Algorithmische Komplexitätstheorie

Prof. Dr. Egon Wanke

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
Algorithmische Komplexitätstheorie

Einführung, Grundlagen, Turingmaschinen (60)

7. Mai 2021

Vorbemerkungen

Vorbemerkungen

Anmerkung 1: Sie studieren grundsätzlich freiwillig. Es besteht weder eine Anwesenheits- noch eine Studiumspflicht. Niemand wird Sie dazu zwingen, den Inhalt dieser Veranstaltung zu erlernen.

Vorbemerkungen

Anmerkung 1: Sie studieren grundsätzlich freiwillig. Es besteht weder eine Anwesenheits- noch eine Studiumspflicht. Niemand wird Sie dazu zwingen, den Inhalt dieser Veranstaltung zu erlernen.

Anmerkung 2: Dieses Folienskript ersetzt kein Lehrbuch. Es kann Ihnen nicht die Teilnahme an den Vorlesungen und Übungen ersparen. Für eine erfolgreiche Teilnahme an der Lehrveranstaltung empfehle ich Ihnen unbedingt die Vorlesungen zu besuchen, in Lehrbüchern zu lesen und die wöchentlichen Übungsaufgaben zu bearbeiten. Sie sollten sich vor allem die Zeit nehmen, den Stoff in aller Ruhe aufzuarbeiten, um die Zusammenhänge zu verstehen.

Literatur

1.) (Diese Bibel, gehört unters Kopfkissen)
M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman, 1979

2.) C.H. Papadimitriou, Computational Complexity, Addision-Wesley, 1994

3.) J.E. Hopcroft und J.D. Ullman, Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie, Addision-Weslay, 1994

4.) I. Wegener, Theoretische Informatik, B.G. Teubner, 1993

... Alle anderen Bücher und Wissensquellen über Theoretische Informatik, die es sonst noch so gibt.

Buchstaben, Alphabete und Sprachen
字母、字母和语言

Ein Alphabet ist eine nicht leere endliche Menge. Wir verwenden für Alphabete in der Regel die Bezeichner Σ oder Γ.

字母表是一个非空的有限集合。

Die Elemente in einem Alphabet nennen wir Buchstaben. Wir verwenden für Buchstaben in der Regel die Bezeichner $a, b, c, d, \ldots$ oder $0, 1, 2, 3, \ldots$.

我们把字母表中的元素称为字母。我们通常使用标识符a，b，c，d，...或0，1，2，3，...表示字母。

Buchstaben lassen sich mit der Konkatenation $\circ$ zu Wörtern verknüpfen. Wir verwenden für Wörter in der Regel die Bezeichner $u, v, w, \ldots$.

字母可以用连词。连接到单词。我们通常用u, v, w, ...来表示词的标识。

Wenn klar ist, was gemeint ist, lassen wir das Konkatenationssymbol $\circ$ einfach weg.如果很清楚是什么意思，我们只需省略连接符号

Buchstaben, Alphabete und Sprachen

Beispiel

Sei $\Sigma = \{0, 1\}$ ein Alphabet mit den Buchstaben 0 und 1.

Sei $u = 0 \circ 1 \circ 1 \circ 0 \circ \ 0 \circ 1$ und $v = 1 \circ 1$, dann ist

$$w = u \circ v = 0 \circ 1 \circ 1 \circ 0 \circ 0 \circ 1 \circ 1 \circ 1.$$

bzw. $u = 011001$, $v = 11$ und $w = uv = 01100111$.

加在后面

Buchstaben, Alphabete und Sprachen

Beispiel

Sei $\Sigma = \{0, 1\}$ ein Alphabet mit den Buchstaben 0 und 1.

Sei $u = 0 \circ 1 \circ 1 \circ 0 \circ \ 0 \circ 1$ und $v = 1 \circ 1$, dann ist

$$w = u \circ v = 0 \circ 1 \circ 1 \circ 0 \circ 0 \circ 1 \circ 1 \circ 1.$$

bzw. $u = 011001$, $v = 11$ und $w = uv = 01100111$.

Die Menge aller endlichen Wörter, die sich mit den Buchstaben aus einem Alphabet Σ bilden lassen, zusammen mit dem leeren Wort ϵ bezeichnen wir mit Σ^*.用字母表Σ中的字母可以组成的所有有限词的集合，加上空词，我们用Σ表示。

Σ^* ist der Kleene Star Abschluss von Σ, bzw. die Kleenesche Hülle von Σ.
Σ是　Σ的Kleene Star终止，或Σ的Kleene H¨ulle。

$$\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$$

Buchstaben, Alphabete und Sprachen

$(\Sigma^*, \circ)$ ist ein Monoid, eine Menge mit assoziativer Verknüpfung $\circ$ und einem neutralem Element ϵ.

