

► **Aufgabe 10.1** (Weihnachtsversion): Es seien die beiden Folgen von Tupeln gegeben:

- $K_1 = (\text{🌨️🌲🌲}, \text{🌲👶🌲}, (\text{🌲🌲🌨️}, \text{🌲}), (\text{🌨️🌲👶🌨️}, \text{👶}), (\text{🌲👶}, \text{🌨️🌲👶}), (\text{🌨️}, \text{🌨️🌨️🌲}))$
- $K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$

Zeigen oder widerlegen Sie die folgende Aussagen:

- (i) $K_1 \in PCP$
- (ii) $K_2 \in PCP$

$$K_1 = (\text{❄️🌲🌲}, \text{🌲👶🌲}, (\text{🌲🌲❄️}, \text{🌲}), (\text{❄️🌲👶❄️}, \text{👶}), (\text{🌲👶}, \text{❄️🌲👶}), (\text{❄️}, \text{❄️❄️🌲}))$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$.

$$K_1 = (\text{❄️} \text{🌲} \text{🌲}, \text{🌲} \text{👶} \text{🌲}), (\text{🌲} \text{🌲} \text{❄️}, \text{🌲}), (\text{❄️} \text{🌲} \text{👶} \text{❄️}, \text{👶}), (\text{🌲} \text{👶}, \text{❄️} \text{🌲} \text{👶}), (\text{❄️}, \text{❄️} \text{❄️} \text{🌲})$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:



$$K_1 = (\text{❄️🌲🌲, 🌲👶🌲}, (\text{🌲🌲❄️, 🌲}), (\text{❄️🌲👶❄️, 👶}), (\text{🌲👶, ❄️🌲👶}), (\text{❄️, ❄️❄️🌲}))$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:



$$K_1 = (\text{❄️} \text{🌲} \text{🌲}, \text{🌲} \text{👶}, \text{🌲}), (\text{🌲} \text{🌲} \text{❄️}, \text{🌲}), (\text{❄️} \text{🌲} \text{👶} \text{❄️}, \text{👶}), (\text{🌲} \text{👶}, \text{❄️} \text{🌲} \text{👶}), (\text{❄️}, \text{❄️} \text{❄️} \text{🌲})$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:



$$K_1 = (\text{❄️} \text{🌲} \text{🌲}, \text{🌲} \text{👶} \text{🌲}), (\text{🌲} \text{🌲} \text{❄️}, \text{🌲}), (\text{❄️} \text{🌲} \text{👶} \text{❄️}, \text{👶}), (\text{🌲} \text{👶}, \text{❄️} \text{🌲} \text{👶}), (\text{❄️}, \text{❄️} \text{❄️} \text{🌲})$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:



$$K_1 = (\text{❄️} \text{🌲} \text{🌲}, \text{🌲} \text{👶}, \text{🌲}), (\text{🌲} \text{🌲} \text{❄️}, \text{🌲}), (\text{❄️} \text{🌲} \text{👶} \text{❄️}, \text{👶}), (\text{🌲} \text{👶}, \text{❄️} \text{🌲} \text{👶}), (\text{❄️}, \text{❄️} \text{❄️} \text{🌲})$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:



$$K_1 = (\text{❄️} \text{🌲} \text{🌲}, \text{🌲} \text{👶} \text{🌲}), (\text{🌲} \text{🌲} \text{❄️}, \text{🌲}), (\text{❄️} \text{🌲} \text{👶} \text{❄️}, \text{👶}), (\text{🌲} \text{👶}, \text{❄️} \text{🌲} \text{👶}), (\text{❄️}, \text{❄️} \text{❄️} \text{🌲})$$

$K_1 \in PCP$ mit $i_1 = 5$, $i_2 = 1$, $i_3 = 3$, $i_4 = 5$, $i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:



$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

$K_2 \notin PCP$.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

$K_2 \notin PCP$. Der einzig mögliche Anfang der Folge kann nur das erste Tupel $(101, 10)$ sein, da sich bei allen anderen Tupeln die beiden Komponenten im ersten Symbol unterscheiden.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

$K_2 \notin PCP$. Der einzig mögliche Anfang der Folge kann nur das erste Tupel $(101, 10)$ sein, da sich bei allen anderen Tupeln die beiden Komponenten im ersten Symbol unterscheiden. Darauf muss ein Tupel folgen, dessen zweite Komponente mit 1 beginnt.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

