Simulation von Beweissystemen

Bachelor Kolloquium

Nils Wisiol

Lehrstuhl für Informatik IV
Institut für Informatik
Julius-Maximilians-Universität Würzburg

04. Juni 2012

Outline

Einführung

Definitionen und Voraussetzungen

Resultate

Hauptsatz

Folgerungen

Zusammenfassung

Literatur

Die P-NP-Frage

"If $P = NP$, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in 'creative leaps', no fundamental gap between solving a problem and recognizing the solution once it's found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss; everyone who could recognize a good investment strategy would be Warren Buffett."

Scott Aaronson

scottaaronson.com

Konsequenzen aus der Theorie der Beweissysteme für die P-NP-Frage

Lemma

$NP \neq co\text{-}NP \implies P \neq NP.$

Proposition

Es ist genau dann NP = co-NP, wenn ein polynomiell beschränktes Beweissystem für TAUT existiert.

[KMT03]

Konsequenzen aus der Theorie der Beweissysteme für die P-NP-Frage

Lemma

$NP \neq co\text{-}NP \implies P \neq NP.$

Proposition

Es ist genau dann $NP = co\text{-}NP$, wenn ein polynomiell beschränktes Beweissystem für TAUT existiert.

[KMT03]

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Beweissysteme

Eine $\mathcal{FP}$-Funktion h ist ein *Beweissystem* für eine Sprache L, wenn $f(\Sigma^*) = L$. Ist $h(w) = x$, so ist w ein *h-Beweis* für x. h ist *polynomiell beschränkt*, wenn es ein Polynom p gibt, so dass für jedes $x \in L$ ein h-Beweis w mit $|w| \leq p(|x|)$ existiert.

Sind h und h' Beweissysteme für die Sprache L, und gibt es ein Polynom p und eine Funktion f so dass für alle h'-Beweise w

$$h(f(w)) = h'(w)$$

mit $|f(w)| \leq p(|w|)$ gilt, dann *simuliert* h das Beweissystem h'.

Simuliert ein Beweissystem h für die Sprache L jedes Beweissystem der Sprache L, so nennen wir es *optimal*. Die Komplexitätsklasse aller Sprachen, die optimale Beweissysteme besitzen, sei mit OPT bezeichnet.

Korollar 9

$P = NP \Rightarrow NP = co\text{-}NP \Leftrightarrow TAUT \in NP \Leftrightarrow TAUT \in OPT$

- Denn P = co-P
- Lemma 6
- Jede Sprache in NP ist auch in OPT; ist $TAUT \in OPT$ dann ist durch Raten von Beweisen $TAUT \in NP$.

Korollar 9

$P = NP \Rightarrow NP = \text{co-NP} \Leftrightarrow TAUT \in NP \Leftrightarrow TAUT \in OPT$

- Denn $P = \text{co-P}$
- Lemma 6
- Jede Sprache in NP ist auch in OPT; ist $TAUT \in OPT$ dann ist durch Raten von Beweisen $TAUT \in NP$.

Korollar 9

$$P = NP \Rightarrow NP = \text{co-NP} \Leftrightarrow TAUT \in NP \Leftrightarrow TAUT \in OPT$$

- Denn P = co-P
- Lemma 6
- Jede Sprache in NP ist auch in OPT; ist $TAUT \in OPT$ dann ist durch Raten von Beweisen $TAUT \in NP$.

Korollar 9

$$P = NP \Rightarrow NP = \text{co-NP} \Leftrightarrow TAUT \in NP \Leftrightarrow TAUT \in OPT$$

- Denn P = co-P
- Lemma 6
- Jede Sprache in NP ist auch in OPT; ist $TAUT \in OPT$ dann ist durch Raten von Beweisen $TAUT \in NP$.

Korollar 9

$$P = NP \Rightarrow NP = \text{co-NP} \Leftrightarrow TAUT \in NP \Leftrightarrow TAUT \in OPT$$

- Denn P = co-P
- Lemma 6
- Jede Sprache in NP ist auch in OPT; ist $TAUT \in OPT$ dann ist durch Raten von Beweisen $TAUT \in NP$.

Sprachen ohne optimales Beweissystem

Theorem

Es gibt eine Sprache $L \in$ co-NTIME(2^n), die kein optimales Beweissystem besitzt.