$(\Sigma^*, \circ)$是一个单体，一个具有关联联结的集合 $\circ$和一个中性元素 ϵ。

Für Wörter $u, v, w \in \Sigma^*$ gilt $u \circ (v \circ w) = (u \circ v) \circ w$.

Für jedes Wort $u \in \Sigma^*$ gilt $u \circ \epsilon = u = \epsilon \circ u$.

Die Länge $|u|$ eines Wortes u ist die Anzahl seiner Buchstaben aus Σ. Somit gilt $|\epsilon| = 0$.

Weiterhin sei $\Sigma^n = \{u \in \Sigma^* \mid |u| = n\}$, und $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$.

Für $u \in \Sigma^*$ und eine positive Zahl $n \in \mathbb{N}$ ist $u^n = \overbrace{u \circ \cdots \circ u}^{n}$.

Für $n = 0$ ist $u^n = \epsilon$.

Weiterhin gilt $|uv| = |u| + |v|$ und $|w^n| = |w| \cdot n$.

Buchstaben, Alphabete und Sprachen

Eine Teilmenge $L \subseteq \Sigma^*$ von Σ^* ist eine Sprache über Σ.

Σ的一　个子集L⊆s是一个Σ语言。

Für zwei Sprachen $L_1, L_2 \subseteq \Sigma^*$ sei

$$L_1 \circ L_2 = L_1 L_2 = \{uv | u \in L_1, v \in L_2\}.$$

Beispiel

Sei $\Sigma = \{0, 1\}$.

$L_1 = \{0, 1, 00\} \subseteq \Sigma^*$ $\quad L_1 L_1 = \{00, 01, 000, 10, 11, 100, 001, 0000\}$

$L_2 = \overline{L_1} = \Sigma^* \setminus L_1 \subseteq \Sigma^*$ $\quad L_1 L_2 = \Sigma \setminus \{\epsilon, 01, 10, 11, 000, 100\}$

$L_3 = \{0^n 1^n | n \in \mathbb{N}\} \subseteq \Sigma^*$ $\quad L_3 L_4 = \emptyset$

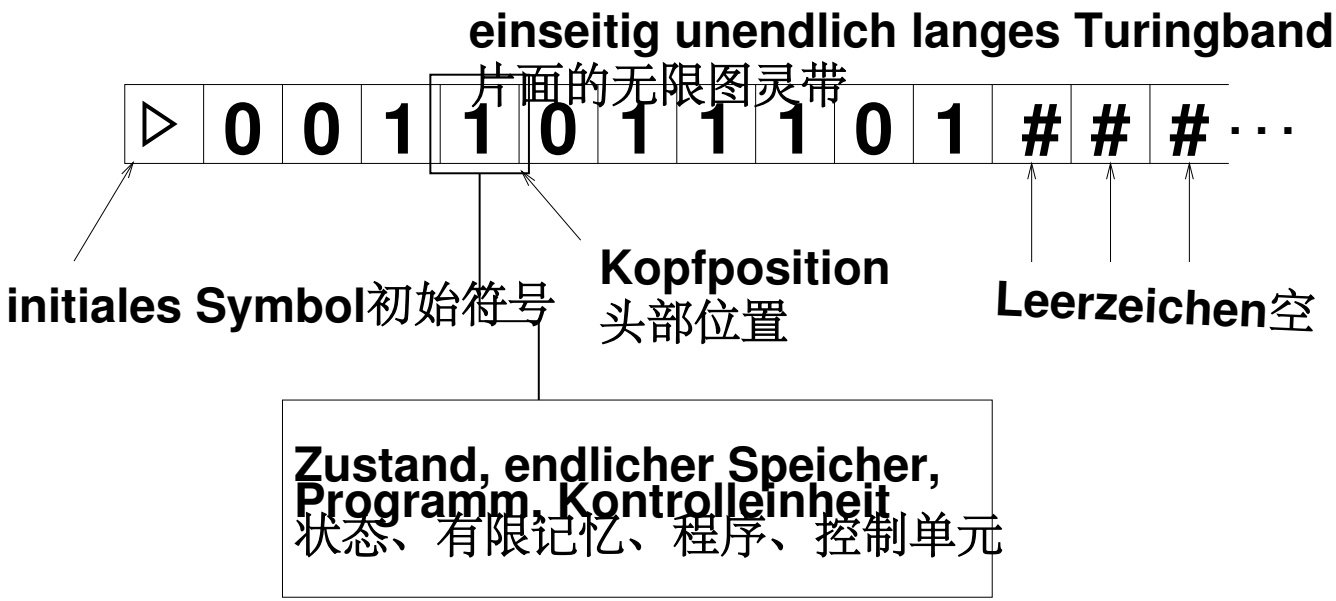
$L_4 = \emptyset \subseteq \Sigma^*$ $\quad L_3 L_5 = L_3$