$K_2 \notin PCP$. Der einzig mögliche Anfang der Folge kann nur das erste Tupel $(101, 10)$ sein, da sich bei allen anderen Tupeln die beiden Komponenten im ersten Symbol unterscheiden. Darauf muss ein Tupel folgen, dessen zweite Komponente mit 1 beginnt. Mit dem Tupel $(101, 10)$ ergibt sich allerdings $101\underline{1}01 \dots \neq 101\underline{0} \dots \nmid$.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

$K_2 \notin PCP$. Der einzig mögliche Anfang der Folge kann nur das erste Tupel $(101, 10)$ sein, da sich bei allen anderen Tupeln die beiden Komponenten im ersten Symbol unterscheiden. Darauf muss ein Tupel folgen, dessen zweite Komponente mit 1 beginnt. Mit dem Tupel $(101, 10)$ ergibt sich allerdings $101\underline{1}01 \dots \neq 101\underline{0} \dots$. Also muss das zweite gewählte Tupel $(010, 10)$ sein.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010...

1010...

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Beobachtung: Es muss auf jeden Fall das Tupel **(1, 01)** verwendet werden, da bei allen anderen Tupeln die erste Komponente länger als die zweite ist.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Beobachtung: Es muss auf jeden Fall das Tupel $(1, 01)$ verwendet werden, da bei allen anderen Tupeln die erste Komponente länger als die zweite ist. Weiter kann das genannte Tupel höchstens zweimal hintereinander verwendet werden, da es keine Möglichkeit gibt 111 mit den zweiten Komponenten zu erzeugen.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010101 ...

101010 ...

Angenommen wir wählen als drittes Tupel (101, 10).

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010101 ...

101010 ...

Angenommen wir wählen als drittes Tupel (101, 10). Dann sind Tupel (1, 01), (10, 0) und (101, 10) nicht möglich.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

1010101011 ...

10101001 ...

Angenommen wir wählen als drittes Tupel (101, 10). Dann sind Tupel (1, 01), (10, 0) und (101, 10) nicht möglich.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

10101010110...

1010100...

Angenommen wir wählen als drittes Tupel (101, 10). Dann sind Tupel (1, 01), (10, 0) und (101, 10) nicht möglich.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010101101 ...

10101010 ...

Angenommen wir wählen als drittes Tupel (101, 10). Dann sind Tupel (1, 01), (10, 0) und (101, 10) nicht möglich.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010101010...

10101010...

Angenommen wir wählen als drittes Tupel (101, 10). Dann sind Tupel (1, 01), (10, 0) und (101, 10) nicht möglich. Das vierte Tupel muss also (010, 10) sein.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010101010...

10101010...

Angenommen wir wählen als drittes Tupel (101, 10). Dann sind Tupel (1, 01), (10, 0) und (101, 10) nicht möglich. Das vierte Tupel muss also (010, 10) sein. Hierauf folgt wieder (101, 10) und diese Konstellation wiederholt sich immer wieder.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

101010010 ...

101010 ...

Es ist also nicht möglich, $(101, 10)$ als drittes Tupel zu verwenden. Das heißt, das dritte gewählte Tupel muss $(010, 10)$ sein.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

10101001011 ...

1010100101 ...

Es ist also nicht möglich, $(101, 10)$ als drittes Tupel zu verwenden. Das heißt, das dritte gewählte Tupel muss $(010, 10)$ sein. Nun lassen wir auf dieses Tupel zweimal $(1, 01)$ folgen (einzig sinnvolle Wahl).

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Es ergibt sich:

10101001011010 ...

101010010110 ...

Es ist also nicht möglich, (101, 10) als drittes Tupel zu verwenden. Das heißt, das dritte gewählte Tupel muss (010, 10) sein. Nun lassen wir auf dieses Tupel zweimal (1, 01) folgen (einzig sinnvolle Wahl). Das nächste Tupel muss (010, 10) sein und wir befinden uns wieder in der Anfangssituation.

$$K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$$

Zusammengefasst bedeutet das, dass $K_2 \notin PCP$.

- **Aufgabe 10.2:** Sei 01-PCP die Variante des Postschen Korrespondenzproblems, bei der die Eingabetupel auf das Alphabet $\{0,1\}$ beschränkt sind (d.h. die Wörter der Wortpaarfolge sind nicht-leere Binärstrings). Zeigen Sie, dass diese Variante **unentscheidbar** ist.

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist.

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei eine PCP-Instanz über einem beliebigen Alphabet Σ .