Beweisidee

1. $f_1, f_2, \ldots$: Aufzählung aller $\mathcal{FP}$-Funktionen
2. $L_i = 0^i 1 0^*$

   L'_i sind die Wörter aus L_i, für die keine kurzen f_i-Beweise existieren

   $L = \bigcup_i L'_i \in \text{co-NTIME}(2^n)$
3. L-Beweissystem f_i: $L'_i = L_i$, daher gibt es nur lange f_i-Beweise für $L'_i \subset L$
4. Daher führt die Anname, dass f_i ein optimales Beweissystem für L ist, zum Widerspruch

Beweisidee

1. $f_1, f_2, \ldots$: Aufzählung aller $\mathcal{FP}$-Funktionen
2. $L_i = 0^i 10^*$
   L_i' sind die Wörter aus L_i, für die keine kurzen f_i-Beweise existieren
   $L = \bigcup_i L_i' \in \text{co-NTIME}(2^n)$
3. L-Beweissystem f_i: $L_i' = L_i$, daher gibt es nur lange f_i-Beweise für $L_i' \subset L$
4. Daher führt die Anname, dass f_i ein optimales Beweissystem für L ist, zum Widerspruch

Beweisidee

1. $f_1, f_2, \ldots$: Aufzählung aller $\mathcal{FP}$-Funktionen
2. $L_i = 0^i 10^*$
   L_i' sind die Wörter aus L_i, für die keine kurzen f_i-Beweise existieren
   $L = \bigcup_i L_i' \in \text{co-NTIME}(2^n)$
3. L-Beweissystem f_i: $L_i' = L_i$, daher gibt es nur lange f_i-Beweise für $L_i' \subset L$
4. Daher führt die Anname, dass f_i ein optimales Beweissystem für L ist, zum Widerspruch

Beweisidee

1. $f_1, f_2, \ldots$: Aufzählung aller $\mathcal{FP}$-Funktionen
2. $L_i = 0^i 10^*$
   L_i' sind die Wörter aus L_i, für die keine kurzen f_i-Beweise existieren
   $L = \bigcup_i L_i' \in$ co-NTIME(2^n)
3. L-Beweissystem f_i: $L_i' = L_i$, daher gibt es nur lange f_i-Beweise für $L_i' \subset L$
4. Daher führt die Anname, dass f_i ein optimales Beweissystem für L ist, zum Widerspruch

Beweisidee

1. $f_1, f_2, \ldots$: Aufzählung aller $\mathcal{FP}$-Funktionen
2. $L_i = 0^i 10^*$
   L_i' sind die Wörter aus L_i, für die keine kurzen f_i-Beweise existieren
   $L = \bigcup_i L_i' \in$ co-NTIME(2^n)
3. L-Beweissystem f_i: $L_i' = L_i$, daher gibt es nur lange f_i-Beweise für $L_i' \subset L$
4. Daher führt die Anname, dass f_i ein optimales Beweissystem für L ist, zum Widerspruch

Beweisidee

1. $f_1, f_2, \ldots$: Aufzählung aller $\mathcal{FP}$-Funktionen
2. $L_i = 0^i 10^*$
   L'_i sind die Wörter aus L_i, für die keine kurzen f_i-Beweise existieren
   $L = \bigcup_i L'_i \in$ co-NTIME(2^n)
3. L-Beweissystem f_i: $L'_i = L_i$, daher gibt es nur lange f_i-Beweise für $L'_i \subset L$
4. Daher führt die Anname, dass f_i ein optimales Beweissystem für L ist, zum Widerspruch

Eine Aufzählung aller $\mathcal{FP}$-Funktionen

- Gödel: $M_1, M_2, \ldots$
- $M_1', M_2', \ldots$: M_i mit Wecker-Modifikation, sodass $\text{time}(M_i) \leq n^i + i$
- f_i: die von M_i berechnete Funktion
- $n^i + i$ unbeschränkt, daher alle $\mathcal{FP}$-Funktionen

Kurt Gödel
1906 – 1978

Eine Aufzählung aller $\mathcal{FP}$-Funktionen

- Gödel: $M_1, M_2, \ldots$
- $M'_1, M'_2, \ldots$: M_i mit Wecker-Modifikation, sodass $\text{time}(M_i) \leq n^i + i$
- f_i: die von M_i berechnete Funktion
- $n^i + i$ unbeschränkt, daher alle $\mathcal{FP}$-Funktionen