$L_5 = \{\epsilon\} \subseteq \Sigma^*$

Turingmaschinen

Eine Turingmaschine ist ein einfaches, formales, mathematisches Modell, mit dem die Berechenbarkeit eines Allzweckcomputers modelliert werden kann. Die Turingmaschine wurde 1936 von Alan Turing entwickelt.

Skizze einer Turingmaschine



Turingmaschinen

Definition

Eine Turingmaschine ist ein System $M = (Q, \Sigma, \delta, s)$ mit:

- Q ist eine endliche Menge von Zuständen,
- $s \in Q$ ist der Startzustand,
- Σ ist eine endliche Menge von Symbolen (das Alphabet von M), die immer die beiden Symbole $\#$ (Leerzeichen) und $\triangleright$ (initiales Symbol) enthält, und
- $\delta : Q \times \Sigma \longrightarrow (Q \cup \{h, h_y, h_n\}) \times \Sigma \times \{\rightarrow, \leftarrow, \perp\}$ ist die Übergangsfunktion (das Programm) mit den Haltezuständen

  h allgemeiner Haltezustand,
  h_y akzeptierender Haltezustand und
  h_n ablehnender Haltezustand,

  und den Richtungen

  $\rightarrow$ rechts, $\leftarrow$ links und $\perp$ stehenbleiben.

Motivation

Eine Turingmaschine M arbeitet mit einem Kopf auf einem Band (Turingband).

Das Turingband ist zu Beginn jeder Berechnung mit dem Wort "$\triangleright w$" initialisiert, wobei $w \in (\Sigma - \{\#, \triangleright\})^*$ die Eingabe für M ist.

Der Kopf steht zu Beginn jeder Berechnung auf dem ersten (initialen) Symbol $\triangleright$.

In den Beispielen kennzeichnen wir die Kopfposition durch ein unterstrichenes Symbol.

Einschränkungen und Konfigurationen限制和约束

Einschränkungen:
$\forall\ q \in Q\ :\ \delta(q, \triangleright) = (q', \triangleright, \rightarrow)$ oder $(q', \triangleright, \bot)$ für ein $q' \in Q$.

1. M läuft niemals links über das initiale Symbol $\triangleright$ hinweg.
2. Das initiale Symbol wird niemals überschrieben.

Definition

Eine Konfiguration einer Turingmaschine $M = (Q, \Sigma, \delta, s)$ ist ein Tripel (q, u, w) mit:

- $q \in Q \cup \{h, h_y, h_n\}$ ist der aktuelle Zustand,
- $u \in \{\triangleright\} \times \Sigma^*$ ist das Wort links vom Kopf einschließlich dem Symbol, auf dem der Kopf zeigt, und
- $w \in \Sigma^*$ ist das Wort rechts vom Kopf.

Konfigurationen

Definition

Die Initiale Konfiguration für eine Turingmaschine $M = (Q, \Sigma, \delta, s)$ mit Eingabe $x \in (\Sigma - \{\triangleright, \#\})^*$ ist $\beta_0 = (s, \triangleright, x)$.

Eine Konfiguration $\beta = (q, u, w)$ mit $q \in \{h, h_y, h_n\}$, heißt Haltekonfiguration.

