Theoretische Informatik II

Einheit 7

Theoretische Informatik im Rückblick

1. Berechenbarkeitsmodelle
2. Berechenbarkeitstheorie
3. Komplexitätstheorie
4. Methodik des Aufgabenlösens

Berechenbarkeitsmodelle

- **Turingmaschinen**
  - Endlicher Automat mit unendlichem Band als Gedächtnis
  - Beschreibung durch Zustandsüberführungstabellen und Konfigurationen
  - Akzeptierende und berechnende Variante sind äquivalent
  - Nichtdeterministische Variante mit exponentiellem Aufwand simulierbar
  - Varianten: Register, Mehrere Spuren, Mehrere Bänder
    Halbseitiges Band, Beschränktes Alphabet ... alle äquivalent
  - **Turingmaschinen erkennen genau die Typ-0 Sprachen**
  - Berechenbarkeitsbegriff übertragbar auf Zahlen, Mengen, ...

Berechenbarkeitsmodelle

- **Turingmaschinen**
  - Endlicher Automat mit unendlichem Band als Gedächtnis
  - Beschreibung durch Zustandsüberführungstabellen und Konfigurationen
  - Akzeptierende und berechnende Variante sind äquivalent
  - Nichtdeterministische Variante mit exponentiellem Aufwand simulierbar
  - Varianten: Register, Mehrere Spuren, Mehrere Bänder
    Halbseitiges Band, Beschränktes Alphabet ... alle äquivalent
  - **Turingmaschinen erkennen genau die Typ-0 Sprachen**
  - Berechenbarkeitsbegriff übertragbar auf Zahlen, Mengen, ...
- **Rekursive Funktionen**
  - Mathematischer Funktionenkalkül auf Zahlen
  - Anwendung von Operationen ($\circ$, Pr, μ) auf Grundfunktionen (s, pr_k^n, c_k^n)
  - Alle Programmiertechniken simulierbar
  - Primitiv-rekursive Funktionen als wichtige Teilklasse
  - **Äquivalent zu Turing-berechenbaren Funktionen**

Berechenbarkeitsmodelle

- **λ-Kalkül**
  - Einfaches mathematisches Modell funktionaler Programmiersprachen
  - Funktionsdefinition, -anwendung und -auswertung
  - Datenstrukturen wie Zahlen, Listen, Boole'sche Operatoren codierbar
  - **Äquivalent zu μ-rekursiven Funktionen**

Berechenbarkeitsmodelle

- **λ-Kalkül**
  - Einfaches mathematisches Modell funktionaler Programmiersprachen
  - Funktionsdefinition, -anwendung und -auswertung
  - Datenstrukturen wie Zahlen, Listen, Boole'sche Operatoren codierbar
  - **Äquivalent zu μ-rekursiven Funktionen**
- **Arithmetische Repräsentierbarkeit**
  - Beweisbasiertes Modell für logische Programmiersprachen
  - Formeln repräsentieren Ein-/Ausgabeverhalten von Funktionen
  - **Einfaches Modell, äquivalent zu rekursiven Funktionen**

Berechenbarkeitsmodelle

- **λ-Kalkül**
  - Einfaches mathematisches Modell funktionaler Programmiersprachen
  - Funktionsdefinition, -anwendung und -auswertung
  - Datenstrukturen wie Zahlen, Listen, Boole'sche Operatoren codierbar
  - **Äquivalent zu μ-rekursiven Funktionen**
- **Arithmetische Repräsentierbarkeit**
  - Beweisbasiertes Modell für logische Programmiersprachen
  - Formeln repräsentieren Ein-/Ausgabeverhalten von Funktionen
  - **Einfaches Modell, äquivalent zu rekursiven Funktionen**
- **Weitere Modelle sind ebenfalls äquivalent**
  - Abakus, Registermaschinen, Mini-Pascal, Markov-Algorithmen, …

Berechenbarkeitsmodelle

- **λ-Kalkül**
  - Einfaches mathematisches Modell funktionaler Programmiersprachen
  - Funktionsdefinition, -anwendung und -auswertung
  - Datenstrukturen wie Zahlen, Listen, Boole'sche Operatoren codierbar
  - **Äquivalent zu μ-rekursiven Funktionen**
- **Arithmetische Repräsentierbarkeit**
  - Beweisbasiertes Modell für logische Programmiersprachen
  - Formeln repräsentieren Ein-/Ausgabeverhalten von Funktionen
  - **Einfaches Modell, äquivalent zu rekursiven Funktionen**
- **Weitere Modelle sind ebenfalls äquivalent**
  - Abakus, Registermaschinen, Mini-Pascal, Markov-Algorithmen, ...
- **Church'sche These**
  - **Intuitiv berechenbare Funktionen sind Turing-berechenbar**
  - Unbeweisbare aber praktisch sehr nützliche Arbeitshypothese

Berechenbarkeitstheorie – losgelöst vom Modell

- **Alle Theorie stützt sich auf nur vier Gesetze**
  1. Numerierbarkeit von berechenbaren Funktionen und Rechenzeiten
  2. Entscheidbarkeit der Rechenzeit
  3. Berechenbarkeit der universellen Funktion
  4. Effektive Kombinierbarkeit berechenbarer Funktionen

