Theoretische Informatik II

Einheit 7.3

Beweistechniken für unlösbare Probleme

1. Diagonalisierung
2. Monotonieargumente
3. Problemreduktion
4. Der Satz von Rice

Grenzen der Berechenbarkeit

Wie beweist man die Unlösbarkeit eines Problems?

Wie beweist man die Unlösbarkeit eines Problems?

- **Diagonalisierung**
  - Zeige, daß eine Funktion von jeder berechenbaren Funktion an mindestens einer Stelle abweicht, also selbst nicht berechenbar sein kann

Grenzen der Berechenbarkeit

Wie beweist man die Unlösbarkeit eines Problems?

- **Diagonalisierung**
  - Zeige, daß eine Funktion von jeder berechenbaren Funktion an mindestens einer Stelle abweicht, also selbst nicht berechenbar sein kann
- **Wachstums- und Monotonieargumente**
  - Zeige, daß eine Funktion stärker wächst als jede berechenbare Funktion

Grenzen der Berechenbarkeit

Wie beweist man die Unlösbarkeit eines Problems?

- **Diagonalisierung**
  - Zeige, daß eine Funktion von jeder berechenbaren Funktion an mindestens einer Stelle abweicht, also selbst nicht berechenbar sein kann
- **Wachstums- und Monotonieargumente**
  - Zeige, daß eine Funktion stärker wächst als jede berechenbare Funktion
- **Reduktionsmethode und Abschlußeigenschaften**
  - Zeige, daß Lösung des Problems zu einer Lösung eines bekanntermaßen unlösbaren Problems führen würde

Grenzen der Berechenbarkeit

Wie beweist man die Unlösbarkeit eines Problems?

- **Diagonalisierung**
  - Zeige, daß eine Funktion von jeder berechenbaren Funktion an mindestens einer Stelle abweicht, also selbst nicht berechenbar sein kann
- **Wachstums- und Monotonieargumente**
  - Zeige, daß eine Funktion stärker wächst als jede berechenbare Funktion
- **Reduktionsmethode und Abschlußeigenschaften**
  - Zeige, daß Lösung des Problems zu einer Lösung eines bekanntermaßen unlösbaren Problems führen würde
- **Anwendung allgemeiner theoretischer Resultate**
  - Unlösbarkeit folgt direkt aus bekannten Sätzen

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß
  - Aus dem Widerspruch folgt, daß die Annahme nicht gelten kann

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß
  - Aus dem Widerspruch folgt, daß die Annahme nicht gelten kann
- **Konstruktion des neuen Elementes**

  Cantor'sches Diagonalverfahren:

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß
  - Aus dem Widerspruch folgt, daß die Annahme nicht gelten kann
- **Konstruktion des neuen Elementes**

  Cantor'sches Diagonalverfahren:
  - Trage alle Elemente von M als Zeilen einer Tabelle auf

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß
  - Aus dem Widerspruch folgt, daß die Annahme nicht gelten kann
- **Konstruktion des neuen Elementes**

  Cantor'sches Diagonalverfahren:
  - Trage alle Elemente von M als Zeilen einer Tabelle auf
  - Konstruiere x auf Diagonale mit Abweichung an jedem Punkt

Diagonalisierung

- **Ziel**
  - Zeige, daß eine unendliche Menge M eine Eigenschaft P nicht besitzt
  - z.B. Entscheidbarkeit, Aufzählbarkeit, Abzählbarkeit
- **Methodik**
  - Wir nehmen an, M habe die Eigenschaft P
  - Konstruiere ein Element x, das von allen Elementen von M verschieden ist
  - Zeige, daß $x \in M$ aufgrund der Annahme gelten muß
  - Aus dem Widerspruch folgt, daß die Annahme nicht gelten kann
- **Konstruktion des neuen Elementes**

  Cantor'sches Diagonalverfahren:
  - Trage alle Elemente von M als Zeilen einer Tabelle auf
  - Konstruiere x auf Diagonale mit Abweichung an jedem Punkt
  - Also kann x nicht als Zeile vorkommen

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
  - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

    - Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)+1$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

    - Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)+1$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)+1$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

    - Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)+1$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)+1$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)+1$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

    - Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)+1$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)+1$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)+1$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

    - Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$
    - f ist offensichtlich total, kann aber in der Tabelle nicht vorkommen

Diagonalbeweise I: Überabzählbarkeit von $\mathbb{N}\to\mathbb{N}$

- **Die Menge $\mathbb{N}\to\mathbb{N}$ "überabzählbar" unendlich**
  - Die Menge aller Funktionen über $\mathbb{N}$ kann nicht durchnumeriert werden
    - **Annahme**: $\mathbb{N}\to\mathbb{N}$ ist abzählbar
    - Dann können alle Funktionen über $\mathbb{N}$ in eine Tabelle eingetragen werden

	0	1	2	3	4	...
f_0	$f_0(0)+1$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)+1$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)+1$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)+1$	$f_3(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

    - Definiere eine neue Funktion $f{:}\mathbb{N}\to\mathbb{N}$ durch $f(x) = f_x(x)+1$
    - f ist offensichtlich total, kann aber in der Tabelle nicht vorkommen
    - Ansonsten wäre $f{=}f_i$ für ein i und $f_i(i) = f(i) = f_i(i)+1$ ✓

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N} \to \mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N} \to \mathbb{N}$ is nicht abzählbar

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N} \to \mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N} \to \mathbb{N}$ is nicht abzählbar
  - Es gibt mehr Funktionen als es berechenbare Funktionen geben kann

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N}\to\mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N}\to\mathbb{N}$ is nicht abzählbar
  - Es gibt mehr Funktionen als es berechenbare Funktionen geben kann
  - Es gibt nichtberechenbare Funktionen auf $\mathbb{N}\to\mathbb{N}$ ✓

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N} \to \mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N} \to \mathbb{N}$ is nicht abzählbar
  - Es gibt mehr Funktionen als es berechenbare Funktionen geben kann
  - Es gibt nichtberechenbare Funktionen auf $\mathbb{N} \to \mathbb{N}$ ✓
- **Konkretes Argument: Angabe eines Beispiels**

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N}\to\mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N}\to\mathbb{N}$ is nicht abzählbar
  - Es gibt mehr Funktionen als es berechenbare Funktionen geben kann
  - Es gibt nichtberechenbare Funktionen auf $\mathbb{N}\to\mathbb{N}$ ✓
- **Konkretes Argument: Angabe eines Beispiels**
  - Das Halteproblem $H = \{(i,n) \mid \varphi_i(n)\neq\perp\}$ ist unentscheidbar (Beweis folgt)

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N}\to\mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N}\to\mathbb{N}$ is nicht abzählbar
  - Es gibt mehr Funktionen als es berechenbare Funktionen geben kann
  - Es gibt nichtberechenbare Funktionen auf $\mathbb{N}\to\mathbb{N}$ ✓
- **Konkretes Argument: Angabe eines Beispiels**
  - Das Halteproblem $H = \{(i,n) \mid \varphi_i(n)\neq\perp\}$ ist unentscheidbar (Beweis folgt)
  - Die charakteristische Funktion $\chi_H{:}\mathbb{N}\to\mathbb{N}$ ist nicht berechenbar ✓

Existenz unberechenbarer Funktionen

- **Abstraktes Argument: es gibt zu viele Funktionen**
  - Die Menge der berechenbaren Funktionen in $\mathbb{N}\to\mathbb{N}$ ist abzählbar
  - Die Menge $\mathbb{N}\to\mathbb{N}$ is nicht abzählbar
  - Es gibt mehr Funktionen als es berechenbare Funktionen geben kann
  - Es gibt nichtberechenbare Funktionen auf $\mathbb{N}\to\mathbb{N}$ ✓
- **Konkretes Argument: Angabe eines Beispiels**
  - Das Halteproblem $H = \{(i,n) \mid \varphi_i(n)\neq\perp\}$ ist unentscheidbar (Beweis folgt)
  - Die charakteristische Funktion $\chi_H{:}\mathbb{N}\to\mathbb{N}$ ist nicht berechenbar ✓
  - Weitere konkrete Beispiele folgen

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $\boldsymbol{H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$ **ist entscheidbar**

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

	0	1	2	3	4	...
φ_0	×	×	×	$\perp$	×	...
φ_1	$\perp$	$\perp$	×	×	×	...
φ_2	×	×	$\perp$	×	×	...
φ_3	$\perp$	×	$\perp$	×	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\times$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\perp$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\perp$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\times$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\perp$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\perp$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\times$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\perp$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\times$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\times$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Dann ist f berechenbar, denn $f(n) = \mu_z[\chi_H(n,n) = 0]$

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Dann ist f berechenbar, denn $f(n) = \mu_z[\chi_H(n,n) = 0]$

– Also gibt es ein i mit $f = \varphi_i$

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Dann ist f berechenbar, denn $f(n) = \mu_z[\chi_H(n,n) = 0]$

– Also gibt es ein i mit $f = \varphi_i$

– Aber für dieses i gilt: $\varphi_i(i)$ hält $\Leftrightarrow$ $f(i)$ hält $\Leftrightarrow$ $\varphi_i(i)$ hält nicht

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Dann ist f berechenbar, denn $f(n) = \mu_z[\chi_H(n,n) = 0]$

– Also gibt es ein i mit $f = \varphi_i$

– Aber für dieses i gilt: $\varphi_i(i)$ hält $\Leftrightarrow$ $f(i)$ hält $\Leftrightarrow$ $\varphi_i(i)$ hält nicht

– Dies ist ein Widerspruch, also ist die Annahme "H entscheidbar" falsch ✓

Diagonalbeweise II: Unentscheidbarkeit des Halteproblems

Satz J.a

Annahme: $H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$ ist entscheidbar

– Dann ist $\chi_H : \mathbb{N} \to \mathbb{N}$ berechenbar, wobei $\chi_H(i,n) = \begin{cases} 1 & \text{wenn } \varphi_i(n) \text{ hält} \\ 0 & \text{sonst} \end{cases}$

– Definiere eine neue Funktion $f : \mathbb{N} \to \mathbb{N}$ durch

$$f(n) := \begin{cases} 0 & \text{wenn } \varphi_n(n) \text{ nicht hält} \\ \perp & \text{sonst} \end{cases}$$

	0	1	2	3	4	...
φ_0	$\perp$	$\times$	$\times$	$\perp$	$\times$	...
φ_1	$\perp$	$\times$	$\times$	$\times$	$\times$	...
φ_2	$\times$	$\times$	$\times$	$\times$	$\times$	...
φ_3	$\perp$	$\times$	$\perp$	$\perp$	$\perp$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

– Dann ist f berechenbar, denn $f(n) = \mu_z[\chi_H(n,n) = 0]$

– Also gibt es ein i mit $f = \varphi_i$

– Aber für dieses i gilt: $\varphi_i(i)$ hält $\Leftrightarrow$ $f(i)$ hält $\Leftrightarrow$ $\varphi_i(i)$ hält nicht

– Dies ist ein Widerspruch, also ist die Annahme "H entscheidbar" falsch ✓

Terminierung von Programmen ist nicht testbar

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

· Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

- Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$
- Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

· Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$

· Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

· Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

· Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$

· Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

· Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

- Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$
- Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)+1$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

- Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

- Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$
- Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)+1$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)+1$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

- Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

- Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$
- Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)+1$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)+1$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

- Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$
- h ist offensichtlich total und berechenbar, denn $h(n) = u(f(n), n)+1$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

- Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$
- Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)+1$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)+1$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

- Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$
- h ist offensichtlich total und berechenbar, denn $h(n) = u(f(n), n)+1$
- Also gibt es ein $i \in \mathcal{R}_\varphi$ mit $h = \varphi_i$ und damit ein $j \in \mathbb{N}$ mit $i{=}f(j)$

Diagonalbeweise III: Total berechenbare Funktionen sind nicht aufzählbar

Annahme: $\mathcal{R}_\varphi = \{i \mid \varphi_i \text{ total}\}$ ist aufzählbar

- Dann gibt es eine berechenbare totale Funktion f mit $\text{range}(f) = \mathcal{R}_\varphi$
- Dann lassen sich alle Funktionen aus $\mathcal{R}$ wie folgt in eine Tabelle eintragen

	0	1	2	3	4	...
$\varphi_{f(0)}$	$\varphi_{f(0)}(0)+1$	$\varphi_{f(0)}(1)$	$\varphi_{f(0)}(2)$	$\varphi_{f(0)}(3)$	$\varphi_{f(0)}(4)$	...
$\varphi_{f(1)}$	$\varphi_{f(1)}(0)$	$\varphi_{f(1)}(1)+1$	$\varphi_{f(1)}(2)$	$\varphi_{f(1)}(3)$	$\varphi_{f(1)}(4)$	...
$\varphi_{f(2)}$	$\varphi_{f(2)}(0)$	$\varphi_{f(2)}(1)$	$\varphi_{f(2)}(2)+1$	$\varphi_{f(2)}(3)$	$\varphi_{f(2)}(4)$	...
$\varphi_{f(3)}$	$\varphi_{f(3)}(0)$	$\varphi_{f(3)}(1)$	$\varphi_{f(3)}(2)$	$\varphi_{f(3)}(3)+1$	$\varphi_{f(3)}(4)$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...

- Definiere eine neue Funktion $h{:}\mathbb{N}\to\mathbb{N}$ durch $h(n) = \varphi_{f(n)}(n)+1$
- h ist offensichtlich total und berechenbar, denn $h(n) = u(f(n), n)+1$
- Also gibt es ein $i \in \mathcal{R}_\varphi$ mit $h = \varphi_i$ und damit ein $j \in \mathbb{N}$ mit $i{=}f(j)$
- Für dieses j gilt $\varphi_{f(j)}(j) = h(j) = \varphi_{f(j)}(j)+1$ ✓

Andere unlösbare Probleme mit Diagonalbeweisen

- **Selbstanwendbarkeitsproblem:**
  - $S = \{i \mid \varphi_i(i) \neq \perp\}$ unentscheidbar, aber aufzählbar

Andere unlösbare Probleme mit Diagonalbeweisen

- **Selbstanwendbarkeitsproblem:**
  - $S = \{i \mid \varphi_i(i) \neq \perp\}$ unentscheidbar, aber aufzählbar
- **Entscheidungsproblem:**
  - $E = \{(i,j) \mid \varphi_i(j) = 1\}$ unentscheidbar, aber aufzählbar

Andere unlösbare Probleme mit Diagonalbeweisen

- **Selbstanwendbarkeitsproblem:**
  - $S = \{i \mid \varphi_i(i) \neq \perp\}$ unentscheidbar, aber aufzählbar
- **Entscheidungsproblem:**
  - $E = \{(i,j) \mid \varphi_i(j) = 1\}$ unentscheidbar, aber aufzählbar
- **Monotone Funktionen:**
  - $MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}$ nicht aufzählbar

Andere unlösbare Probleme mit Diagonalbeweisen

- **Selbstanwendbarkeitsproblem:**
  - $S = \{i \mid \varphi_i(i) \neq \perp\}$ unentscheidbar, aber aufzählbar
- **Entscheidungsproblem:**
  - $E = \{(i,j) \mid \varphi_i(j) = 1\}$ unentscheidbar, aber aufzählbar
- **Monotone Funktionen:**
  - $MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}$ nicht aufzählbar
- **Entscheidungsfunktionen:**
  - $EF = \{i \mid \forall j.\ \varphi_i(j) \in \{0,1\}\}$ nicht aufzählbar

Monotoniearguments

- **Ziel**
  - Zeige, daß eine Funktion f eine Eigenschaft P nicht besitzt
  - z.B. primitiv rekursiv, berechenbar, maximale Komplexität (Rechenzeit)

Monotoniearguments

- **Ziel**
  - Zeige, daß eine Funktion f eine Eigenschaft P nicht besitzt
  - z.B. primitiv rekursiv, berechenbar, maximale Komplexität (Rechenzeit)
- **Methodik**
  - Zeige daß f stärker wächst als jede Funktion mit Eigenschaft P

Monotoniearguments

- **Ziel**
  - Zeige, daß eine Funktion f eine Eigenschaft P nicht besitzt
  - z.B. primitiv rekursiv, berechenbar, maximale Komplexität (Rechenzeit)
- **Methodik**
  - Zeige daß f stärker wächst als jede Funktion mit Eigenschaft P
    - Induktive Analyse des Wachstumsverhaltens von f
    - Analyse des maximalen Wachstums von Funktionen mit Eigenschaft P

Monotoniearguments

- **Ziel**
  - Zeige, daß eine Funktion f eine Eigenschaft P nicht besitzt
  - z.B. primitiv rekursiv, berechenbar, maximale Komplexität (Rechenzeit)
- **Methodik**
  - Zeige daß f stärker wächst als jede Funktion mit Eigenschaft P
    - Induktive Analyse des Wachstumsverhaltens von f
    - Analyse des maximalen Wachstums von Funktionen mit Eigenschaft P
  - f kann also nicht selbst Eigenschaft P besitzen

Monotoniearguments

- **Ziel**
  - Zeige, daß eine Funktion f eine Eigenschaft P nicht besitzt
  - z.B. primitiv rekursiv, berechenbar, maximale Komplexität (Rechenzeit)
- **Methodik**
  - Zeige daß f stärker wächst als jede Funktion mit Eigenschaft P
    - Induktive Analyse des Wachstumsverhaltens von f
    - Analyse des maximalen Wachstums von Funktionen mit Eigenschaft P
  - f kann also nicht selbst Eigenschaft P besitzen
- **Beispiele**
  - Die Ackermann Funktion ist nicht primitiv-rekursiv
  - Die Busy-Beaver Funktion ist nicht berechenbar folgt
  - Komplexitätsanalysen

Das Busy-Beaver Problem

Biber stauen Bäche, indem sie Holzstücke in den Bach tragen. Fleißige Biber tragen mehr Holzstücke zusammen als faule. Größere Biber können mehr leisten als kleine.

Die Busy-Beaver Funktion liefert die Länge der längsten ununterbrochenen Staumauer, die ein Biber zusammentragen kann, ohne daß schon eine Teilmauer vorhanden war.

Das Busy-Beaver Problem

Biber stauen Bäche, indem sie Holzstücke in den Bach tragen. Fleißige Biber tragen mehr Holzstücke zusammen als faule. Größere Biber können mehr leisten als kleine.

Die Busy-Beaver Funktion liefert die Länge der längsten ununterbrochenen Staumauer, die ein Biber zusammentragen kann, ohne daß schon eine Teilmauer vorhanden war.

- **Beschreibe Biber durch Turingmaschinen**
  - Holzstücke werden durch das Symbol | beschrieben
  - $\tau = (\{1..n\}, \{|\}, \{|, b\}, \delta, 1, b)$ heißt Busy-Beaver TM der Größe n
  - BBT(n) sei die Menge aller Busy-Beaver Turingmaschinen der Größe n

Das Busy-Beaver Problem

Biber stauen Bäche, indem sie Holzstücke in den Bach tragen. Fleißige Biber tragen mehr Holzstücke zusammen als faule. Größere Biber können mehr leisten als kleine.

Die Busy-Beaver Funktion liefert die Länge der längsten ununterbrochenen Staumauer, die ein Biber zusammentragen kann, ohne daß schon eine Teilmauer vorhanden war.

- **Beschreibe Biber durch Turingmaschinen**
  - Holzstücke werden durch das Symbol | beschrieben
  - $\tau = (\{1..n\}, \{|\}, \{|, b\}, \delta, 1, b)$ heißt Busy-Beaver TM der Größe n
  - BBT(n) sei die Menge aller Busy-Beaver Turingmaschinen der Größe n
- **Beschreibe Produktivität von Bibern**
  - $\text{Produktivität}(\tau) = \begin{cases} n & \text{wenn } h_\tau(\epsilon) = |^n \\ 0 & \text{wenn } \tau \text{ bei Eingabe } \epsilon \text{ nicht hält} \end{cases}$

Das Busy-Beaver Problem

Biber stauen Bäche, indem sie Holzstücke in den Bach tragen. Fleißige Biber tragen mehr Holzstücke zusammen als faule. Größere Biber können mehr leisten als kleine.

Die Busy-Beaver Funktion liefert die Länge der längsten ununterbrochenen Staumauer, die ein Biber zusammentragen kann, ohne daß schon eine Teilmauer vorhanden war.

- **Beschreibe Biber durch Turingmaschinen**
  - Holzstücke werden durch das Symbol | beschrieben
  - $\tau = (\{1..n\}, \{|\}, \{|, b\}, \delta, 1, b)$ heißt Busy-Beaver TM der Größe n
  - $\mathrm{BBT}(n)$ sei die Menge aller Busy-Beaver Turingmaschinen der Größe n
- **Beschreibe Produktivität von Bibern**
  - $\mathrm{Produktivität}(\tau) = \begin{cases} n & \text{wenn } h_\tau(\epsilon) = |^n \\ 0 & \text{wenn } \tau \text{ bei Eingabe } \epsilon \text{ nicht hält} \end{cases}$
- **Beschreibe maximal mögliche Leistung von Bibern**
  - $BB(n) = \max\{\mathrm{Produktivität}(\tau) \mid \tau \in \mathrm{BBT}(n)\}$

Das Busy-Beaver Problem

Biber stauen Bäche, indem sie Holzstücke in den Bach tragen. Fleißige Biber tragen mehr Holzstücke zusammen als faule. Größere Biber können mehr leisten als kleine.
Die Busy-Beaver Funktion liefert die Länge der längsten ununterbrochenen Staumauer, die ein Biber zusammentragen kann, ohne daß schon eine Teilmauer vorhanden war.

- **Beschreibe Biber durch Turingmaschinen**
  - Holzstücke werden durch das Symbol | beschrieben
  - $\tau = (\{1..n\}, \{|\}, \{|, b\}, \delta, 1, b)$ heißt Busy-Beaver TM der Größe n
  - $\mathrm{BBT}(n)$ sei die Menge aller Busy-Beaver Turingmaschinen der Größe n
- **Beschreibe Produktivität von Bibern**
  - $\mathrm{Produktivität}(\tau) = \begin{cases} n & \text{wenn } h_\tau(\epsilon) = |^n \\ 0 & \text{wenn } \tau \text{ bei Eingabe } \epsilon \text{ nicht hält} \end{cases}$
- **Beschreibe maximal mögliche Leistung von Bibern**
  - $BB(n) = \max\{\mathrm{Produktivität}(\tau) \mid \tau \in \mathrm{BBT}(n)\}$

Ist die Busy-Beaver Funktion berechenbar?