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei eine PCP-Instanz über einem beliebigen Alphabet Σ . Wir müssen nun jedes Symbol $\alpha \in \Sigma$ als **Binärstring** kodieren.

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei eine PCP-Instanz über einem beliebigen Alphabet Σ . Wir müssen nun jedes Symbol $\alpha \in \Sigma$ als **Binärstring** kodieren. Allerdings muss man bei der Kodierung aufpassen, dass keine **“falschen” Lösungen** entstehen.

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei eine PCP-Instanz über einem beliebigen Alphabet Σ . Wir müssen nun jedes Symbol $\alpha \in \Sigma$ als **Binärstring** kodieren. Allerdings muss man bei der Kodierung aufpassen, dass keine **“falschen” Lösungen** entstehen. Würde bspw. ein Symbol α mit 0 kodiert werden und ein zweites Symbol β mit 00, dann wäre $0 \cdot 0 = 00$, allerdings $\alpha \cdot \alpha \neq \beta$.

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei eine PCP-Instanz über einem beliebigen Alphabet Σ . Wir müssen nun jedes Symbol $\alpha \in \Sigma$ als **Binärstring** kodieren. Allerdings muss man bei der Kodierung aufpassen, dass keine **“falschen” Lösungen** entstehen. Würde bspw. ein Symbol α mit 0 kodiert werden und ein zweites Symbol β mit 00, dann wäre $0 \cdot 0 = 00$, allerdings $\alpha \cdot \alpha \neq \beta$. Um diese Problematik zu umgehen, kann man die Kodierung so wählen, dass alle Kodierungen die gleiche Länge haben oder, dass bei allen Kodierungen der “Rand” eines Symbols explizit erkennbar ist.

Die Idee bei dieser Aufgabe ist es, das PCP auf 01-PCP zu **reduzieren**; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei eine PCP-Instanz über einem beliebigen Alphabet Σ . Wir müssen nun jedes Symbol $\alpha \in \Sigma$ als **Binärstring** kodieren. Allerdings muss man bei der Kodierung aufpassen, dass keine **“falschen” Lösungen** entstehen. Würde bspw. ein Symbol α mit 0 kodiert werden und ein zweites Symbol β mit 00, dann wäre $0 \cdot 0 = 00$, allerdings $\alpha \cdot \alpha \neq \beta$. Um diese Problematik zu umgehen, kann man die Kodierung so wählen, dass alle Kodierungen die gleiche Länge haben oder, dass bei allen Kodierungen der “Rand” eines Symbols explizit erkennbar ist. Prinzipiell funktioniert hier jede präfixfreie (oder suffixfreie) Kodierung.

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$.

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$. Wir ersetzen jedes Symbol α_i durch den Binärstring 01^i (also eine Null gefolgt von i Einsen).

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$. Wir ersetzen jedes Symbol α_i durch den Binärstring 01^i (also eine Null gefolgt von i Einsen). Außerdem ersetzen wir in jedem Wort der Wortpaarfolge der gegebenen Problem Instanz die Symbole passend zur Kodierung.

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$. Wir ersetzen jedes Symbol α_i durch den Binärstring 01^i (also eine Null gefolgt von i Einsen). Außerdem ersetzen wir in jedem Wort der Wortpaarfolge der gegebenen Probleminstanz die Symbole passend zur Kodierung. Diese Funktion ist total und berechenbar und erfüllt somit die Eigenschaften einer Reduktionsfunktion.

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$. Wir ersetzen jedes Symbol α_i durch den Binärstring 01^i (also eine Null gefolgt von i Einsen). Außerdem ersetzen wir in jedem Wort der Wortpaarfolge der gegebenen Probleminstance die Symbole passend zur Kodierung. Diese Funktion ist total und berechenbar und erfüllt somit die Eigenschaften einer Reduktionsfunktion. Da die Funktion bijektiv ist, gilt auch offensichtlich, dass die gegebene Probleminstance genau dann eine Lösung hat, wenn auch die kodierte Instanz eine Lösung besitzt.

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$. Wir ersetzen jedes Symbol α_i durch den Binärstring 01^i (also eine Null gefolgt von i Einsen). Außerdem ersetzen wir in jedem Wort der Wortpaarfolge der gegebenen Problem Instanz die Symbole passend zur Kodierung. Diese Funktion ist total und berechenbar und erfüllt somit die Eigenschaften einer Reduktionsfunktion. Da die Funktion bijektiv ist, gilt auch offensichtlich, dass die gegebene Problem Instanz genau dann eine Lösung hat, wenn auch die kodierte Instanz eine Lösung besitzt und somit gilt $\text{PCP} \leq \text{01-PCP}$.