Berechenbarkeitstheorie – losgelöst vom Modell

- **Alle Theorie stützt sich auf nur vier Gesetze**
  1. Numerierbarkeit von berechenbaren Funktionen und Rechenzeiten
  2. Entscheidbarkeit der Rechenzeit
  3. Berechenbarkeit der universellen Funktion
  4. Effektive Kombinierbarkeit berechenbarer Funktionen

- **Aufzählbarkeit & Entscheidbarkeit**
  – Definiert durch Berechenbarkeit (partiell) charakteristischer Funktionen
  – Viele äquivalente Charakterisierungen von Aufzählbarkeit
  – M entscheidbar $\Leftrightarrow$ M und $\overline{\mathrm{M}}$ aufzählbar
  – Abschlußeigenschaften: Vereinigung, Durchschnitt, Urbild
  Für Entscheidbarkeit auch Komplement und Differenz

Berechenbarkeitstheorie – losgelöst vom Modell

- **Alle Theorie stützt sich auf nur vier Gesetze**
  1. Numerierbarkeit von berechenbaren Funktionen und Rechenzeiten
  2. Entscheidbarkeit der Rechenzeit
  3. Berechenbarkeit der universellen Funktion
  4. Effektive Kombinierbarkeit berechenbarer Funktionen

- **Aufzählbarkeit & Entscheidbarkeit**
  – Definiert durch Berechenbarkeit (partiell) charakteristischer Funktionen
  – Viele äquivalente Charakterisierungen von Aufzählbarkeit
  – M entscheidbar $\Leftrightarrow$ M und $\overline{\mathrm{M}}$ aufzählbar
  – Abschlußeigenschaften: Vereinigung, Durchschnitt, Urbild
  Für Entscheidbarkeit auch Komplement und Differenz

- **Beweistechniken für unlösbare Probleme**
  – Diagonalisierung: Unendliche Konstruktion von Gegenbeispielen
  – Monotonieargumente: Widersprüche im Wachstumsverhalten
  – Problemreduktion: Abbildung auf bekanntes unlösbares Problem
  – Satz von Rice: keine extensionale Eigenschaft ist entscheidbar

KOMPLEXITÄTSTHEORIE

- **Komplexitätsmaße**
  – Zeit- und Platzkomplexität relativ zur Größe der Eingabe
  – Vereinfachte Komplexitätsabschätzungen genügen
  – Asymptotische Meßgröße $\mathcal{O}(f)$ für worst-case Analyse
  – Obergrenze für Handhabbarkeit ist polynomielles Wachstum

Komplexitätstheorie

- **Komplexitätsmaße**
  - Zeit- und Platzkomplexität relativ zur Größe der Eingabe
  - Vereinfachte Komplexitätsabschätzungen genügen
  - Asymptotische Meßgröße $\mathcal{O}(f)$ für worst-case Analyse
  - Obergrenze für Handhabbarkeit ist polynomielles Wachstum
- **Analyse der Komplexität von Algorithmen**
  - Suchverfahren - lineare und logarithmische Verfahren
  - Sortierverfahren - quadratische und $\mathcal{O}(n * \log_2 n)$ Algorithmen
  - Travelling Salesman – nur exponentiell bekannt

Komplexitätstheorie

- **Komplexitätsmaße**
  - Zeit- und Platzkomplexität relativ zur Größe der Eingabe
  - Vereinfachte Komplexitätsabschätzungen genügen
  - Asymptotische Meßgröße $\mathcal{O}(f)$ für worst-case Analyse
  - Obergrenze für Handhabbarkeit ist polynomielles Wachstum
- **Analyse der Komplexität von Algorithmen**
  - Suchverfahren - lineare und logarithmische Verfahren
  - Sortierverfahren - quadratische und $\mathcal{O}(n * \log_2 n)$ Algorithmen
  - Travelling Salesman – nur exponentiell bekannt
- **Komplexität von Problemen**
  - Untere Schranken für Komplexität von Sortieren: $\mathcal{O}(n * \log_2 n)$
  - Nichtdeterministische Maschinen lösen Suchprobleme effizient
  - Komplexitätsklassen: $..LOGSPACE \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq PSPACE \subseteq ..$

$\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ **ist die wichtigste noch offene Frage**

- **$\mathcal{NP}$-Vollständigkeit**
  - Die schwierigste Klasse innerhalb von $\mathcal{NP}$
  - Erklärbar durch polynomielle Reduzierbarkeit $\leq_p$
  - **Satz von Cook**: Expliziter $\mathcal{NP}$-vollständigkeitsbeweis für SAT
  - Sonstige Beweise via $\leq_p$: $3SAT, CLIQUE, VC, KP, GC, \ldots$
  - Klassen jenseits von $\mathcal{NP}$: $co\text{–}\mathcal{NP}$, $PSPACE$-vollständig, ...