Kurt Gödel
1906 – 1978

Eine Aufzählung aller $\mathcal{FP}$-Funktionen

- Gödel: $M_1, M_2, \ldots$
- $M_1', M_2', \ldots$: M_i mit Wecker-Modifikation, sodass $\text{time}(M_i) \leq n^i + i$
- f_i: die von M_i berechnete Funktion
- $n^i + i$ unbeschränkt, daher alle $\mathcal{FP}$-Funktionen

Kurt Gödel
1906 – 1978

Eine Aufzählung aller $\mathcal{FP}$-Funktionen

- Gödel: $M_1, M_2, \ldots$
- $M_1', M_2', \ldots$: M_i mit Wecker-Modifikation, sodass $\text{time}(M_i) \leq n^i + i$
- f_i: die von M_i berechnete Funktion
- $n^i + i$ unbeschränkt, daher alle $\mathcal{FP}$-Funktionen

Kurt Gödel
1906 – 1978

Eine Aufzählung aller $\mathcal{FP}$-Funktionen

- Gödel: $M_1, M_2, \ldots$
- $M_1', M_2', \ldots$: M_i mit Wecker-Modifikation, sodass $\text{time}(M_i) \leq n^i + i$
- f_i: die von M_i berechnete Funktion
- $n^i + i$ unbeschränkt, daher alle $\mathcal{FP}$-Funktionen

Kurt Gödel
1906 – 1978

Konstruktion der gesuchten Sprache L

Kurt Gödel
1906 – 1978

- $L_i = 0^i 10^*$
- Wähle $x \in L'_i$, die keine kurzen f_i-Beweise haben

$$L'_i = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$$

- Vereinigung

$$L = \bigcup_{i>0} L'_i$$

Konstruktion der gesuchten Sprache L

Kurt Gödel
1906 – 1978

- $L_i = 0^i 10^*$
- Wähle $x \in L'_i$, die keine kurzen f_i-Beweise haben

$$L'_i = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$$

- Vereinigung

$$L = \bigcup_{i>0} L'_i$$

Konstruktion der gesuchten Sprache L

Kurt Gödel
1906 – 1978

- $L_i = 0^i 10^*$
- Wähle $x \in L'_i$, die keine kurzen f_i-Beweise haben

$$L'_i = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$$

- Vereinigung

$$L = \bigcup_{i>0} L'_i$$

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME$(2^n) \Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee (\exists_{y \in \Sigma^*} (|y|^{2i} \leq 2^{|x|}) \wedge (f_i(y) = x))\}$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee (\exists_{y \in \Sigma^*} (|y|^{2i} \leq 2^{|x|}) \wedge (f_i(y) = x))\}$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wähle i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wählte i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wählte i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wählte i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wählte i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME(2^n) $\Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wählte i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

L liegt in co-NTIME(2^n)

Betrachte Komplexität des Komplements

- $L \in$ co-NTIME$(2^n) \Leftrightarrow \overline{L} \in$ NTIME(2^n)
- $\overline{L} = \overline{\bigcup_{i>0} L'_i} = \bigcap_{i>0} \overline{L'_i}$
- zu zeigen: $\bigcap_{i>0} \overline{L'_i} \in$ NTIME(2^n)

$$\overline{L'_i} = \{x \in \Sigma^* : x \notin L_i \vee \left(\exists_{y \in \Sigma^*} \left(|y|^{2i} \leq 2^{|x|}\right) \wedge \left(f_i(y) = x\right)\right)\}$$

Sei dazu x beliebiges Wort.

- Prüfe, ob x in irgendeinem L_i: falls nicht, dann $x \in \overline{L}$
- Wählte i^* so, dass $x \in L_{i^*}$
- für jedes y mit $|y|^{2i} \leq 2^{|x|}$: berechne $f_{i^*}(y)$. Genau falls $f_{i^*}(y) = x$, dann $x \in \overline{L}$.

Eigenschaft von L-Beweissystemen

Erinnerung: $L_i' = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$

- jedes Beweissystem für L ist ein f_i
- für dieses ist $L_i = L_i'$

Proof.

