ ist ein Palindrom gerader Länge}\}$$

dann gilt $L \leqslant^{\mathsf{p}} M$

Polynomielle Reduktion

Wir geben eine polynomiell berechenbare Abbildung $R\colon \{a,b\}^* \to \{a,b\}^*$ an, sodass $x \in L \Leftrightarrow R(x) \in M$:

- definiere R', sodass $R'(a) := a$ und $R'(b) := a$
- definiere R als Erweiterung von R' auf Wörter
- R ist eine Stringfunktion, die ein Wort aus $\{a,b\}^n$ in das Wort a^n umwandelt
- Genau dann wenn n gerade ist, ist a^n ein Palindrom gerader Länge

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^{\mathrm{p}} L$

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^{\mathrm{p}} L$

dann ist L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-hart.

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^{\mathrm{p}} L$

dann ist L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-hart.

Beispiel

SAT ist NP-hart

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^{\mathrm{p}} L$

dann ist L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-hart.

Beispiel

SAT ist NP-hart , dh. jedes Problem in NP ist in polynomieller Zeit auf SAT reduzierbar.

Definition

1. $\mathcal{C}$ eine beliebige Komplexitätsklasse
2. L eine Sprache über Σ
3. $\forall$ Sprachen $M \in \mathcal{C}$ gilt: $M \leqslant^{\mathrm{p}} L$

dann ist L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-hart.

Beispiel

SAT ist NP-hart , dh. jedes Problem in NP ist in polynomieller Zeit auf SAT reduzierbar.

Definition

für eine Sprache L, sei

1. L $\leqslant^{\mathrm{p}}$-hart für $\mathcal{C}$ und
2. $L \in \mathcal{C}$

dann ist L $\leqslant^{\mathrm{p}}$-vollständig für $\mathcal{C}$ oder (kurz) $\mathcal{C}$-vollständig .