Nicht-handhabbare Probleme

- **$\mathcal{NP}$-Vollständigkeit**
  - Die schwierigste Klasse innerhalb von $\mathcal{NP}$
  - Erklärbar durch polynomielle Reduzierbarkeit $\leq_p$
  - **Satz von Cook**: Expliziter $\mathcal{NP}$-vollständigkeitsbeweis für SAT
  - Sonstige Beweise via $\leq_p$: $3SAT, CLIQUE, VC, KP, GC, \ldots$
  - Klassen jenseits von $\mathcal{NP}$: $co{-}\mathcal{NP}$, $PSPACE$-vollständig, ...
- **Grenzüberschreitung ist möglich**
  - Approximationsalgorithmen, wenn suboptimale Lösung akzeptabel
  - Probabilistische Algorithmen reduzieren Fehlerwahrscheinlichkeit
  - Pseudopolynomielle Algorithmen, wenn es an großen Zahlen liegt

Wie löst man Aufgaben effektiv und korrekt?

Nicht alles läuft nach Schema, aber Methodik führt zu mehr Erfolg

Wie löst man Aufgaben effektiv und korrekt?

Nicht alles läuft nach Schema, aber Methodik führt zu mehr Erfolg

1. Voraussetzungen präzisieren (essentiell für alles Weitere)

– Welche Begriffe sind zum Verständnis der Aufgabe erforderlich

– Was ist eigentlich genau zu tun?

Wie löst man Aufgaben effektiv und korrekt?

Nicht alles läuft nach Schema, aber Methodik führt zu mehr Erfolg

1. Voraussetzungen präzisieren (essentiell für alles Weitere)

– Welche Begriffe sind zum Verständnis der Aufgabe erforderlich

– Was ist eigentlich genau zu tun?

2. Lösungsweg entwickeln und konkretisieren

– Welche Einzelschritte benötigt man, um das Problem zu lösen?

– Lösungen der einzelnen Schritte knapp, aber präzise skizzieren

Wie löst man Aufgaben effektiv und korrekt?

Nicht alles läuft nach Schema, aber Methodik führt zu mehr Erfolg

1. Voraussetzungen präzisieren (essentiell für alles Weitere)

– Welche Begriffe sind zum Verständnis der Aufgabe erforderlich

– Was ist eigentlich genau zu tun?

2. Lösungsweg entwickeln und konkretisieren

– Welche Einzelschritte benötigt man, um das Problem zu lösen?

– Lösungen der einzelnen Schritte knapp, aber präzise skizzieren

3. Argumente zu Lösung zusammenfassen

– Ergebnis sollte ein zusammenhängendes schlüssiges Argument sein

· Lösungen der Einzelschritte Gesamtergebnis zusammenführen

· Zusammenfassend hinschreiben, was jetzt insgesamt gezeigt ist

– Auf lesbaren und verständlichen Text achten

Formeln/Textfragmente ohne erkennbaren Sinn aneinanderzureihen ist unakzeptabel

Wie löst man Aufgaben effektiv und korrekt?

Nicht alles läuft nach Schema, aber Methodik führt zu mehr Erfolg

1. Voraussetzungen präzisieren

(essentiell für alles Weitere)

– Welche Begriffe sind zum Verständnis der Aufgabe erforderlich

– Was ist eigentlich genau zu tun?

2. Lösungsweg entwickeln und konkretisieren

– Welche Einzelschritte benötigt man, um das Problem zu lösen?

– Lösungen der einzelnen Schritte knapp, aber präzise skizzieren

3. Argumente zu Lösung zusammenfassen

– Ergebnis sollte ein zusammenhängendes schlüssiges Argument sein

· Lösungen der Einzelschritte Gesamtergebnis zusammenführen

· Zusammenfassend hinschreiben, was jetzt insgesamt gezeigt ist

– Auf lesbaren und verständlichen Text achten

Formeln/Textfragmente ohne erkennbaren Sinn aneinanderzureihen ist unakzeptabel

Beispiele im Anhang dieser Folien

FRAGEN ?

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $h:\mathbb{N}\longrightarrow\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

1. Voraussetzungen präzisieren

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

1. Voraussetzungen präzisieren

– h:$\mathbb{N}^k\to\mathbb{N}$ ist primitiv-rekursiv, wenn h aus primitiv-rekursiven Funktionen durch Komposition oder primitive Rekursion entsteht

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $h:\mathbb{N}\rightarrow\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

1. Voraussetzungen präzisieren

- $h{:}\mathbb{N}^k\rightarrow\mathbb{N}$ ist primitiv-rekursiv, wenn h aus primitiv-rekursiven Funktionen durch Komposition oder primitive Rekursion entsteht
- Die Grundfunktionen s, pr_k^n, und c_k^n sind primitiv-rekursiv

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

1. Voraussetzungen präzisieren

– h:$\mathbb{N}^k\to\mathbb{N}$ ist primitiv-rekursiv, wenn h aus primitiv-rekursiven Funktionen durch Komposition oder primitive Rekursion entsteht

– Die Grundfunktionen s, pr_k^n, und c_k^n sind primitiv-rekursiv

– Bekannte primitiv-rekursive Funktionen aus Vorlesung und Übungen

· p, sub, mul, exp, …

· Fallunterscheidung, Summierung, beschränkte Minimierung, …

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

1. Voraussetzungen präzisieren

– h:$\mathbb{N}^k\to\mathbb{N}$ ist primitiv-rekursiv, wenn h aus primitiv-rekursiven Funktionen durch Komposition oder primitive Rekursion entsteht

– Die Grundfunktionen s, pr_k^n, und c_k^n sind primitiv-rekursiv

– Bekannte primitiv-rekursive Funktionen aus Vorlesung und Übungen

· p, sub, mul, exp, ...