- angenommen, es gibt ein $x = 0^i 1 z \in L_i$ das nicht in L_i' liegt
- dann gibt es y mit $y^{2i} \leq 2^{|x|}$ und $f_i(y) = x$
- folglich $x \in L_i'$ und daher $x \in L$

□

Eigenschaft von L-Beweissystemen

Erinnerung: $L_i' = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$

- jedes Beweissystem für L ist ein f_i
- für dieses ist $L_i = L_i'$

Proof.

- angenommen, es gibt ein $x = 0^i 1 z \in L_i$ das nicht in L_i' liegt
- dann gibt es y mit $y^{2i} \leq 2^{|x|}$ und $f_i(y) = x$
- folglich $x \in L_i'$ und daher $x \in L$

□

Eigenschaft von L-Beweissystemen

Erinnerung: $L_i' = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$

- jedes Beweissystem für L ist ein f_i
- für dieses ist $L_i = L_i'$

Proof.

- angenommen, es gibt ein $x = 0^i 1 z \in L_i$ das nicht in L_i' liegt
- dann gibt es y mit $y^{2i} \leq 2^{|x|}$ und $f_i(y) = x$
- folglich $x \in L_i'$ und daher $x \in L$

□

Eigenschaft von L-Beweissystemen

Erinnerung: $L_i' = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$

- jedes Beweissystem für L ist ein f_i
- für dieses ist $L_i = L_i'$

Proof.

- angenommen, es gibt ein $x = 0^i 1z \in L_i$ das nicht in L_i' liegt
- dann gibt es y mit $y^{2i} \leq 2^{|x|}$ und $f_i(y) = x$
- folglich $x \in L_i'$ und daher $x \in L$

Eigenschaft von L-Beweissystemen

Erinnerung: $L_i' = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$

- jedes Beweissystem für L ist ein f_i
- für dieses ist $L_i = L_i'$

Proof.

- angenommen, es gibt ein $x = 0^i 1z \in L_i$ das nicht in L_i' liegt
- dann gibt es y mit $y^{2i} \leq 2^{|x|}$ und $f_i(y) = x$
- folglich $x \in L_i'$ und daher $x \in L$

Eigenschaft von L-Beweissystemen

Erinnerung: $L'_i = \{x \in L_i : \forall_{y \in \Sigma^*} |y|^{2i} \leq 2^{|x|} \implies f_i(y) \neq x\}$

- jedes Beweissystem für L ist ein f_i
- für dieses ist $L_i = L'_i$

Proof.

- angenommen, es gibt ein $x = 0^i 1z \in L_i$ das nicht in L'_i liegt
- dann gibt es y mit $y^{2i} \leq 2^{|x|}$ und $f_i(y) = x$
- folglich $x \in L'_i$ und daher $x \in L$

Dünne Sprachen ohne Beweissystem

Zusammenfassung

Literatur I

[AB09] Sanjeev Arora and Boaz Barak, *Computational complexity: A modern approach*, 1st ed., Cambridge University Press, New York, NY, USA, 2009.

[Coo71] Stephen A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing (New York, NY, USA), STOC '71, ACM, 1971, pp. 151–158.

[CR79] Stephen A. Cook and Robert A. Reckhow, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic **44** (1979), 36–50.

[For09] Lance Fortnow, *The status of the P versus NP problem*, Commun. ACM **52** (2009), no. 9, 78–86.

Literatur II

[KM00] Johannes Köbler and Jochen Messner, *Is the standard proof system for SAT p-optimal?*, Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (London, UK, UK), FST TCS 2000, Springer-Verlag, 2000, pp. 361–372.

[KMT03] Johannes Köbler, Jochen Messner, and Jacobo Torán, *Optimal proof systems imply complete sets for promise classes*, Inf. Comput. **184** (2003), no. 1, 71–92.

[KP89] Jan Krajícek and Pavel Pudlák, *Propositional proof systems, the consistency of first order theories and the complexity of computations*, J. Symb. Log. **54** (1989), no. 3, 1063–1079.

Literatur III

[Mes99] Jochen Messner, *On optimal algorithms and optimal proof systems*, Proceedings of the 16th annual conference on Theoretical aspects of computer science (Berlin, Heidelberg), STACS'99, Springer-Verlag, 1999, pp. 541–550.

[Pap94] Christos H. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994.

Bildquellen I

- Kurt Gödel: Wikipedia, `de.wikipedia.org`