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1		2		r
1	b	2		l
2		2		h
2	b	1		l

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	␣	2	␣	r
1	b	2	␣	l
2	␣	2	␣	h
2	b	1	␣	l

Arbeitsweise:

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise: $b1b$

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$

$\mapsto |2b$

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$
$\mapsto |2|||$

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$
$\mapsto |2|||$
stop

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$
$\mapsto |2|||$
stop

  - Produktivität ist 3 (4, wenn man alle Holzstücke zählt)

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$
$\mapsto |2|||$
stop

  - Produktivität ist 3 (4, wenn man alle Holzstücke zählt)

- **$BB(n)$ bekannt für kleine n:**

n	1	2	3	4	5	6	...
	1	4	6	13	$\geq$4098	$\geq 6.4{*}10^{462}$	

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$
$\mapsto |2|||$
stop

  - Produktivität ist 3 (4, wenn man alle Holzstücke zählt)

- **$BB(n)$ bekannt für kleine n:**

n	1	2	3	4	5	6	...
	1	4	6	13	$\geq$4098	$\geq 6.4{*}10^{462}$	

- **Vollständige Analyse nicht möglich**
  - $|\mathrm{BBT}(n)|$ ist $(|S|*|\Gamma|*|\{r,l\}|)^{(|S|*|\Gamma|-1)} * (|S|*|\Gamma|*|\{h\}|) * (|S|*|\Gamma|) = (4n)^{2n}$
    $|\mathrm{BBT}(1)|$=16, $|\mathrm{BBT}(2)|$=4096, $|\mathrm{BBT}(3)|$=2985984, ...

Busy-Beaver Problem: Intuitive Analyse

- **Beispiel einer BBT(2) Maschine**

$\delta =$

s	a	s'	a'	P
1	|	2	|	r
1	b	2	|	l
2	|	2	|	h
2	b	1	|	l

Arbeitsweise:

$b1b$
$\mapsto |2b$
$\mapsto b1||$
$\mapsto b2b||$
$\mapsto b1b|||$
$\mapsto |2|||$
stop

  - Produktivität ist 3 (4, wenn man alle Holzstücke zählt)

- ***BB*(*n*) bekannt für kleine *n*:**

n	1	2	3	4	5	6	...
	1	4	6	13	$\geq$4098	$\geq 6.4 * 10^{462}$	

- **Vollständige Analyse nicht möglich**
  - $|\text{BBT}(n)|$ ist $(|S|*|\Gamma|*|\{r,l\}|)^{(|S|*|\Gamma|-1)} * (|S|*|\Gamma|*|\{h\}|) * (|S|*|\Gamma|) = (4n)^{2n}$
    $|\text{BBT}(1)|$=16, $|\text{BBT}(2)|$=4096, $|\text{BBT}(3)|$=2985984, ...
  - Anzahl möglicher Bandkonfigurationen einer TM ist unbegrenzt

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton:** $i > j \Leftrightarrow \mathrm{BB}(i) > \mathrm{BB}(j)$

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)
- **BB berechenbar $\Rightarrow \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))$ für ein k**

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)
- **BB berechenbar $\Rightarrow \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))$ für ein k**
  - Wähle $k :=$ Anzahl der Zustände der TM über $\Gamma=\{|, b\}$, die BB berechnet

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)
- **BB berechenbar $\Rightarrow \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))$ für ein k**
  - Wähle $k :=$ Anzahl der Zustände der TM über $\Gamma=\{|, b\}$, die BB berechnet
  - Mit n Zuständen generiere n Striche
  - Mit k Zuständen berechne jetzt $\mathrm{BB}(n)$
  - Mit witeren k Zuständen berechne $\mathrm{BB}(\mathrm{BB}(n))$

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)
- **BB berechenbar $\Rightarrow \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))$ für ein k**
  - Wähle $k :=$ Anzahl der Zustände der TM über $\Gamma=\{|, b\}$, die BB berechnet
  - Mit n Zuständen generiere n Striche
  - Mit k Zuständen berechne jetzt $\mathrm{BB}(n)$
  - Mit witeren k Zuständen berechne $\mathrm{BB}(\mathrm{BB}(n))$
- **BB kann nicht berechenbar sein**

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)
- **BB berechenbar $\Rightarrow \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))$ für ein k**
  - Wähle $k :=$ Anzahl der Zustände der TM über $\Gamma=\{|, b\}$, die BB berechnet
  - Mit n Zuständen generiere n Striche
  - Mit k Zuständen berechne jetzt $\mathrm{BB}(n)$
  - Mit witeren k Zuständen berechne $\mathrm{BB}(\mathrm{BB}(n))$
- **BB kann nicht berechenbar sein**
  - Sonst gibt es ein k, so daß für alle n: $\mathrm{BB}(n+8+2k) \geq \mathrm{BB}(\mathrm{BB}(n+8)) \geq \mathrm{BB}(2n)$

Das Busy-Beaver Problem ist unlösbar

Satz L

- **BB ist streng monoton: $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$**
  - Für alle n gilt $\mathrm{BB}(n+1)>\mathrm{BB}(n)$
    - Schreibe in Zustand 1 ein | und beginne mit der BB(n)-Maschine
  - $i>j \Leftrightarrow \mathrm{BB}(i)>\mathrm{BB}(j)$ folgt nun durch Induktion über $i-j$
- **Für alle n gilt: $\mathrm{BB}(n+8)\geq 2n$**
  - Mit n Zuständen kann man n Striche generieren
  - Mit 8 Zuständen kann man Striche verdoppeln (vgl τ_4 aus Kapitel 6.1)
- **BB berechenbar $\Rightarrow \mathrm{BB}(n+2k)\geq \mathrm{BB}(\mathrm{BB}(n))$ für ein k**
  - Wähle $k :=$ Anzahl der Zustände der TM über $\Gamma=\{|, b\}$, die BB berechnet
  - Mit n Zuständen generiere n Striche
  - Mit k Zuständen berechne jetzt $\mathrm{BB}(n)$
  - Mit witeren k Zuständen berechne $\mathrm{BB}(\mathrm{BB}(n))$
- **BB kann nicht berechenbar sein**
  - Sonst gibt es ein k, so daß für alle n: $\mathrm{BB}(n+8+2k) \geq \mathrm{BB}(\mathrm{BB}(n+8)) \geq \mathrm{BB}(2n)$
  - Für $n=2k+9$ widerspräche $\mathrm{BB}(4k+17) \geq \mathrm{BB}(4k+18)$ der Monotonie ✓

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist
  - Zeige, daß jede Lösung für P in eine Lösung für P' transformiert würde

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist
  - Zeige, daß jede Lösung für P in eine Lösung für P' transformiert würde
  - P kann also nicht lösbar sein

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist
  - Zeige, daß jede Lösung für P in eine Lösung für P' transformiert würde
  - P kann also nicht lösbar sein
- **Methodik zur Konstruktion einer Lösung**
  - Transformiere ein anderes Problem P', das als lösbar bekannt ist, in P

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist
  - Zeige, daß jede Lösung für P in eine Lösung für P' transformiert würde
  - P kann also nicht lösbar sein
- **Methodik zur Konstruktion einer Lösung**
  - Transformiere ein anderes Problem P', das als lösbar bekannt ist, in P
  - Zeige, wie eine Lösung für P' in eine Lösung für P transformiert wird

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist
  - Zeige, daß jede Lösung für P in eine Lösung für P' transformiert würde
  - P kann also nicht lösbar sein
- **Methodik zur Konstruktion einer Lösung**
  - Transformiere ein anderes Problem P', das als lösbar bekannt ist, in P
  - Zeige, wie eine Lösung für P' in eine Lösung für P transformiert wird
- **Hilfsmittel: Reduzierbarkeit $P' \leq P$** Definition M
  - $P' \leq P$, falls $P' = f^{-1}(P) = \{x \mid f(x) \in P\}$ für ein total-berechenbares f
  - "P' ist reduzierbar auf P"

Problemreduktion

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
  - Unlösbar $\hat{=}$ unentscheidbar, nicht aufzählbar, nicht in Zeit t lösbar, ...
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere P in ein anderes Problem P', das als unlösbar bekannt ist
  - Zeige, daß jede Lösung für P in eine Lösung für P' transformiert würde
  - P kann also nicht lösbar sein
- **Methodik zur Konstruktion einer Lösung**
  - Transformiere ein anderes Problem P', das als lösbar bekannt ist, in P
  - Zeige, wie eine Lösung für P' in eine Lösung für P transformiert wird
- **Hilfsmittel: Reduzierbarkeit $P' \leq P$** Definition M
  - $P' \leq P$, falls $P' = f^{-1}(P) = \{x \mid f(x) \in P\}$ für ein total-berechenbares f
  - "P' ist reduzierbar auf P" (Begriff gilt für Teilmengen von Zahlen, Worten, ...)

Beweisführung durch Reduktion

- $P' \leq P$ **bedeutet "P' ist leichter als P"**

Beweisführung durch Reduktion

- $P' \leq P$ **bedeutet "P' ist leichter als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden

Beweisführung durch Reduktion

- $P' \leq P$ **bedeutet "P' ist leichter als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden
    - Bei Eingabe x bestimme $f(x)$ (f ist die Reduktionsfunktion)

Beweisführung durch Reduktion

- $P' \leq P$ **bedeutet "P' ist leichter als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden
    - Bei Eingabe x bestimme $f(x)$ (f ist die Reduktionsfunktion)
    - Löse $f(x)$ mit der Lösungsmethode für P

Beweisführung durch Reduktion

- $P' \leq P$ **bedeutet "P' ist leichter als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden
    - Bei Eingabe x bestimme $f(x)$ (f ist die Reduktionsfunktion)
    - Löse $f(x)$ mit der Lösungsmethode für P
    - Es gilt $x \in P' \Leftrightarrow f(x) \in P$, also überträgt sich das Ergebnis

Beweisführung durch Reduktion

- **$P' \leq P$ bedeutet "P' ist leichter als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden
    - Bei Eingabe x bestimme $f(x)$ (f ist die Reduktionsfunktion)
    - Löse $f(x)$ mit der Lösungsmethode für P
    - Es gilt $x \in P' \Leftrightarrow f(x) \in P$, also überträgt sich das Ergebnis
- **Aus $P' \leq P$ und P entscheidbar folgt P' entscheidbar**
  - Übertragung von Entscheidbarkeit: $\chi_{P'}(x) = \chi_{f^{-1}(P)}(x) = \chi_P(f(x))$

Beweisführung durch Reduktion

- **$P' \leq P$ bedeutet "P' ist leichter als P"**
  - Ist P lösbar, dann kann P' wie folgt gelöst werden
    - Bei Eingabe x bestimme $f(x)$ (f ist die Reduktionsfunktion)
    - Löse $f(x)$ mit der Lösungsmethode für P
    - Es gilt $x \in P' \Leftrightarrow f(x) \in P$, also überträgt sich das Ergebnis
- **Aus $P' \leq P$ und P entscheidbar folgt P' entscheidbar**
  - Übertragung von Entscheidbarkeit: $\chi_{P'}(x) = \chi_{f^{-1}(P)}(x) = \chi_P(f(x))$
- **Aus $P' \leq P$ und P aufzählbar folgt P' aufzählbar**
  - Übertragung von Aufzählbarkeit: $\psi_{P'}(x) = \psi_{f^{-1}(P)}(x) = \psi_P(f(x))$

Beispiele von Problemreduktion

- $S = \{i \mid \varphi_i(i) \neq \perp\} \leq H = \{(i,n) \mid \varphi_i(n) \neq \perp\}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

- $\boldsymbol{\overline{H} = \{(i,n) \mid \varphi_i(n) = \perp\} \ \leq \ PROG_z = \{i \mid \varphi_i = z\}}$

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \leq H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

- $\boldsymbol{\overline{H} = \{(i,n) \mid \varphi_i(n) = \perp\} \leq PROG_z = \{i \mid \varphi_i = z\}}$

  – Es gilt $(i,n) \in \overline{H} \Leftrightarrow \forall t \in \mathbb{N}.\Phi_i(n) \neq t$.