Konfigurationen

Definition

Sei $\beta = (q, u, w)$ mit $q \in Q$, $u = \hat{u}a$, $\hat{u} \in \Sigma^*$ und $a \in \Sigma$ eine Konfiguration. Konfiguration $\beta' = (q', u', w')$ ist die Nachfolgekonfiguration von β bezeichnet mit

$$(q, u, w) \overset{M}{\longmapsto} (q', u', w'),$$

falls $\delta(q, a) = (q', b, D)$ für ein $b \in \Sigma$ und ein $D \in \{\rightarrow, \leftarrow, \perp\}$, so dass folgende Bedingungen erfüllt sind:

- Falls $D = \perp$, dann ist $u' = \hat{u}b$ und $w' = w$,
- falls $D = \leftarrow$, dann ist $u' = \hat{u}$ und $w' = bw$,
- falls $D = \rightarrow$ und $w = \epsilon$, dann ist $u' = \hat{u}b\#$ und $w' = \epsilon$,
- falls $D = \rightarrow$ und $w = c\hat{w}$ für ein $c \in \Sigma$ und $\hat{w} \in \Sigma^*$, dann ist $u' = \hat{u}bc$ und $w' = \hat{w}$.

Berechnungen

Definition

$\overset{M^k}{\longmapsto}$ bezeichnet eine Berechnung in k Schritten.

$\overset{M^*}{\longmapsto}$ bezeichnet eine Berechnung in endlich vielen Schritten (= reflexive und transitive Hülle von $\overset{M}{\longmapsto}$).

Wir sagen, M hält auf Eingabe x, falls es eine Berechnung $(s, \triangleright, x) \overset{M^*}{\longmapsto} (q, u, w)$ gibt mit $q \in \{h, h_y, h_n\}$.

Ausgabe einer Turingmaschine

Definition

Die Ausgabe einer Turingmaschine, bezeichnet mit $M(x)$, ist wie folgt definiert.

- Falls M auf Eingabe x im Zustand h hält, es also eine Berechnung $(s, \triangleright, x) \overset{M^*}{\longmapsto} (h, u, w)$ gibt, ist $M(x)$ das Wort uw ohne das initiale Symbol ganz links und ohne die rechten Leerzeichen.
- Falls M auf Eingabe x im Zustand h_y oder h_n hält, ist $M(x) = \text{yes}$ bzw. $M(x) = \text{no}$.
- Falls M auf Eingabe x nicht hält, es also keine Berechnung $(s, \triangleright, x) \overset{M^*}{\longmapsto} (q, u, w)$ mit $q \in \{h, h_y, h_n\}$ gibt, ist $M(x) = \nearrow$.

Turingmaschinenberechnung

Beispiel

Sei $M = (Q, \Sigma, \delta, s)$ mit $Q = \{s, q_0, q_1, q\}$, $\Sigma = \{0, 1, \triangleright, \#\}$, und

δ	0	1	#	$\triangleright$
s	$(s, 0, \rightarrow)$	$(s, 1, \rightarrow)$	$(q, \#, \leftarrow)$	$(s, \triangleright, \rightarrow)$
q_0	$(s, 0, \leftarrow)$	$(s, 0, \leftarrow)$	$(s, 0, \leftarrow)$	$(h, \triangleright, \perp)$
q_1	$(s, 1, \leftarrow)$	$(s, 1, \leftarrow)$	$(s, 1, \leftarrow)$	$(h, \triangleright, \perp)$
q	$(q_0, \#, \rightarrow)$	$(q_1, \#, \rightarrow)$	$(q, \#, \perp)$	$(h, \triangleright, \perp)$

Berechnung: $M(010) = \#010$

$(s,$	$\triangleright, 010)$	$(q_0,$	$\triangleright 01\#\#, \epsilon)$	$(q_0,$	$\triangleright\#\#, 10)$
$(s,$	$\triangleright 0, 10)$	$(s,$	$\triangleright 01\#, 0)$	$(s,$	$\triangleright\#, 010)$
$(s,$	$\triangleright 01, 0)$	$(q,$	$\triangleright 01, \#0)$	$(q,$	$\triangleright, \#010)$
$(s,$	$\triangleright 010, \epsilon)$	$(q_1,$	$\triangleright 0\#\#, 0)$	$(h,$	$\triangleright, \#010)$
$(s,$	$\triangleright 010\#, \epsilon)$	$(s,$	$\triangleright 0\#, 10)$		
$(q,$	$\triangleright 010, \#)$	$(q,$	$\triangleright 0, \#10)$		

Turingmaschinenberechnung

Beispiel (Alternative Darstellung)

Sei $M = (Q, \Sigma, \delta, s)$ mit $Q = \{s, q_0, q_1, q\}$, $\Sigma = \{0, 1, \triangleright, \#\}$, und