- **Aufgabe 10.3:** Sei 1-PCP die Variante des Postschen Korrespondenzproblems, bei dem man die Eingabetupel auf das Alphabet $\{1\}$ beschränkt. Zeigen Sie, dass 1-PCP **entscheidbar** ist.

Wenn das Alphabet nur aus einem Zeichen besteht, dann kommt es nur auf die **Längen** der einzelnen Teilworte an. Darum kann man das ganze “rechnerisch” entscheiden. Folgender Algorithmus entscheidet das unäre PCP:

1. Wenn für alle Paare (x_i, y_i) gilt: $|x_i| > |y_i|$, dann **lehne ab**.
2. Wenn für alle Paare (x_i, y_i) gilt: $|x_i| < |y_i|$, dann **lehne ab**.
3. Sonst **akzeptiere**.

Im letzten Schritt kann der Algorithmus problemlos akzeptieren, denn:
Entweder es gibt ein Paar (x_i, y_i) mit $|x_i| = |y_i|$. Dann ist dieses Paar eine gültige Lösung.

Im letzten Schritt kann der Algorithmus problemlos akzeptieren, denn:
Entweder es gibt ein Paar (x_i, y_i) mit $|x_i| = |y_i|$. Dann ist dieses Paar eine gültige Lösung. Oder es existieren zwei Paare (x_i, y_i) und (x_j, y_j) , so dass gilt: $|x_i| > |y_i|$ und $|x_j| < |y_j|$.

Im letzten Schritt kann der Algorithmus problemlos akzeptieren, denn: Entweder es gibt ein Paar (x_i, y_i) mit $|x_i| = |y_i|$. Dann ist dieses Paar eine gültige Lösung. Oder es existieren zwei Paare (x_i, y_i) und (x_j, y_j) , so dass gilt: $|x_i| > |y_i|$ und $|x_j| < |y_j|$. Um nun mit den beiden Paaren zwei gleiche Wörter zu bilden müssen wir $n \cdot |x_i| + m \cdot |x_j| = n \cdot |y_i| + m \cdot |y_j|$ lösen.

Im letzten Schritt kann der Algorithmus problemlos akzeptieren, denn: Entweder es gibt ein Paar (x_i, y_i) mit $|x_i| = |y_i|$. Dann ist dieses Paar eine gültige Lösung. Oder es existieren zwei Paare (x_i, y_i) und (x_j, y_j) , so dass gilt: $|x_i| > |y_i|$ und $|x_j| < |y_j|$. Um nun mit den beiden Paaren zwei gleiche Wörter zu bilden müssen wir $n \cdot |x_i| + m \cdot |x_j| = n \cdot |y_i| + m \cdot |y_j|$ lösen. Diese Gleichung ist äquivalent zu $n \cdot (|x_i| - |y_i|) = m \cdot (|y_j| - |x_j|)$

Im letzten Schritt kann der Algorithmus problemlos akzeptieren, denn: Entweder es gibt ein Paar (x_i, y_i) mit $|x_i| = |y_i|$. Dann ist dieses Paar eine gültige Lösung. Oder es existieren zwei Paare (x_i, y_i) und (x_j, y_j) , so dass gilt: $|x_i| > |y_i|$ und $|x_j| < |y_j|$. Um nun mit den beiden Paaren zwei gleiche Wörter zu bilden müssen wir $n \cdot |x_i| + m \cdot |x_j| = n \cdot |y_i| + m \cdot |y_j|$ lösen. Diese Gleichung ist äquivalent zu $n \cdot (|x_i| - |y_i|) = m \cdot (|y_j| - |x_j|)$ und hat eine Lösung mit $n = (|y_j| - |x_j|)$ und $m = (|x_i| - |y_i|)$.