· Fallunterscheidung, Summierung, beschränkte Minimierung, ...

– Was ist zu tun?

Drücke h durch obige Funktionen und Operatoren aus

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

· Einfache Komposition bekannter Funktionen reicht nicht

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

- Einfache Komposition bekannter Funktionen reicht nicht
- Fallunterscheidung und Minimierung passen nicht

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

– Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

- Einfache Komposition bekannter Funktionen reicht nicht
- Fallunterscheidung und Minimierung passen nicht
- Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

2. Lösungsweg konkretisieren

– Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

- · Einfache Komposition bekannter Funktionen reicht nicht
- · Fallunterscheidung und Minimierung passen nicht
- · Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\rightarrow}\mathbb{N}$ und $g{:}\mathbb{N}^2{\rightarrow}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

BEISPIEL I: $n!$ IST PRIMITIV-REKURSIV

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

· Einfache Komposition bekannter Funktionen reicht nicht

· Fallunterscheidung und Minimierung passen nicht

· Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\rightarrow}\mathbb{N}$ und $g{:}\mathbb{N}^2{\rightarrow}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $h(0) = 0! = 1 = f()$

$$h(y{+}1) = (y{+}1)! = (y{+}1){*}y! = (y{+}1){*}h(y) = g(y,h(y))$$

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

– Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

· Einfache Komposition bekannter Funktionen reicht nicht

· Fallunterscheidung und Minimierung passen nicht

· Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\to}\mathbb{N}$ und $g{:}\mathbb{N}^2{\to}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $h(0) = 0! = 1 = f()$

$$h(y{+}1) = (y{+}1)! = (y{+}1){*}y! = (y{+}1){*}h(y) = g(y,h(y))$$

– Es folgt f() = 1 also $\boldsymbol{f = c_1^0}$

und $g(y,z) = (y{+}1){*}z = mul(s(y),z)$, also $\boldsymbol{g = mul{\circ}(s{\circ}pr_1^2, pr_2^2)}$

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

- · Einfache Komposition bekannter Funktionen reicht nicht
- · Fallunterscheidung und Minimierung passen nicht
- · Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\to}\mathbb{N}$ und $g{:}\mathbb{N}^2{\to}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $h(0) = 0! = 1 = f()$

$$h(y{+}1) = (y{+}1)! = (y{+}1){*}y! = (y{+}1){*}h(y) = g(y,h(y))$$

– Es folgt f() = 1 also $\qquad \boldsymbol{f = c_1^0}$

und $g(y,z) = (y{+}1){*}z = mul(s(y),z)$, also $\qquad \boldsymbol{g = mul{\circ}(s{\circ}pr_1^2, pr_2^2)}$

3. Argumente zu Lösung zusammenfassen

BEISPIEL I: $n!$ IST PRIMITIV-REKURSIV

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

· Einfache Komposition bekannter Funktionen reicht nicht

· Fallunterscheidung und Minimierung passen nicht

· Versuche Schema der primitiven Rekursion

– h=$Pr[f,g]$ gilt, wenn $h(\vec{x},0)$=$f(\vec{x})$, $h(\vec{x},y{+}1)$=$g(\vec{x},y,h(\vec{x},y))$

– Dabei muß f:$\mathbb{N}^0 \to \mathbb{N}$ und g:$\mathbb{N}^2 \to \mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $h(0) = 0! = 1 = f()$

$$h(y{+}1) = (y{+}1)! = (y{+}1){*}y! = (y{+}1){*}h(y) = g(y,h(y))$$

– Es folgt f() = 1 also $\boldsymbol{f = c_1^0}$

und $g(y,z) = (y{+}1){*}z = mul(s(y),z)$, also $\boldsymbol{g = mul{\circ}(s{\circ}pr_1^2, pr_2^2)}$

3. Argumente zu Lösung zusammenfassen

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

– Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

- · Einfache Komposition bekannter Funktionen reicht nicht
- · Fallunterscheidung und Minimierung passen nicht
- · Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\to}\mathbb{N}$ und $g{:}\mathbb{N}^2{\to}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $h(0) = 0! = 1 = f()$

$$h(y{+}1) = (y{+}1)! = (y{+}1){*}y! = (y{+}1){*}h(y) = g(y,h(y))$$

– Es folgt f() = 1 also $\quad \boldsymbol{f = c_1^0}$

und $g(y,z) = (y{+}1){*}z = mul(s(y),z)$, also $\quad \boldsymbol{g = mul{\circ}(s{\circ}pr_1^2, pr_2^2)}$