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

- $\boldsymbol{\overline{H} = \{(i,n) \mid \varphi_i(n) = \perp\} \ \leq \ PROG_z = \{i \mid \varphi_i = z\}}$

  – Es gilt $(i,n) \in \overline{H} \ \Leftrightarrow \ \forall t \in \mathbb{N}.\Phi_i(n) \neq t$.

  – Da $\Phi = \{(i,n,t) \mid \Phi_i(n) = t\}$ entscheidbar ist, gibt es ein j mit

$$\varphi_j(i,n,t) = \chi_\Phi(i,n,t) = \begin{cases} 1 & \text{falls } (i,n,t) \in \Phi \\ 0 & \text{sonst} \end{cases}$$

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \leq H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

- $\boldsymbol{\overline{H} = \{(i,n) \mid \varphi_i(n) = \perp\} \leq PROG_z = \{i \mid \varphi_i = z\}}$

  – Es gilt $(i,n) \in \overline{H} \Leftrightarrow \forall t \in \mathbb{N}.\Phi_i(n) \neq t$.

  – Da $\Phi = \{(i,n,t) \mid \Phi_i(n) = t\}$ entscheidbar ist, gibt es ein j mit

  $$\varphi_j(i,n,t) = \chi_\Phi(i,n,t) = \begin{cases} 1 & \text{falls } (i,n,t) \in \Phi \\ 0 & \text{sonst} \end{cases}$$

  – Nach dem SMN Theorem gibt es eine total-berechenbare Funktion f mit $\varphi_{f(i,n)}(t) = \varphi_j(i,n,t)$

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

- $\boldsymbol{\overline{H} = \{(i,n) \mid \varphi_i(n) = \perp\} \ \leq \ PROG_z = \{i \mid \varphi_i = z\}}$

  – Es gilt $(i,n) \in \overline{H} \ \Leftrightarrow \ \forall t \in \mathbb{N}.\Phi_i(n) \neq t$.

  – Da $\Phi = \{(i,n,t) \mid \Phi_i(n) = t\}$ entscheidbar ist, gibt es ein j mit

  $$\varphi_j(i,n,t) = \chi_\Phi(i,n,t) = \begin{cases} 1 & \text{falls } (i,n,t) \in \Phi \\ 0 & \text{sonst} \end{cases}$$

  – Nach dem SMN Theorem gibt es eine total-berechenbare Funktion f mit $\varphi_{f(i,n)}(t) = \varphi_j(i,n,t)$

  – Es folgt $(i,n) \in \overline{H} \ \Leftrightarrow \ \forall t \in \mathbb{N}.\Phi_i(n) \neq t \ \Leftrightarrow \ \forall t \in \mathbb{N}.\varphi_{f(i,n)}(t) = 0 \ \Leftrightarrow \ \varphi_{f(i,n)} = z$

  $\Leftrightarrow \ f(i,n) \in PROG_z$

Beispiele von Problemreduktion

- $\boldsymbol{S = \{i \mid \varphi_i(i) \neq \perp\} \ \leq \ H = \{(i,n) \mid \varphi_i(n) \neq \perp\}}$

  "Das Selbstanwendbarkeitsproblem ist leichter als das Halteproblem"

  – Es gilt $i \in S \Leftrightarrow (i,i) \in H$.

  – Wähle $f(i) := (i,i)$. Dann ist f total-berechenbar und $S = f{-}1(H)$

  – Man kann auch H auf S reduzieren (aufwendig)

- $\boldsymbol{\overline{H} = \{(i,n) \mid \varphi_i(n) = \perp\} \ \leq \ PROG_z = \{i \mid \varphi_i = z\}}$

  – Es gilt $(i,n) \in \overline{H} \ \Leftrightarrow \ \forall t \in \mathbb{N}.\Phi_i(n) \neq t$.

  – Da $\Phi = \{(i,n,t) \mid \Phi_i(n) = t\}$ entscheidbar ist, gibt es ein j mit

  $$\varphi_j(i,n,t) = \chi_\Phi(i,n,t) = \begin{cases} 1 & \text{falls } (i,n,t) \in \Phi \\ 0 & \text{sonst} \end{cases}$$

  – Nach dem SMN Theorem gibt es eine total-berechenbare Funktion f mit $\varphi_{f(i,n)}(t) = \varphi_j(i,n,t)$

  – Es folgt $(i,n) \in \overline{H} \ \Leftrightarrow \ \forall t \in \mathbb{N}.\Phi_i(n) \neq t \ \Leftrightarrow \ \forall t \in \mathbb{N}.\varphi_{f(i,n)}(t) = 0 \ \Leftrightarrow \ \varphi_{f(i,n)} = z$

  $\Leftrightarrow \ f(i,n) \in PROG_z$

  also $\overline{H} = f^{-1}(PROG_z)$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$

    Lösbar mit Korrespondenz 1323, denn $u_1u_3u_2u_3 = v_1v_3v_2v_3 = 101110011$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$
    Lösbar mit Korrespondenz 1323, denn $u_1u_3u_2u_3 = v_1v_3v_2v_3 = 101110011$
  - $K_2 = \{(1, 10),\ (101, 01)\}$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$
    Lösbar mit Korrespondenz 1323, denn $u_1u_3u_2u_3 = v_1v_3v_2v_3 = 101110011$
  - $K_2 = \{(1, 10),\ (101, 01)\}$
    Unlösbar: alle u_i haben mehr Einsen als Nullen, die v_i sind ausgewogen

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$
    Lösbar mit Korrespondenz 1323, denn $u_1u_3u_2u_3 = v_1v_3v_2v_3 = 101110011$
  - $K_2 = \{(1, 10),\ (101, 01)\}$
    Unlösbar: alle u_i haben mehr Einsen als Nullen, die v_i sind ausgewogen
  - $K_3 = \{(001, 0),\ (01, 011),\ (01, 101),\ (10, 001)\}$

Das Post'sche Korrespondenzproblem

- **Gegeben:**
  - Alphabet X, Menge von Wortpaaren $\{(u_1, v_1), .., (u_k, v_k)\}$ in X^+
  - Eine Korrespondenz ist eine Indexfolge $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$
  - $\{(u_1, v_1), .., (u_k, v_k)\}$ heißt lösbar, wenn es eine Korrespondenz gibt
- **Post'sches Korrespondenzproblem, präzisiert**
  - $PKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_1, .., i_n.\ u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}\}$
  - Technisches Problem ohne direkte praktische Relevanz
  - Ausgangspunkt für Beweise von Unentscheidbarkeiten auf Grammatiken
- **Beispiele**
  - $K_1 = \{(1, 101),\ (10, 00),\ (011, 11)\}$
    Lösbar mit Korrespondenz 1323, denn $u_1u_3u_2u_3 = v_1v_3v_2v_3 = 101110011$
  - $K_2 = \{(1, 10),\ (101, 01)\}$
    Unlösbar: alle u_i haben mehr Einsen als Nullen, die v_i sind ausgewogen
  - $K_3 = \{(001, 0),\ (01, 011),\ (01, 101),\ (10, 001)\}$
    Lösbar mit 243442124343443442144213411344421211134341214421411341131131214113

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{\texttt{"("}, \texttt{")"}, \texttt{","}\})^*$

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{"(", ")", ","\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{"(", ")", ","\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k
  - Zähle alle möglichen Indexfolgen $i_1, .., i_n$ mit $i_j \leq k$ auf

    (Verwende Umkehrung der Standardtupelfunktion für Listen $\langle i_1, .., i_n \rangle^*$)

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{"(", ")", ","\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k
  - Zähle alle möglichen Indexfolgen $i_1, .., i_n$ mit $i_j \leq k$ auf

    (Verwende Umkehrung der Standardtupelfunktion für Listen $\langle i_1, .., i_n \rangle^*$)
    - Falls $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$, so akzeptiere w (Ausgabe 1)
    - Ansonsten generiere die nächste Indexfolge

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{"(", ")", ","\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k
  - Zähle alle möglichen Indexfolgen $i_1, .., i_n$ mit $i_j \leq k$ auf

    (Verwende Umkehrung der Standardtupelfunktion für Listen $\langle i_1, .., i_n \rangle^*$)

    · Falls $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$, so akzeptiere w (Ausgabe 1)

    · Ansonsten generiere die nächste Indexfolge

- **Algorithmus berechnet ψ_{PKP}**

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{"(", ")", ","\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k
  - Zähle alle möglichen Indexfolgen $i_1, .., i_n$ mit $i_j \leq k$ auf

    (Verwende Umkehrung der Standardtupelfunktion für Listen $\langle i_1, .., i_n \rangle^*$)

    · Falls $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$, so akzeptiere w (Ausgabe 1)

    · Ansonsten generiere die nächste Indexfolge

- **Algorithmus berechnet ψ_{PKP}**
  - $w \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$

    $\Rightarrow$ Aufzählung endet bei $\langle i_1, .., i_n \rangle^*$ mit Ausgabe 1

Post'sches Korrespondenzproblem: Aufzählbarkeit

- **Aufzählungsalgorithmus:**
  - **Eingabe**: Wort $w = \{(u_1, v_1), .., (u_k, v_k)\} \in (X \cup \{"(", ")", ","\})^*$
  - Durch Klammerzählung bestimme alle u_i und v_i und die Anzahl k
  - Zähle alle möglichen Indexfolgen $i_1, .., i_n$ mit $i_j \leq k$ auf

    (Verwende Umkehrung der Standardtupelfunktion für Listen $\langle i_1, .., i_n \rangle^*$)

    · Falls $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$, so akzeptiere w (Ausgabe 1)

    · Ansonsten generiere die nächste Indexfolge

- **Algorithmus berechnet ψ_{PKP}**
  - $w \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n}$

    $\Rightarrow$ Aufzählung endet bei $\langle i_1, .., i_n \rangle^*$ mit Ausgabe 1
  - $w \notin PKP \Rightarrow$ Es gibt keine Korrespondenz

    $\Rightarrow$ Aufzählung terminiert nicht, da Test niemals erfolgreich

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
  - Zeige: $MPKP \leq PKP$

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
  - Zeige: $MPKP \leq PKP$
  - Zeige: $MPKP$ ist trotz der Einschränkung unentscheidbar