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$x_i^n \cdot x_j^m = y_i^n \cdot y_j^m$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned} x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\ \equiv x_i^{|y_j|-|x_j|} \cdot x_j^{|x_i|-|y_i|} &= y_i^{|y_j|-|x_j|} \cdot y_j^{|x_i|-|y_i|} \end{aligned}$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j| - |x_j|} \cdot x_j^{|x_i| - |y_i|} &= y_i^{|y_j| - |x_j|} \cdot y_j^{|x_i| - |y_i|} \\
 \equiv (|y_j| - |x_j|) \cdot |x_i| + (|x_i| - |y_i|) \cdot |x_j| &= (|y_j| - |x_j|) \cdot |y_i| + (|x_i| - |y_i|) \cdot |y_j|
 \end{aligned}$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j| - |x_j|} \cdot x_j^{|x_i| - |y_i|} &= y_i^{|y_j| - |x_j|} \cdot y_j^{|x_i| - |y_i|} \\
 \equiv (|y_j| - |x_j|) \cdot |x_i| + (|x_i| - |y_i|) \cdot |x_j| &= (|y_j| - |x_j|) \cdot |y_i| + (|x_i| - |y_i|) \cdot |y_j| \\
 \equiv |x_i||y_j| - |x_i||x_j| + |x_i||x_j| - |x_j||y_i| &= |y_i||y_j| - |x_j||y_i| + |x_i||y_j| - |y_i||y_j|
 \end{aligned}$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j| - |x_j|} \cdot x_j^{|x_i| - |y_i|} &= y_i^{|y_j| - |x_j|} \cdot y_j^{|x_i| - |y_i|} \\
 \equiv (|y_j| - |x_j|) \cdot |x_i| + (|x_i| - |y_i|) \cdot |x_j| &= (|y_j| - |x_j|) \cdot |y_i| + (|x_i| - |y_i|) \cdot |y_j| \\
 \equiv |x_i||y_j| - |x_i||x_j| + |x_i||x_j| - |x_j||y_i| &= |y_i||y_j| - |x_j||y_i| + |x_i||y_j| - |y_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= -|x_j||y_i| + |x_i||y_j|
 \end{aligned}$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j|-|x_j|} \cdot x_j^{|x_i|-|y_i|} &= y_i^{|y_j|-|x_j|} \cdot y_j^{|x_i|-|y_i|} \\
 \equiv (|y_j| - |x_j|) \cdot |x_i| + (|x_i| - |y_i|) \cdot |x_j| &= (|y_j| - |x_j|) \cdot |y_i| + (|x_i| - |y_i|) \cdot |y_j| \\
 \equiv |x_i||y_j| - |x_i||x_j| + |x_i||x_j| - |x_j||y_i| &= |y_i||y_j| - |x_j||y_i| + |x_i||y_j| - |y_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= -|x_j||y_i| + |x_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= |x_i||y_j| - |x_j||y_i|
 \end{aligned}$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j|-|x_j|} \cdot x_j^{|x_i|-|y_i|} &= y_i^{|y_j|-|x_j|} \cdot y_j^{|x_i|-|y_i|} \\
 \equiv (|y_j| - |x_j|) \cdot |x_i| + (|x_i| - |y_i|) \cdot |x_j| &= (|y_j| - |x_j|) \cdot |y_i| + (|x_i| - |y_i|) \cdot |y_j| \\
 \equiv |x_i||y_j| - |x_i||x_j| + |x_i||x_j| - |x_j||y_i| &= |y_i||y_j| - |x_j||y_i| + |x_i||y_j| - |y_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= -|x_j||y_i| + |x_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= |x_i||y_j| - |x_j||y_i| \\
 \equiv 1^{|x_i||y_j|-|x_j||y_i|} &= 1^{|x_i||y_j|-|x_j||y_i|}
 \end{aligned}$$

Wenn man nun n -mal Paar (x_i, y_i) und m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j| - |x_j|} \cdot x_j^{|x_i| - |y_i|} &= y_i^{|y_j| - |x_j|} \cdot y_j^{|x_i| - |y_i|} \\
 \equiv (|y_j| - |x_j|) \cdot |x_i| + (|x_i| - |y_i|) \cdot |x_j| &= (|y_j| - |x_j|) \cdot |y_i| + (|x_i| - |y_i|) \cdot |y_j| \\
 \equiv |x_i||y_j| - |x_i||x_j| + |x_i||x_j| - |x_j||y_i| &= |y_i||y_j| - |x_j||y_i| + |x_i||y_j| - |y_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= -|x_j||y_i| + |x_i||y_j| \\
 \equiv |x_i||y_j| - |x_j||y_i| &= |x_i||y_j| - |x_j||y_i| \\
 \equiv 1^{|x_i||y_j| - |x_j||y_i|} &= 1^{|x_i||y_j| - |x_j||y_i|}
 \end{aligned}$$

Da $|x_i| > |x_j| > 0$ und $|y_j| > |y_i| > 0$ ist $1^{|x_i||y_j| - |x_j||y_i|}$ ein gültiges Wort.