3. Argumente zu Lösung zusammenfassen

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

– Operatorenschema: $h = Pr[c_1^0, (mul{\circ}(s{\circ}pr_1^2, pr_2^2))]$

Beispiel I: $n!$ ist primitiv-rekursiv

2. Lösungsweg konkretisieren

–Einzelschritte: versuche h durch ein Operatorenschema zu beschreiben

· Einfache Komposition bekannter Funktionen reicht nicht

· Fallunterscheidung und Minimierung passen nicht

· Versuche Schema der primitiven Rekursion

– $h{=}Pr[f,g]$ gilt, wenn $h(\vec{x},0){=}f(\vec{x})$, $h(\vec{x},y{+}1){=}g(\vec{x},y,h(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\rightarrow}\mathbb{N}$ und $g{:}\mathbb{N}^2{\rightarrow}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $h(0) = 0! = 1 = f()$

$$h(y{+}1) = (y{+}1)! = (y{+}1)*y! = (y{+}1)*h(y) = g(y,h(y))$$

– Es folgt f() = 1 also $\boldsymbol{f = c_1^0}$

und $g(y,z) = (y{+}1)*z = mul(s(y),z)$, also $\boldsymbol{g = mul\circ(s\circ pr_1^2, pr_2^2)}$

3. Argumente zu Lösung zusammenfassen

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

– Operatorenschema: $h = Pr[c_1^0, (mul\circ(s\circ pr_1^2, pr_2^2))]$

– Wir setzen ein: $mul = Pr[c_0^1, (add\circ(pr_1^3, pr_3^3))]$ und $add = Pr[pr_1^1, s\circ pr_3^3]$

$$\boldsymbol{h = Pr[c_1^0, (Pr[c_0^1, (Pr[pr_1^1, s\circ pr_3^3]\circ(pr_1^3, pr_3^3))]\circ(s\circ pr_1^2, pr_2^2))]}$$

Beispiel I: $n!$ ist primitiv-rekursiv

Zeige, daß $h:\mathbb{N}\longrightarrow\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

Beispiel I: $n!$ ist primitiv-rekursiv

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

Beispiel I: $n!$ ist primitiv-rekursiv

Zeige, daß $h:\mathbb{N}\rightarrow\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

– Es ist $h(0) = 0! = 1 = f()$,

$$h(y+1) = (y+1)! = (y+1)*y! = (y+1)*h(y) = g(y, h(y))$$

BEISPIEL I: $n!$ IST PRIMITIV-REKURSIV

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

– Es ist $h(0) = 0! = 1 = f()$,

$$h(y{+}1) = (y{+}1)! = (y{+}1)*y! = (y{+}1)*h(y) = g(y, h(y))$$

– Es folgt $f() = 1$, also $\qquad f = c_1^0$

und $g(y,z) = (y{+}1)*z = mul(s(y), z)$, also $\qquad g = mul\circ(s\circ pr_1^2, pr_2^2)$

BEISPIEL I: $n!$ IST PRIMITIV-REKURSIV

Zeige, daß $h:\mathbb{N}\rightarrow\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

– Es ist $h(0) = 0! = 1 = f()$,

$$h(y+1) = (y+1)! = (y+1)*y! = (y+1)*h(y) = g(y, h(y))$$

– Es folgt $f() = 1$, also $f = c_1^0$

und $g(y,z) = (y+1)*z = mul(s(y),z)$, also $g = mul\circ(s\circ pr_1^2, pr_2^2)$

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

---

Beispiel I: $n!$ ist primitiv-rekursiv

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

– Es ist $h(0) = 0! = 1 = f()$,

$$h(y+1) = (y+1)! = (y+1)*y! = (y+1)*h(y) = g(y, h(y))$$

– Es folgt $f() = 1$, also $\qquad f = c_1^0$

und $g(y, z) = (y+1)*z = mul(s(y), z)$, also $\qquad g = mul\circ(s\circ pr_1^2, pr_2^2)$

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

---

– Operatorenschema: $h = Pr[c_1^0, (mul\circ(s\circ pr_1^2, pr_2^2))]$

BEISPIEL I: $n!$ IST PRIMITIV-REKURSIV

Zeige, daß $h : \mathbb{N} \to \mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

– Es ist $h(0) = 0! = 1 = f()$,

$$h(y+1) = (y+1)! = (y+1)*y! = (y+1)*h(y) = g(y, h(y))$$

– Es folgt $f() = 1$, also $\qquad f = c_1^0$

und $g(y, z) = (y+1)*z = mul(s(y), z)$, also $\qquad g = mul\circ(s\circ pr_1^2, pr_2^2)$

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

---

– Operatorenschema: $h = Pr[c_1^0, (mul\circ(s\circ pr_1^2, pr_2^2))]$

– Nach Einsetzen

$$h = Pr[c_1^0, (Pr[c_0^1, (Pr[pr_1^1, s\circ pr_3^3]\circ(pr_1^3, pr_3^3))]\circ(s\circ pr_1^2, pr_2^2))]$$

Beispiel I: $n!$ ist primitiv-rekursiv

Zeige, daß $h:\mathbb{N}\to\mathbb{N}$ mit $h(n) = n!$ primitiv rekursiv ist.