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
  - Zeige: $MPKP \leq PKP$
  - Zeige: $MPKP$ ist trotz der Einschränkung unentscheidbar
- **Zeige: $H \leq MPKP$**
  - Zeige, daß jede Turingmaschine τ als MPKP beschrieben werden kann

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
  - Zeige: $MPKP \leq PKP$
  - Zeige: $MPKP$ ist trotz der Einschränkung unentscheidbar
- **Zeige: $H \leq MPKP$**
  - Zeige, daß jede Turingmaschine τ als MPKP beschrieben werden kann
    - (u_1, v_1) beschreibt die Erzeugung der Anfangskonfiguration
    - Weitere Wortpaare entsprechen Konfigurationsübergängen

      (vgl Simulation von TM durch Typ-0 Grammatiken)

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
  - Zeige: $MPKP \leq PKP$
  - Zeige: $MPKP$ ist trotz der Einschränkung unentscheidbar
- **Zeige: $H \leq MPKP$**
  - Zeige, daß jede Turingmaschine τ als MPKP beschrieben werden kann
    - (u_1, v_1) beschreibt die Erzeugung der Anfangskonfiguration
    - Weitere Wortpaare entsprechen Konfigurationsübergängen

      (vgl Simulation von TM durch Typ-0 Grammatiken)
  - τ hält, wenn das MPKP eine terminierende Berechnung beschreiben kann

Beweis durch doppelte Reduktion

- **Verwende eingeschränkte Version $MPKP$**
  - $MPKP = \{(u_1, v_1), .., (u_k, v_k) \mid u_i, v_i \in X^+ \wedge \exists i_2, .., i_n.\ u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}\}$
  - Zeige: $MPKP \leq PKP$
  - Zeige: $MPKP$ ist trotz der Einschränkung unentscheidbar
- **Zeige: $H \leq MPKP$**
  - Zeige, daß jede Turingmaschine τ als MPKP beschrieben werden kann
    - (u_1, v_1) beschreibt die Erzeugung der Anfangskonfiguration
    - Weitere Wortpaare entsprechen Konfigurationsübergängen

      (vgl Simulation von TM durch Typ-0 Grammatiken)
  - τ hält, wenn das MPKP eine terminierende Berechnung beschreiben kann
  - Korrespondenz bedeutet, daß jedes Ergebnis eines Konfigurationsübergangs Anfangspunkt des nächsten Übergangs ist

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1 ... a_n$ zu $\hat{w} = a_1 \# a_2 \# ... \# a_n$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1 ... a_n$ zu $\hat{w} = a_1 \# a_2 \# ... \# a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1, v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2, v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1}, v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2}, v'_{k+2})} \}$$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1 ... a_n$ zu $\hat{w} = a_1 \# a_2 \# ... \# a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1, v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2, v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1}, v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2}, v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1 ... a_n$ zu $\hat{w} = a_1 \# a_2 \# ... \# a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1, v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2, v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1}, v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2}, v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1}\underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}}..\#\underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}}\underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1}\underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}}\#..\underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}}\underbrace{\#\$}_{v'_{k+2}}$$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1}\underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}}..\#\underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}}\underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1}\underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}}\#..\underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}}\underbrace{\#\$}_{v'_{k+2}}$$

$$\Rightarrow 1, i_2{+}1, .., i_n{+}1, k{+}2 \text{ löst } f(K) \text{ also } f(K) \in PKP$$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1}\underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}}..\#\underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}}\underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1}\underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}}\#..\underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}}\underbrace{\#\$}_{v'_{k+2}}$$

$$\Rightarrow 1, i_2{+}1, .., i_n{+}1, k{+}2 \text{ löst } f(K) \text{ also } f(K) \in PKP$$

  - $f(K) \in PKP$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1}\underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}}..\#\underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}}\underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1}\underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}}\#..\underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}}\underbrace{\#\$}_{v'_{k+2}}$$

$$\Rightarrow 1, i_2{+}1, .., i_n{+}1, k{+}2 \text{ löst } f(K) \text{ also } f(K) \in PKP$$

  - $f(K) \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u'_{i_1}..u'_{i_n} = v'_{i_1}..v'_{i_n}$

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1}\underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}}..\#\underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}}\underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1}\underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}}\#..\underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}}\underbrace{\#\$}_{v'_{k+2}}$$

$\Rightarrow 1, i_2{+}1, .., i_n{+}1, k{+}2$ löst $f(K)$ also $f(K) \in PKP$

  - $f(K) \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u'_{i_1}..u'_{i_n} = v'_{i_1}..v'_{i_n}$

$\Rightarrow i_1{=}1$, da nur u'_1, v'_1 dasselbe Anfangssymbol haben

$i_n{=}k{+}2$, da nur u'_{k+2}, v'_{k+2} dasselbe Endsymbol haben

$MPKP \leq PKP$

- **Abbildung f erzwingt erstes Wortpaar als Anfang**
  - Erweitere Alphabet X zu $X' = X \cup \{\#, \$\}$
  - Modifiziere Worte $w = a_1...a_n$ zu $\hat{w} = a_1\#a_2\#...\#a_n$
  - Definiere Abbildung f durch

$$f\{(u_1, v_1), .., (u_k, v_k)\} = \{\underbrace{(\#\hat{u}_1\#, \#\hat{v}_1)}_{(u'_1,v'_1)}, \underbrace{(\hat{u}_1\#, \#\hat{v}_1)}_{(u'_2,v'_2)}, .. \underbrace{(\hat{u}_k\#, \#\hat{v}_k)}_{(u'_{k+1},v'_{k+1})}, \underbrace{(\$, \#\$)}_{(u'_{k+2},v'_{k+2})} \}$$

- **Zeige $K \in MPKP \Leftrightarrow f(K) \in PKP$**
  - $K \in MPKP \Rightarrow$ Es gibt $i_2, .., i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$

$$\Rightarrow \underbrace{\#\hat{u}_1\#}_{u'_1}\underbrace{\hat{u}_{i_2}\#}_{u'_{i_2+1}}..\#\underbrace{\hat{u}_{i_n}\#}_{u'_{i_n+1}}\underbrace{\$}_{u'_{k+2}} = \underbrace{\#\hat{v}_1}_{v'_1}\underbrace{\#\hat{v}_{i_2}}_{v'_{i_2+1}}\#..\underbrace{\#\hat{v}_{i_n}}_{v'_{i_n+1}}\underbrace{\#\$}_{v'_{k+2}}$$

$\Rightarrow 1, i_2{+}1, .., i_n{+}1, k{+}2$ löst $f(K)$ also $f(K) \in PKP$

  - $f(K) \in PKP \Rightarrow$ Es gibt $i_1, .., i_n$ mit $u'_{i_1}..u'_{i_n} = v'_{i_1}..v'_{i_n}$

$\Rightarrow i_1{=}1$, da nur u'_1, v'_1 dasselbe Anfangssymbol haben

$i_n{=}k{+}2$, da nur u'_{k+2}, v'_{k+2} dasselbe Endsymbol haben

$\Rightarrow 1, i_2{-}1, .., i_{n-1}{-}1$ löst K also $K \in MPKP$ ✓

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ,w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ,w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

– Wähle Alphabet $Y := \Gamma \cup S \cup \{\#\}$ für das MPKP

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ, w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

– Wähle Alphabet $Y := \Gamma \cup S \cup \{\#\}$ für das MPKP

– Erzeuge Anfangskonfiguration in $(u_1, v_1) := (\#, \#s_0 w \#)$

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ, w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

– Wähle Alphabet $Y := \Gamma \cup S \cup \{\#\}$ für das MPKP

– Erzeuge Anfangskonfiguration in $(u_1, v_1) := (\#, \#s_0 w \#)$

– Beschreibe Konfigurationsübergänge durch Wortpaare

$\cdot\ (s\,a,\ a'\,s')$	für $\delta(s,a) = (s', a', r)$
$\cdot\ (s\,\#,\ a'\,s'\#)$	für $\delta(s,b) = (s', a', r)$
$\cdot\ (d\,s\,a,\ s'\,d\,a')$	für $d \in \Gamma$ und $\delta(s,a) = (s', a', l)$
$\cdot\ (\#\,s\,a,\ \#\,s'\,b\,a')$	für $\delta(s,a) = (s', a', l)$
$\cdot\ (d\,s\,\#,\ s'\,d\,a'\,\#)$	für $d \in \Gamma$ und $\delta(s,b) = (s', a', r)$
$\cdot\ (s\,a,\ q_f\,a')$	für $\delta(s,a) = (s', a', h)$

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ, w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

– Wähle Alphabet $Y := \Gamma \cup S \cup \{\#\}$ für das MPKP

– Erzeuge Anfangskonfiguration in $(u_1, v_1) := (\#,\ \#s_0 w \#)$

– Beschreibe Konfigurationsübergänge durch Wortpaare

- $(s\,a,\ a'\,s')$ für $\delta(s,a) = (s', a', r)$
- $(s\,\#,\ a'\,s'\#)$ für $\delta(s,b) = (s', a', r)$
- $(d\,s\,a,\ s'\,d\,a')$ für $d \in \Gamma$ und $\delta(s,a) = (s', a', l)$
- $(\#\,s\,a,\ \#\,s'\,b\,a')$ für $\delta(s,a) = (s', a', l)$
- $(d\,s\,\#,\ s'\,d\,a'\,\#)$ für $d \in \Gamma$ und $\delta(s,b) = (s', a', r)$
- $(s\,a,\ q_f\,a')$ für $\delta(s,a) = (s', a', h)$

– Ergänze "Kopierregeln" (a, a) für alle $a \in \Gamma \cup \{\#\}$

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ, w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

– Wähle Alphabet $Y := \Gamma \cup S \cup \{\#\}$ für das MPKP

– Erzeuge Anfangskonfiguration in $(u_1, v_1) := (\#, \#s_0 w \#)$

– Beschreibe Konfigurationsübergänge durch Wortpaare

- $(s\,a,\ a'\,s')$ für $\delta(s,a) = (s', a', r)$
- $(s\,\#,\ a'\,s'\#)$ für $\delta(s,b) = (s', a', r)$
- $(d\,s\,a,\ s'\,d\,a')$ für $d \in \Gamma$ und $\delta(s,a) = (s', a', l)$
- $(\#\,s\,a,\ \#\,s'\,b\,a')$ für $\delta(s,a) = (s', a', l)$
- $(d\,s\,\#,\ s'\,d\,a'\,\#)$ für $d \in \Gamma$ und $\delta(s,b) = (s', a', r)$
- $(s\,a,\ q_f\,a')$ für $\delta(s,a) = (s', a', h)$

– Ergänze "Kopierregeln" (a, a) für alle $a \in \Gamma \cup \{\#\}$

– Ergänze "Löschregeln" $(a\,q_f, q_f)$ und $(q_f\,a, q_f)$ für alle $a \in \Gamma$

$H \leq MPKP$: Transformation

Transformiere ein Halteproblem τ, w in ein MPKP K

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$, $w \in X^*$. Bestimme $K := f(\tau, w)$ wie folgt