Stelle h explizit durch Operatoren und Grundfunktionen dar

Kurze, aufgeschriebene Lösung

– Wir beschreiben h durch das Schema der primitiven Rekursion

– Es ist $h(0) = 0! = 1 = f()$,

$$h(y+1) = (y+1)! = (y+1)*y! = (y+1)*h(y) = g(y, h(y))$$

– Es folgt $f() = 1$, also $\qquad f = c_1^0$

und $g(y,z) = (y+1)*z = mul(s(y), z)$, also $\qquad g = mul\circ(s\circ pr_1^2, pr_2^2)$

– Da f und g primitiv rekursiv sind, folgt daß h primitiv-rekursiv ist

---

– Operatorenschema: $h = Pr[c_1^0, (mul\circ(s\circ pr_1^2, pr_2^2))]$

– Nach Einsetzen

$$h = Pr[c_1^0, (Pr[c_0^1, (Pr[pr_1^1, s\circ pr_3^3]\circ(pr_1^3, pr_3^3))]\circ(s\circ pr_1^2, pr_2^2))]$$

In vielen Fällen greift ein anderes Schema (Komposition, Minimierung, etc.) besser

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ **ist nicht entscheidbar**

1. Voraussetzungen präzisieren

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

1. Voraussetzungen präzisieren

– L ist entscheidbar, wenn χ_L berechenbar ist

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

1. Voraussetzungen präzisieren

– L ist entscheidbar, wenn χ_L berechenbar ist

– Charakteristische Funktion $\chi_L(\vec{x}) = \begin{cases} 1 & \text{falls } \vec{x} \in L, \\ 0 & \text{sonst} \end{cases}$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i, y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n)=y\}$ ist nicht entscheidbar

1. Voraussetzungen präzisieren

- L ist entscheidbar, wenn χ_L berechenbar ist
- Charakteristische Funktion $\chi_L(\vec{x}) = \begin{cases} 1 & \text{falls } \vec{x} \in L, \\ 0 & \text{sonst} \end{cases}$
- φ: Numerierung berechenbarer Funktionen,

  Für jede berechenbare Funktion f gibt es in j mit $f = \varphi_j$

BEISPIEL II: DIAGONALISIERUNG

Zeige: $\boldsymbol{RG_\varphi} = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

1. Voraussetzungen präzisieren

– L ist entscheidbar, wenn χ_L berechenbar ist

– Charakteristische Funktion $\chi_L(\vec{x}) = \begin{cases} 1 & \text{falls } \vec{x} \in L, \\ 0 & \text{sonst} \end{cases}$

– φ: Numerierung berechenbarer Funktionen,

Für jede berechenbare Funktion f gibt es in j mit $f = \varphi_j$

– Zeige, daß die Annahme "RG_φ ist entscheidbar" zum Widerspruch führt

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

1. Voraussetzungen präzisieren

– L ist entscheidbar, wenn χ_L berechenbar ist

– Charakteristische Funktion $\chi_L(\vec{x}) = \begin{cases} 1 & \text{falls } \vec{x} \in L, \\ 0 & \text{sonst} \end{cases}$

– φ: Numerierung berechenbarer Funktionen,

Für jede berechenbare Funktion f gibt es in j mit $f = \varphi_j$

– Zeige, daß die Annahme “RG_φ ist entscheidbar” zum Widerspruch führt

– Mögliche Techniken: Diagonalisierung, Monotonieargumente,

Problemreduktion, Satz von Rice

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte:
1. Annahme RG_φ ist entscheidbar
2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte:
1. Annahme RG_φ ist entscheidbar
2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte: 1. Annahme RG_φ ist entscheidbar

2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte: 1. Annahme RG_φ ist entscheidbar

2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Es gilt $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte:
1. Annahme RG_φ ist entscheidbar
2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Es gilt $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Da f berechenbar ist, gibt es einen Index j mit $f = \varphi_j$

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte: 1. Annahme RG_φ ist entscheidbar

2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Es gilt $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Da f berechenbar ist, gibt es einen Index j mit $f = \varphi_j$

zu 4.: Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi$

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte: 1. Annahme RG_φ ist entscheidbar

2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Es gilt $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Da f berechenbar ist, gibt es einen Index j mit $f = \varphi_j$

zu 4.: Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi$

$\Leftrightarrow \quad \exists n \in \mathbb{N}.\ \varphi_j(n) = j$ (nach Definition von RG_φ)

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte:
1. Annahme RG_φ ist entscheidbar
2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Es gilt $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Da f berechenbar ist, gibt es einen Index j mit $f = \varphi_j$

zu 4.: Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi$

$\Leftrightarrow\ \exists n \in \mathbb{N}.\ \varphi_j(n) = j$ (nach Definition von RG_φ)

$\Leftrightarrow\ \exists n \in \mathbb{N}.\ f(n) = j$ ($f = \varphi_j$)

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte: 1. Annahme RG_φ ist entscheidbar

2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f=\varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i)\in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i)\in RG_\varphi$

Es gilt $(i,i)\in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Da f berechenbar ist, gibt es einen Index j mit $f=\varphi_j$

zu 4.: Wir betrachten das Verhalten von f auf j

Es gilt $(j,j)\in RG_\varphi$

$\Leftrightarrow \quad \exists n\in\mathbb{N}.\ \varphi_j(n) = j$ (nach Definition von RG_φ)

$\Leftrightarrow \quad \exists n\in\mathbb{N}.\ f(n) = j$ ($f=\varphi_j$)

$\Leftrightarrow \quad (j,j)\notin RG_\varphi$ (nach Konstruktion von f)