– Wähle Alphabet $Y := \Gamma \cup S \cup \{\#\}$ für das MPKP

– Erzeuge Anfangskonfiguration in $(u_1, v_1) := (\#, \#s_0 w \#)$

– Beschreibe Konfigurationsübergänge durch Wortpaare

- $(s\,a,\ a'\,s')$ für $\delta(s,a) = (s', a', r)$
- $(s\,\#,\ a'\,s'\#)$ für $\delta(s,b) = (s', a', r)$
- $(d\,s\,a,\ s'\,d\,a')$ für $d \in \Gamma$ und $\delta(s,a) = (s', a', l)$
- $(\#\,s\,a,\ \#\,s'\,b\,a')$ für $\delta(s,a) = (s', a', l)$
- $(d\,s\,\#,\ s'\,d\,a'\,\#)$ für $d \in \Gamma$ und $\delta(s,b) = (s', a', r)$
- $(s\,a,\ q_f\,a')$ für $\delta(s,a) = (s', a', h)$

– Ergänze "Kopierregeln" (a, a) für alle $a \in \Gamma \cup \{\#\}$

– Ergänze "Löschregeln" $(a\,q_f, q_f)$ und $(q_f\,a, q_f)$ für alle $a \in \Gamma$

– Ergänze "Abschlußregel" $(q_f\,\#\,\#,\ \#)$

$H \leq MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w) \neq \perp \Rightarrow f(\tau, w) \in MPKP$

$H \leq MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w) \neq \perp \Rightarrow f(\tau, w) \in MPKP$

– Wegen $h_\tau(w) \neq \perp$ gibt es eine Konfigurationsfolge

$\kappa_0 = s_0 w \rightarrow \kappa_1 \ldots \rightarrow \kappa_t = x_m..x_0\, s\, y_0..y_n$ mit $\delta(\_, \_) = (s, y_0, h)$

$H{\leq}MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w){\neq}\perp \;\Rightarrow\; f(\tau,w)\in MPKP$

– Wegen $h_\tau(w){\neq}\perp$ gibt es eine Konfigurationsfolge

$\kappa_0{=}s_0w \;\rightarrow\; \kappa_1 \;\ldots\rightarrow\; \kappa_t{=}x_m..x_0\, s\, y_0..y_n$ mit $\delta(\_,\_)=(s,y_0,h)$

– Mit Überführungs- und Kopierregeln bauen wir folgendes Wortpaar auf

· $u = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1$

· $v = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0, q_f\, y_0..y_n\#x_m..x_1$

$H{\leq}MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w){\neq}\perp \Rightarrow f(\tau,w)\in MPKP$

– Wegen $h_\tau(w){\neq}\perp$ gibt es eine Konfigurationsfolge

$\kappa_0{=}s_0w \to \kappa_1 \ldots \to \kappa_t{=}x_m..x_0\, s\, y_0..y_n$ mit $\delta(\_,\_) = (s, y_0, h)$

– Mit Überführungs- und Kopierregeln bauen wir folgendes Wortpaar auf

· $u = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1$

· $v = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0, q_f\, y_0..y_n\#x_m..x_1$

– Mit der Löschregel $(x_0\, q_f, q_f)$ erzeugen wir daraus

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f$

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#x_m..x_1q_f$

$H \leq MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w) \neq \bot \Rightarrow f(\tau, w) \in MPKP$

– Wegen $h_\tau(w) \neq \bot$ gibt es eine Konfigurationsfolge

$\kappa_0 = s_0 w \to \kappa_1 \ldots \to \kappa_t = x_m..x_0\, s\, y_0..y_n$ mit $\delta(\_, \_) = (s, y_0, h)$

– Mit Überführungs- und Kopierregeln bauen wir folgendes Wortpaar auf

· $u = \#s_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_m..x_1$

· $v = \#s_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_m..x_1 x_0, q_f\, y_0..y_n \# x_m..x_1$

– Mit der Löschregel $(x_0\, q_f, q_f)$ erzeugen wir daraus

· $\#s_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_m..x_1 x_0\, q_f$

· $\#s_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_m..x_1 x_0\, q_f\, y_0..y_n \# x_m..x_1 q_f$

– Mit den Kopierregeln bekommen wir

· $\#s_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_m..x_1 x_0\, q_f\, y_0..y_n \#$

· $\#s_0 w \# \kappa_1 \# \ldots \# \kappa_t \# x_m..x_1 x_0\, q_f\, y_0..y_n \# x_m..x_1 q_f\, y_0..y_n \#$

$H{\leq}MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w){\neq}\bot \Rightarrow f(\tau,w)\in MPKP$

– Wegen $h_\tau(w){\neq}\bot$ gibt es eine Konfigurationsfolge

$\kappa_0{=}s_0w \to \kappa_1 \ldots \to \kappa_t{=}x_m..x_0\, s\, y_0..y_n$ mit $\delta(\_,\_)=(s,y_0,h)$

– Mit Überführungs- und Kopierregeln bauen wir folgendes Wortpaar auf

· $u = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1$

· $v = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0, q_f\, y_0..y_n\#x_m..x_1$

– Mit der Löschregel $(x_0\, q_f, q_f)$ erzeugen wir daraus

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f$

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#x_m..x_1q_f$

– Mit den Kopierregeln bekommen wir

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#$

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#x_m..x_1q_f\, y_0..y_n\#$

– Mit den Lösch- und Kopierregeln ergibt sich

· $\#s_0w\#\ldots\ldots\#x_m..x_1q_f\, y_0..y_n\#x_m..x_2q_f\, y_0..y_n\#\ldots\#q_f\, y_n\#$

· $\#s_0w\#\ldots\ldots\#x_m..x_1q_f\, y_0..y_n\#x_m..x_2q_f\, y_0..y_n\#\ldots\#q_f\, y_n\#q_f\#$

$H{\leq}MPKP$: Korrektheit der Transformation f

Zeige $h_\tau(w){\neq}\bot \Rightarrow f(\tau,w)\in MPKP$

– Wegen $h_\tau(w){\neq}\bot$ gibt es eine Konfigurationsfolge

$\kappa_0{=}s_0w \to \kappa_1 \ldots \to \kappa_t{=}x_m..x_0\, s\, y_0..y_n$ mit $\delta(\_,\_) = (s, y_0, h)$

– Mit Überführungs- und Kopierregeln bauen wir folgendes Wortpaar auf

· $u = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1$

· $v = \#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0, q_f\, y_0..y_n\#x_m..x_1$

– Mit der Löschregel $(x_0\, q_f, q_f)$ erzeugen wir daraus

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f$

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#x_m..x_1q_f$

– Mit den Kopierregeln bekommen wir

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#$

· $\#s_0w\#\kappa_1\#\ldots\#\kappa_t\#x_m..x_1x_0\, q_f\, y_0..y_n\#x_m..x_1q_f\, y_0..y_n\#$

– Mit den Lösch- und Kopierregeln ergibt sich

· $\#s_0w\#\ldots\ldots\#x_m..x_1q_f\, y_0..y_n\#x_m..x_2q_f\, y_0..y_n\#\ldots\#q_f\, y_n\#$

· $\#s_0w\#\ldots\ldots\#x_m..x_1q_f\, y_0..y_n\#x_m..x_2q_f\, y_0..y_n\#\ldots\#q_f\, y_n\#q_f\#$

– Mit der Abschlußregel ergibt sich schließlich

· $\#s_0w\#\ldots\ldots\#x_m..x_1q_f\, y_0..y_n\#x_m..x_2q_f\, y_0..y_n\#\ldots\#q_f\, y_n\#q_f\#\#$

· $\#s_0w\#\ldots\ldots\#x_m..x_1q_f\, y_0..y_n\#x_m..x_2q_f\, y_0..y_n\#\ldots\#q_f\, y_n\#q_f\#\#$ ✓

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

– Es gelte $f(\tau, w) \in MPKP$

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

– Es gelte $f(\tau, w) \in MPKP$

– Also gibt es $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

– Es gelte $f(\tau, w) \in MPKP$

– Also gibt es $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$

– $u_1 u_{i_2} .. u_{i_n}$ muß mit $\#s_0 w\#$ beginnen und mit $q_f \# \#$ enden

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

- Es gelte $f(\tau, w) \in MPKP$
- Also gibt es $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$
- $u_1 u_{i_2} .. u_{i_n}$ muß mit $\#s_0 w\#$ beginnen und mit $q_f \# \#$ enden
- Wegen der Überführungsregeln gilt $\#u_{i_2} \hat{=} v_1$, $\#u_{i_3} \hat{=} v_{i_2}$, ...

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

- Es gelte $f(\tau, w) \in MPKP$
- Also gibt es $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$
- $u_1 u_{i_2} .. u_{i_n}$ muß mit $\#s_0 w\#$ beginnen und mit $q_f \# \#$ enden
- Wegen der Überführungsregeln gilt $\#u_{i_2} \hat{=} v_1$, $\#u_{i_3} \hat{=} v_{i_2}$, ...
- Aus der Korrespondenz können wir daher eine Konfigurationsfolge

  $\kappa_0 = s_0 w \rightarrow \ldots \rightarrow \kappa_t = x_m .. x_0\, s\, y_0 .. y_n$ konstruieren mit $\delta(\_, \_) = (s, y_0, h)$

$H \leq MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w) \neq \perp \Leftarrow f(\tau, w) \in MPKP$

- Es gelte $f(\tau, w) \in MPKP$
- Also gibt es $i_2, .., i_n$ mit $u_1 u_{i_2} .. u_{i_n} = v_1 v_{i_2} .. v_{i_n}$
- $u_1 u_{i_2} .. u_{i_n}$ muß mit $\#s_0 w\#$ beginnen und mit $q_f \# \#$ enden
- Wegen der Überführungsregeln gilt $\#u_{i_2} \hat{=} v_1$, $\#u_{i_3} \hat{=} v_{i_2}$, ...
- Aus der Korrespondenz können wir daher eine Konfigurationsfolge
  $\kappa_0 = s_0 w \rightarrow \ldots \rightarrow \kappa_t = x_m .. x_0\, s\, y_0 .. y_n$ konstruieren mit $\delta(\_, \_) = (s, y_0, h)$
- Also hält τ bei Eingabe w ✓

$H{\leq}MPKP$: Korrektheit der Transformation f (II)

Zeige $h_\tau(w){\neq}\perp \Leftarrow f(\tau,w){\in}MPKP$

- Es gelte $f(\tau,w){\in}MPKP$
- Also gibt es $i_2,..,i_n$ mit $u_1u_{i_2}..u_{i_n} = v_1v_{i_2}..v_{i_n}$
- $u_1u_{i_2}..u_{i_n}$ muß mit $\#s_0w\#$ beginnen und mit $q_f\,\#\,\#$ enden
- Wegen der Überführungsregeln gilt $\#u_{i_2}{\hat{=}}v_1$, $\#u_{i_3}{\hat{=}}v_{i_2}$, ...
- Aus der Korrespondenz können wir daher eine Konfigurationsfolge
  $\kappa_0{=}s_0w \rightarrow \ldots \rightarrow \kappa_t{=}x_m..x_0\,s\,y_0..y_n$ konstruieren mit $\delta(\_,\_) = (s,y_0,h)$
- Also hält τ bei Eingabe w ✓