Beispiel II: Diagonalisierung

2. Lösungsweg konkretisieren: Diagonalisierung

Einzelschritte:
1. Annahme RG_φ ist entscheidbar
2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_φ}
3. Zeige: f ist berechenbar ist, also $f = \varphi_j$ für ein j
4. Zeige: f ist auf seinem eigenen Index j widersprüchlich

---

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da Fallunterscheidung mit Test $(i,i) \in RG_\varphi$

Es gilt $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i){=}1$, also $f(i) = \mu_z[\chi_{RG_\varphi}(i,i){=}0]{+}i$

Da f berechenbar ist, gibt es einen Index j mit $f = \varphi_j$

zu 4.: Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi$

$\Leftrightarrow \exists n \in \mathbb{N}.\ \varphi_j(n) = j$ (nach Definition von RG_φ)

$\Leftrightarrow \exists n \in \mathbb{N}.\ f(n) = j$ ($f = \varphi_j$)

$\Leftrightarrow (j,j) \notin RG_\varphi$ (nach Konstruktion von f)

Dies ist ein Widerspruch. **Also kann RG_φ nicht entscheidbar sein**

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n)=y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n)=y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Beispiel II: Diagonalisierung

Zeige: $\boldsymbol{RG_\varphi} = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1$.

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1$.

Damit gibt es einen Index j mit $f = \varphi_j$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1$.

Damit gibt es einen Index j mit $f = \varphi_j$

Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n)=y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1$.

Damit gibt es einen Index j mit $f = \varphi_j$

Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi \Leftrightarrow \exists n \in \mathbb{N}.\ \varphi_j(n)=j$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n)=y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1$.

Damit gibt es einen Index j mit $f = \varphi_j$

Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi \Leftrightarrow \exists n \in \mathbb{N}.\ \varphi_j(n)=j \Leftrightarrow \exists n \in \mathbb{N}.\ f(n)=j$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1.$

Damit gibt es einen Index j mit $f = \varphi_j$

Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi \Leftrightarrow \exists n \in \mathbb{N}.\ \varphi_j(n){=}j \Leftrightarrow \exists n \in \mathbb{N}.\ f(n){=}j \Leftrightarrow (j,j) \notin RG_\varphi$

Beispiel II: Diagonalisierung

Zeige: $RG_\varphi = \{(i,y) \mid \exists n \in \mathbb{N}.\ \varphi_i(n){=}y\}$ ist nicht entscheidbar

Kurze, aufgeschriebene Lösung

Wir nehmen an RG_φ sei entscheidbar.

Dann ist die charakteristische Funktion χ_{RG_φ} berechenbar.

Wir konstruieren mit χ_{RG_φ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\varphi \\ i & \text{sonst} \end{cases}$

f ist berechenbar, da $(i,i) \in RG_\varphi \Leftrightarrow \chi_{RG_\varphi}(i,i) = 1$.

Damit gibt es einen Index j mit $f = \varphi_j$

Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\varphi \Leftrightarrow \exists n \in \mathbb{N}.\ \varphi_j(n){=}j \Leftrightarrow \exists n \in \mathbb{N}.\ f(n){=}j \Leftrightarrow (j,j) \notin RG_\varphi$

Dies ist ein Widerspruch. **Also kann RG_φ nicht entscheidbar sein**

Beispiel III: $\mathcal{NP}$-vollständigkeit

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**

Beispiel III: $\mathcal{NP}$-vollständigkeit

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$

Beispiel III: $\mathcal{NP}$-vollständigkeit

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird

Beispiel III: $\mathcal{NP}$-vollständigkeit

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls $x \in L' \Leftrightarrow f(x) \in L$ für eine polynomiell bb. Funktion f

Beispiel III: $\mathcal{NP}$-vollständigkeit

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls $x \in L' \Leftrightarrow f(x) \in L$ für eine polynomiell bb. Funktion f
  - $\boldsymbol{VC} = \{\, (G,k) \mid \exists V' \subseteq V.\, |V'| \leq k \wedge V'$ Knotenüberdeckung von $G\,\}$

BEISPIEL III: $\mathcal{NP}$-VOLLSTÄNDIGKEIT

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls $x \in L' \Leftrightarrow f(x) \in L$ für eine polynomiell bb. Funktion f
  - $\boldsymbol{VC} = \{ (G, k) \mid \exists V' \subseteq V.\ |V'| \leq k \wedge V'$ Knotenüberdeckung von $G \}$
- **Standard-Lösungsweg**

  1. Zeige $\boldsymbol{VC} \in \boldsymbol{\mathcal{NP}}$:

BEISPIEL III: $\mathcal{NP}$-VOLLSTÄNDIGKEIT

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls $x \in L' \Leftrightarrow f(x) \in L$ für eine polynomiell bb. Funktion f
  - $\boldsymbol{VC} = \{ (G, k) \mid \exists V' \subseteq V.\ |V'| \leq k \wedge V'$ Knotenüberdeckung von $G \}$
- **Standard-Lösungsweg**