Es folgt $H{\leq}MPKP$, also ist $MPKP$ unentscheidbar

Für kontextfreie Grammatiken G, G' sind die folgende Probleme unentscheidbar

1. $L(G) \cap L(G') = \emptyset$
2. $L(G) \cap L(G')$ unendlich
3. $L(G) \cap L(G')$ kontextfrei
4. $L(G) \subseteq L(G')$
5. $L(G) = L(G')$
6. $L(G) = X^*$
7. G mehrdeutig
8. $\overline{L(G)}$ kontextfrei
9. $L(G)$ regulär
10. $L(G) \in$ DPDA

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,

$$A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$$
$$B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$$

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G :=$ $S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' :=$ $S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,
  - Dann gilt $L(G)' = \{uv \$ \overline{uv} \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,
  - Dann gilt $L(G)' = \{uv \$ \overline{uv} \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$
- $L(G) \cap L(G') = \emptyset$ **ist unentscheidbar**

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,
  - Dann gilt $L(G)' = \{uv \$ \overline{uv} \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$
- **$L(G) \cap L(G') = \emptyset$ ist unentscheidbar**
  - Folgt direkt aus $K \in PKP \Leftrightarrow L(G) \cap L(G') \neq \emptyset$

✓

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,
  - Dann gilt $L(G)' = \{uv\,\$\,\overline{uv} \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$
- $L(G) \cap L(G') = \emptyset$ **ist unentscheidbar**
  - Folgt direkt aus $K \in PKP \Leftrightarrow L(G) \cap L(G') \neq \emptyset$ ✓
- $L(G) \cap L(G')$ **unendlich ist unentscheidbar**

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,
  - Dann gilt $L(G)' = \{uv \$ \overline{uv} \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$
- $L(G) \cap L(G') = \emptyset$ **ist unentscheidbar**
  - Folgt direkt aus $K \in PKP \Leftrightarrow L(G) \cap L(G') \neq \emptyset$ ✓
- $L(G) \cap L(G')$ **unendlich ist unentscheidbar**
  - Es gilt $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n} \Rightarrow u_{i_1}..u_{i_n} u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n} v_{i_1}..v_{i_n} \Rightarrow \ldots$

Reduktion auf das Post'sche Korrespondenzproblem

- **Transformiere ein PKP in Grammatiken G und G'**
  - Gegeben $K = \{(u_1, v_1), .., (u_k, v_k)\}$ über $X = \{0, 1\}$
  - Wähle Terminalalphabet $:= \{0, 1, \$, a_1, .., a_k\}$
  - Konstruiere $G := S \to A\$B$,
    $A \to a_1 A u_1, \ldots A \to a_k A u_k,\ A \to a_1 u_1, \ldots A \to a_k u_k$
    $B \to \overline{v_1} B a_1, \ldots B \to \overline{v_k} B a_k,\ B \to \overline{v_1} a_1, \ldots B \to \overline{v_k} a_k$
  - Dann gilt $L(G) = \{a_{i_n}..a_{i_1} u_{i_1}..u_{i_n} \$ \overline{v_{j_m}}..\overline{v_{j_1}} a_{j_1}..a_{j_m} \mid i_\nu, j_\mu \leq k\}$
  - Konstruiere $G' := S \to a_1 S a_1, .., S \to a_k S a_k,\ S \to T$,
    $T \to 0T0,\ T \to 1T1,\ T \to \$$,
  - Dann gilt $L(G)' = \{uv \$ \overline{uv} \mid u \in \{a_1, .., a_k\}^*, v \in \{0, 1\}^*\}$
- $L(G) \cap L(G') = \emptyset$ **ist unentscheidbar**
  - Folgt direkt aus $K \in PKP \Leftrightarrow L(G) \cap L(G') \neq \emptyset$ ✓
- $L(G) \cap L(G')$ **unendlich ist unentscheidbar**
  - Es gilt $u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n} \Rightarrow u_{i_1}..u_{i_n} u_{i_1}..u_{i_n} = v_{i_1}..v_{i_n} v_{i_1}..v_{i_n} \Rightarrow \ldots$
  - Es folgt $K \in PKP \Leftrightarrow L(G) \cap L(G')$ unendlich ✓

Problemreduktion mit Abschlusseigenschaften

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit

Problemreduktion mit Abschlusseigenschaften

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
- **Methodik**
  - Zeige, daß Lösung für P ein unlösbares Problem P' lösen würde

Problemreduktion mit Abschlusseigenschaften

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
- **Methodik**
  - Zeige, daß Lösung für P ein unlösbares Problem P' lösen würde
  - Zeige, wie Lösung eines bekannten Problems P' zur Lösung von P' führt

Problemreduktion mit Abschlusseigenschaften

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
- **Methodik**
  - Zeige, daß Lösung für P ein unlösbares Problem P' lösen würde
  - Zeige, wie Lösung eines bekannten Problems P' zur Lösung von P' führt
- **Hilfsmittel: Abschlußeigenschaften**
  - M, M' entscheidbar, dann auch $M \cup M'$, $M \cap M'$, $M \setminus M'$, $\overline{M}$, $f^{-1}(M)$
  - M, M' aufzählbar, dann auch $M \cup M'$, $M \cap M'$, $g(M)$, $g^{-1}(M)$
  - M entscheidbar $\Leftrightarrow$ M und $\overline{M}$ aufzählbar
  - Reduzierbarkeit ist ein besonders mächtiger Spezialfall

Problemreduktion mit Abschlusseigenschaften

- **Ziel: Wiederverwendung bekannter Ergebnisse**
  - Zur Lösung eines Problems P bzw. zum Nachweis seiner Unlösbarkeit
- **Methodik**
  - Zeige, daß Lösung für P ein unlösbares Problem P' lösen würde
  - Zeige, wie Lösung eines bekannten Problems P' zur Lösung von P' führt
- **Hilfsmittel: Abschlußeigenschaften**
  - M, M' entscheidbar, dann auch $M \cup M'$, $M \cap M'$, $M \setminus M'$, $\overline{M}$, $f^{-1}(M)$
  - M, M' aufzählbar, dann auch $M \cup M'$, $M \cap M'$, $g(M)$, $g^{-1}(M)$
  - M entscheidbar $\Leftrightarrow$ M und $\overline{M}$ aufzählbar
  - Reduzierbarkeit ist ein besonders mächtiger Spezialfall
- **Umkehrung der Abschlußeigenschaften**
  - M nicht entscheidbar $\Rightarrow$ $\overline{M}$ nicht entscheidbar
  - M aufzählbar, nicht entscheidbar $\Rightarrow$ $\overline{M}$ weder aufzählbar noch entscheidbar
  - M entscheidbar, $M \cup M'$ nicht entscheidbar $\Rightarrow$ M' nicht entscheidbar
  - M entscheidbar, $M \setminus M'$ nicht entscheidbar $\Rightarrow$ M' nicht entscheidbar

$\vdots$ $\qquad$ $\vdots$

Resultate aus Abschlusseigenschaften

- **$\{(i,n) \mid \varphi_i(n){=}\perp\}$ ist nicht aufzählbar** Satz J.b
  - $\{(i,n) \mid \varphi_i(n){=}\perp\}$ ist das Komplement von $H = \{(i,n) \mid \varphi_i(n){\neq}\perp\}$
  - H ist aufzählbar aber unentscheidbar, also kann $\overline{H}$ nicht aufzählbar sein

Resultate aus Abschlusseigenschaften

- **$\{(i,n) \mid \varphi_i(n){=}\perp\}$ ist nicht aufzählbar** Satz J.b
  - $\{(i,n) \mid \varphi_i(n){=}\perp\}$ ist das Komplement von $H = \{(i,n) \mid \varphi_i(n){\neq}\perp\}$
  - H ist aufzählbar aber unentscheidbar, also kann $\overline{H}$ nicht aufzählbar sein
- **$\{i \mid \varphi_i(i){=}\perp\}$ ist nicht aufzählbar**
  - $\{i \mid \varphi_i(i){=}\perp\}$ ist das Komplement von $S = \{i \mid \varphi_i(i){\neq}\perp\}$
  - S ist aufzählbar aber unentscheidbar, also kann $\overline{S}$ nicht aufzählbar sein

Resultate aus Abschlusseigenschaften

- **$\{(i,n) \mid \varphi_i(n){=}\perp\}$ ist nicht aufzählbar** Satz J.b
  - $\{(i,n) \mid \varphi_i(n){=}\perp\}$ ist das Komplement von $H = \{(i,n) \mid \varphi_i(n){\neq}\perp\}$
  - H ist aufzählbar aber unentscheidbar, also kann $\overline{H}$ nicht aufzählbar sein
- **$\{i \mid \varphi_i(i){=}\perp\}$ ist nicht aufzählbar**
  - $\{i \mid \varphi_i(i){=}\perp\}$ ist das Komplement von $S = \{i \mid \varphi_i(i){\neq}\perp\}$
  - S ist aufzählbar aber unentscheidbar, also kann $\overline{S}$ nicht aufzählbar sein
- **$PROG_z = \{i \mid \varphi_i = z\}$ ist nicht aufzählbar**
  - Es gilt $\overline{H} \leq PROG_z$ und $\overline{H}$ ist nicht aufzählbar

Resultate aus Abschlusseigenschaften

- **$\{(i,n) \mid \varphi_i(n){=}\bot\}$ ist nicht aufzählbar** Satz J.b
  - $\{(i,n) \mid \varphi_i(n){=}\bot\}$ ist das Komplement von $H = \{(i,n) \mid \varphi_i(n){\neq}\bot\}$
  - H ist aufzählbar aber unentscheidbar, also kann $\overline{H}$ nicht aufzählbar sein
- **$\{i \mid \varphi_i(i){=}\bot\}$ ist nicht aufzählbar**
  - $\{i \mid \varphi_i(i){=}\bot\}$ ist das Komplement von $S = \{i \mid \varphi_i(i){\neq}\bot\}$
  - S ist aufzählbar aber unentscheidbar, also kann $\overline{S}$ nicht aufzählbar sein
- **$PROG_z = \{i \mid \varphi_i = z\}$ ist nicht aufzählbar**
  - Es gilt $\overline{H} \leq PROG_z$ und $\overline{H}$ ist nicht aufzählbar
- **$PF_\varphi = \{i \mid \varphi_i$ partiell$\}$ ist unentscheidbar**
  - PF_φ ist das Komplement von $\mathcal{R}_\varphi = \{i \mid \varphi_i$ total$\}$
  - $\mathcal{R}_\varphi$ ist nicht aufzählbar, also kann $PF_\varphi = \overline{\mathcal{R}_\varphi}$ nicht entscheidbar sein

Welche Verifikationsprobleme sind entscheidbar?

- **Halteproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es bei bestimmten Eingaben hält oder nicht?
  - Bereits als unentscheidbar nachgewiesen

Welche Verifikationsprobleme sind entscheidbar?

- **Halteproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es bei bestimmten Eingaben hält oder nicht?
  - Bereits als unentscheidbar nachgewiesen
- **Korrektheitsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine bestimmte Funktion berechnet oder nicht?
  - Für die Nullfunktion bereits als unentscheidbar nachgewiesen
  - Gilt ähnliches für andere Funktionen?

Welche Verifikationsprobleme sind entscheidbar?