  1. Zeige $\boldsymbol{VC} \in \boldsymbol{\mathcal{NP}}$:

     a) Beschreibe, welchen Lösungsvorschlag die NTM generiert

     b) Beschreibe, wie Lösungsvorschlag deterministisch überprüft wird

     c) Zeige, daß das Prüfverfahren polynomiell ist

BEISPIEL III: $\mathcal{NP}$-VOLLSTÄNDIGKEIT

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls $x \in L' \Leftrightarrow f(x) \in L$ für eine polynomiell bb. Funktion f
  - $\boldsymbol{VC} = \{ (G, k) \mid \exists V' \subseteq V. \; |V'| \leq k \wedge V'$ Knotenüberdeckung von $G \}$
- **Standard-Lösungsweg**
  1. Zeige $\boldsymbol{VC} \in \boldsymbol{\mathcal{NP}}$:
     a) Beschreibe, welchen Lösungsvorschlag die NTM generiert
     b) Beschreibe, wie Lösungsvorschlag deterministisch überprüft wird
     c) Zeige, daß das Prüfverfahren polynomiell ist
  2. Zeige $\exists \boldsymbol{L'} \in \boldsymbol{\mathcal{NPC}}. \; \boldsymbol{L'} \leq_p \boldsymbol{VC}$:

Beispiel III: $\mathcal{NP}$-vollständigkeit

Zeige, daß das Vertex Cover Problem $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls $x \in L' \Leftrightarrow f(x) \in L$ für eine polynomiell bb. Funktion f
  - $\boldsymbol{VC} = \{ (G, k) \mid \exists V' \subseteq V.\, |V'| \leq k \land V'$ Knotenüberdeckung von G $\}$
- **Standard-Lösungsweg**

  1. Zeige $\boldsymbol{VC} \in \boldsymbol{\mathcal{NP}}$:

     a) Beschreibe, welchen Lösungsvorschlag die NTM generiert

     b) Beschreibe, wie Lösungsvorschlag deterministisch überprüft wird

     c) Zeige, daß das Prüfverfahren polynomiell ist

  2. Zeige $\exists \boldsymbol{L'} \in \boldsymbol{\mathcal{NPC}}.\ \boldsymbol{L'} \leq_p \boldsymbol{VC}$:

     d) Wähle ein ähnliches, bekanntes $\mathcal{NP}$-vollständiges Problem L'

     e) Beschreibe Transformationsfunktion f von L' auf VC

     f) Zeige für alle $x \in \Sigma'^*$: $x \in L' \Leftrightarrow f(x) \in VC$

     g) Zeige, daß f in polynomieller Zeit berechnet werden kann

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal* $|V'|$ *Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal* $|V'| * |E| \leq |V|^3$ *Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal* $|V'|$ *Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal* $|V'| * |E| \leq |V|^3$ *Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E$$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \qquad \Leftrightarrow \forall v, v' \in V_c. \{v, v'\} \notin E^c$$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ — *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ — *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \qquad \Leftrightarrow \forall v, v' \in V_c. \{v, v'\} \notin E^c$$

$$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c. \{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$

$\Leftrightarrow$ $V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c.\{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$

$\Leftrightarrow V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c. \{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$

$\Leftrightarrow$ $V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE$

BEISPIEL III: $\mathcal{NP}$-VOLLSTÄNDIGKEIT VON VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \qquad \Leftrightarrow \forall v, v' \in V_c.\{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$

$\Leftrightarrow V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE \qquad \Leftrightarrow G$ hat Clique V_c der Mindestgröße k

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c.\{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$

$\Leftrightarrow V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE \quad \Leftrightarrow G$ hat Clique V_c der Mindestgröße k

$\Leftrightarrow G^c$ hat Knotenüberdeckung $V' = V - V_c$ der Maximalgröße $|V| - k$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c.\{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee \ v' \in V - V_c$

$\Leftrightarrow V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE \quad \Leftrightarrow G$ hat Clique V_c der Mindestgröße k

$\Leftrightarrow G^c$ hat Knotenüberdeckung $V' = V - V_c$ der Maximalgröße $|V| - k$

$\Leftrightarrow f(G, k) = (G^c, |V| - k) \in VC$

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c. \{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee \ v' \in V - V_c$

$\Leftrightarrow$ $V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE \quad \Leftrightarrow G$ hat Clique V_c der Mindestgröße k

$\Leftrightarrow$ G^c hat Knotenüberdeckung $V' = V - V_c$ der Maximalgröße $|V| - k$

$\Leftrightarrow f(G, k) = (G^c, |V| - k) \in VC$

g) f ist in polynomieller Zeit $\mathcal{O}(|V|^2)$ berechenbar

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

1. Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

2. Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem, zeige $CLIQUE \leq_p VC$

e) Es ist V_c eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E \quad \Leftrightarrow \forall v, v' \in V_c.\{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$

$\Leftrightarrow V - V_c$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE \quad \Leftrightarrow G$ hat Clique V_c der Mindestgröße k

$\Leftrightarrow G^c$ hat Knotenüberdeckung $V' = V - V_c$ der Maximalgröße $|V| - k$

$\Leftrightarrow f(G, k) = (G^c, |V| - k) \in VC$

g) f ist in polynomieller Zeit $\mathcal{O}(|V|^2)$ berechenbar

Aus $VC \in \mathcal{NP}$ und $CLIQUE \leq_p VC$ folgt **VC ist $\mathcal{NP}$-vollständig**