- **Halteproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es bei bestimmten Eingaben hält oder nicht?
  - Bereits als unentscheidbar nachgewiesen
- **Korrektheitsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine bestimmte Funktion berechnet oder nicht?
  - Für die Nullfunktion bereits als unentscheidbar nachgewiesen
  - Gilt ähnliches für andere Funktionen?
- **Spezifikationsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine gegebene Spezfikation erfüllt oder nicht?

Welche Verifikationsprobleme sind entscheidbar?

- **Halteproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es bei bestimmten Eingaben hält oder nicht?
  - Bereits als unentscheidbar nachgewiesen
- **Korrektheitsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine bestimmte Funktion berechnet oder nicht?
  - Für die Nullfunktion bereits als unentscheidbar nachgewiesen
  - Gilt ähnliches für andere Funktionen?
- **Spezifikationsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine gegebene Spezfikation erfüllt oder nicht?
- **Äquivalenzproblem**
  - Kann man entscheiden, ob zwei beliebige Programme die gleiche Funktion berechnen oder nicht?

Welche Verifikationsprobleme sind entscheidbar?

- **Halteproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es bei bestimmten Eingaben hält oder nicht?
  - Bereits als unentscheidbar nachgewiesen
- **Korrektheitsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine bestimmte Funktion berechnet oder nicht?
  - Für die Nullfunktion bereits als unentscheidbar nachgewiesen
  - Gilt ähnliches für andere Funktionen?
- **Spezifikationsproblem**
  - Kann man von einem beliebigen Programm entscheiden, ob es eine gegebene Spezfikation erfüllt oder nicht?
- **Äquivalenzproblem**
  - Kann man entscheiden, ob zwei beliebige Programme die gleiche Funktion berechnen oder nicht?

Gibt es eine allgemeine Antwort?

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \bot\}$

– Betrachte $g \equiv \lambda x.\bot$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \bot \\ \bot & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \quad \Rightarrow \quad \forall x.\, \varphi_{f(i)}(x) = h(x)$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \quad \Rightarrow \quad \forall x.\, \varphi_{f(i)}(x) = h(x) \quad \Rightarrow \quad \varphi_{f(i)} = h \in P$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \quad \Rightarrow \quad \forall x.\, \varphi_{f(i)}(x) = h(x) \quad \Rightarrow \quad \varphi_{f(i)} = h \in P \quad \Rightarrow \quad f(i) \in \mathcal{L}_P$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $\begin{matrix} i \in S \\ i \notin S \end{matrix} \quad \Rightarrow \quad \forall x.\, \varphi_{f(i)}(x) = h(x) \quad \Rightarrow \quad \varphi_{f(i)} = h \in P \quad \Rightarrow \quad f(i) \in \mathcal{L}_P$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = h(x) \Rightarrow \varphi_{f(i)} = h \in P \Rightarrow f(i) \in \mathcal{L}_P$

$i \notin S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = \perp$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \bot\}$

– Betrachte $g \equiv \lambda x.\bot$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \bot \\ \bot & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt:
$$\begin{array}{rcccccl} i \in S & \Rightarrow & \forall x.\, \varphi_{f(i)}(x) = h(x) & \Rightarrow & \varphi_{f(i)} = h \in P & \Rightarrow & f(i) \in \mathcal{L}_P \\ i \notin S & \Rightarrow & \forall x.\, \varphi_{f(i)}(x) = \bot & \Rightarrow & \varphi_{f(i)} = g \notin P & & \end{array}$$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \bot\}$

– Betrachte $g \equiv \lambda x.\bot$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \bot \\ \bot & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt:
$$\begin{array}{rcccccl} i \in S & \Rightarrow & \forall x.\, \varphi_{f(i)}(x) = h(x) & \Rightarrow & \varphi_{f(i)} = h \in P & \Rightarrow & f(i) \in \mathcal{L}_P \\ i \notin S & \Rightarrow & \forall x.\, \varphi_{f(i)}(x) = \bot & \Rightarrow & \varphi_{f(i)} = g \notin P & \Rightarrow & f(i) \notin \mathcal{L}_P \end{array}$$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = h(x) \Rightarrow \varphi_{f(i)} = h \in P \Rightarrow f(i) \in \mathcal{L}_P$

$i \notin S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = \perp \Rightarrow \varphi_{f(i)} = g \notin P \Rightarrow f(i) \notin \mathcal{L}_P$

– Insgesamt $i \in S \Leftrightarrow f(i) \in \mathcal{L}_P$, also $S \leq P$

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \bot\}$

– Betrachte $g \equiv \lambda x.\bot$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \bot \\ \bot & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = h(x) \Rightarrow \varphi_{f(i)} = h \in P \Rightarrow f(i) \in \mathcal{L}_P$

$i \notin S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = \bot \Rightarrow \varphi_{f(i)} = g \notin P \Rightarrow f(i) \notin \mathcal{L}_P$

– Insgesamt $i \in S \Leftrightarrow f(i) \in \mathcal{L}_P$, also $S \leq P$

– Da S unentscheidbar ist, muß dies auch für P gelten ✓

Keine nichttriviale extensionale Eigenschaft berechenbarer Funktionen ist entscheidbar

Für $\emptyset \neq P \subset \mathcal{T}_\mu$ ist $\mathcal{L}_P = \{i \mid \varphi_i \in P\}$ nicht entscheidbar

Beweis durch Reduktion auf $S = \{i \mid \varphi_i(i) \neq \perp\}$

– Betrachte $g \equiv \lambda x.\perp$

– Falls $g \notin P$, so wähle $h \in P$ beliebig und definiere

$$h'(i,x) = \begin{cases} h(x) & \text{falls } \varphi_i(i) \neq \perp \\ \perp & \text{sonst} \end{cases}$$

– Dann ist h' berechenbar und nach dem SMN Theorem gibt es ein total-berechenbares f mit $h'(i,x) = \varphi_{f(i)}(x)$

– Es folgt: $i \in S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = h(x) \Rightarrow \varphi_{f(i)} = h \in P \Rightarrow f(i) \in \mathcal{L}_P$

$i \notin S \Rightarrow \forall x.\, \varphi_{f(i)}(x) = \perp \Rightarrow \varphi_{f(i)} = g \notin P \Rightarrow f(i) \notin \mathcal{L}_P$

– Insgesamt $i \in S \Leftrightarrow f(i) \in \mathcal{L}_P$, also $S \leq P$

– Da S unentscheidbar ist, muß dies auch für P gelten ✓

– Falls $g \in S$ wähle ein beliebiges $h \notin S$ und zeige so $S \leq \overline{P}$ ✓

Anwendungen des Satzes von Rice

- $\boldsymbol{MON = \{i \mid \forall k.\, \varphi_i(k) < \varphi_i(k+1)\}}$
  - Monotone Funktionen sind unentscheidbar

Anwendungen des Satzes von Rice

- $\boldsymbol{MON = \{i \mid \forall k.\, \varphi_i(k) < \varphi_i(k+1)\}}$
  - Monotone Funktionen sind unentscheidbar
- $\boldsymbol{EF = \{i \mid \forall j.\, \varphi_i(j) \in \{0,1\}\}}$
  - Entscheidungsfunktionen sind unentscheidbar

Anwendungen des Satzes von Rice

- $\boldsymbol{MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}}$
  - Monotone Funktionen sind unentscheidbar
- $\boldsymbol{EF = \{i \mid \forall j.\ \varphi_i(j) \in \{0,1\}\}}$
  - Entscheidungsfunktionen sind unentscheidbar
- $\boldsymbol{PROG_{spec} = \{i \mid \varphi_i \text{ erfüllt Spezifikation } spec\}}$
  - Allgemeines Spezifikationsproblem ist unentscheidbar

Anwendungen des Satzes von Rice

- $\boldsymbol{MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}}$
  - Monotone Funktionen sind unentscheidbar
- $\boldsymbol{EF = \{i \mid \forall j.\ \varphi_i(j) \in \{0,1\}\}}$
  - Entscheidungsfunktionen sind unentscheidbar
- $\boldsymbol{PROG_{spec} = \{i \mid \varphi_i \text{ erfüllt Spezifikation } spec\}}$
  - Allgemeines Spezifikationsproblem ist unentscheidbar
- $\boldsymbol{PROG_f = \{i \mid \varphi_i = f\}}$
  - Korrektheitsproblem ist unentscheidbar

Anwendungen des Satzes von Rice

- $\boldsymbol{MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}}$
  - Monotone Funktionen sind unentscheidbar
- $\boldsymbol{EF = \{i \mid \forall j.\ \varphi_i(j) \in \{0,1\}\}}$
  - Entscheidungsfunktionen sind unentscheidbar
- $\boldsymbol{PROG_{spec} = \{i \mid \varphi_i}$ **erfüllt Spezifikation** $\boldsymbol{spec\}}$
  - Allgemeines Spezifikationsproblem ist unentscheidbar
- $\boldsymbol{PROG_f = \{i \mid \varphi_i = f\}}$
  - Korrektheitsproblem ist unentscheidbar
- $\boldsymbol{EQ = \{(i,j) \mid \varphi_i = \varphi_j\}}$
  - Äquivalenzproblem ist unentscheidbar

Anwendungen des Satzes von Rice

- $\boldsymbol{MON = \{i \mid \forall k.\, \varphi_i(k) < \varphi_i(k+1)\}}$
  - Monotone Funktionen sind unentscheidbar
- $\boldsymbol{EF = \{i \mid \forall j.\, \varphi_i(j) \in \{0,1\}\}}$
  - Entscheidungsfunktionen sind unentscheidbar
- $\boldsymbol{PROG_{spec} = \{i \mid \varphi_i \text{ erfüllt Spezifikation } spec\}}$
  - Allgemeines Spezifikationsproblem ist unentscheidbar
- $\boldsymbol{PROG_f = \{i \mid \varphi_i = f\}}$
  - Korrektheitsproblem ist unentscheidbar
- $\boldsymbol{EQ = \{(i,j) \mid \varphi_i = \varphi_j\}}$
  - Äquivalenzproblem ist unentscheidbar
- $\boldsymbol{RG = \{(i,j) \mid j \in \mathrm{range}(\varphi_i)\}}$
  - Bildbereiche sind unentscheidbar

Anwendungen des Satzes von Rice

- $MON = \{i \mid \forall k.\ \varphi_i(k) < \varphi_i(k+1)\}$
  - Monotone Funktionen sind unentscheidbar
- $EF = \{i \mid \forall j.\ \varphi_i(j) \in \{0,1\}\}$
  - Entscheidungsfunktionen sind unentscheidbar
- $PROG_{spec} = \{i \mid \varphi_i \text{ erfüllt Spezifikation } spec\}$
  - Allgemeines Spezifikationsproblem ist unentscheidbar
- $PROG_f = \{i \mid \varphi_i = f\}$
  - Korrektheitsproblem ist unentscheidbar
- $EQ = \{(i,j) \mid \varphi_i = \varphi_j\}$
  - Äquivalenzproblem ist unentscheidbar
- $RG = \{(i,j) \mid j \in \mathrm{range}(\varphi_i)\}$
  - Bildbereiche sind unentscheidbar

Keine Programmeigenschaft kann getestet werden
Beweise müssen von Hand geführt werden
Rechnerunterstützung nur in Spezialfällen möglich