δ	0	1	#	$\triangleright$
s	$(s, 0, \rightarrow)$	$(s, 1, \rightarrow)$	$(q, \#, \leftarrow)$	$(s, \triangleright, \rightarrow)$
q_0	$(s, 0, \leftarrow)$	$(s, 0, \leftarrow)$	$(s, 0, \leftarrow)$	$(h, \triangleright, \perp)$
q_1	$(s, 1, \leftarrow)$	$(s, 1, \leftarrow)$	$(s, 1, \leftarrow)$	$(h, \triangleright, \perp)$
q	$(q_0, \#, \rightarrow)$	$(q_1, \#, \rightarrow)$	$(q, \#, \perp)$	$(h, \triangleright, \perp)$

Berechnung: $M(010) = \#010$

$s,$	$\underline{\triangleright}$	0	1	0	
$s,$	$\triangleright$	$\underline{0}$	1	0	
$s,$	$\triangleright$	0	$\underline{1}$	0	
$s,$	$\triangleright$	0	1	$\underline{0}$	
$s,$	$\triangleright$	0	1	0	$\underline{\#}$
$q,$	$\triangleright$	0	1	$\underline{0}$	$\#$
$q_0,$	$\triangleright$	0	1	$\#$	$\underline{\#}$
$s,$	$\triangleright$	0	1	$\underline{\#}$	0

$q,$	$\triangleright$	0	$\underline{1}$	$\#$	0
$q_1,$	$\triangleright$	0	$\#$	$\underline{\#}$	0
$s,$	$\triangleright$	0	$\underline{\#}$	1	0
$q,$	$\triangleright$	$\underline{0}$	$\#$	1	0
$q_0,$	$\triangleright$	$\#$	$\underline{\#}$	1	0
$s,$	$\triangleright$	$\underline{\#}$	0	1	0
$q,$	$\underline{\triangleright}$	$\#$	0	1	0
$h,$	$\underline{\triangleright}$	$\#$	0	1	0

Entscheiden, Akzeptieren

Definition

Sei $L \subseteq (\Sigma - \{\#, \triangleright\})^*$ eine Sprache und $M = (Q, \Sigma, \delta, s)$ eine Turingmaschine.

- M entscheidet L, falls $\forall\, x \in (\Sigma - \{\#, \triangleright\})^*$:
  $x \in L \;\Rightarrow\; M(x) =$ yes und $x \notin L \;\Rightarrow\; M(x) =$ no.
- M akzeptiert L, falls $\forall\, x \in (\Sigma - \{\#, \triangleright\})^*$:
  $x \in L \;\Rightarrow\; M(x) \neq \nearrow$ und $x \notin L \;\Rightarrow\; M(x) = \nearrow$.
- L heißt rekursiv, wenn es eine Turingmaschine M gibt, die L entscheidet.
- L heißt rekursiv aufzählbar, wenn es eine Turingmaschine M gibt, die L akzeptiert.

rekursiv, rekursiv aufzählbar

Satz

Jede rekursive Sprache ist rekursiv aufzählbar.

(Die Umkehrung gilt nicht!)

Beweis.

Sei $M = (Q, \Sigma, \delta, s)$ eine Turingmaschine, die L entscheidet.

Ändere alle Programmschritte $\delta(q, a) = (h_n, b, D)$ für
$q \in Q, a \in \Sigma, b \in \Sigma$ und $D \in \{\leftarrow, \rightarrow, \perp\}$
in $\delta(q, a) = (q, a, \perp)$.

Wurde vorher der Zustand h_n erreicht, so läuft M nun in eine „Endlosschleife".

(Partiell) rekursive Funktionen

Definition

Sei $f : (\Sigma - \{\#, \triangleright\})^* \longrightarrow \Sigma^*$ eine (totale) Funktion und M eine Turingmaschine mit Alphabet Σ.

- M berechnet f, falls $\forall\, x \in (\Sigma - \{\#, \triangleright\})^* \;:\; M(x) = f(x)$.
- f ist eine rekursive Funktion, falls es eine Maschine M mit $M(x) = f(x)$ gibt.

Sei $f : (\Sigma - \{\#, \triangleright\})^* \longrightarrow \Sigma^*$ eine partielle Funktion und M eine Turingmaschine mit Alphabet Σ.

- M berechnet f, falls $\forall\, x \in (\Sigma - \{\#, \triangleright\})^*$ mit definiertem $f(x) : M(x) = f(x)$.
- f ist eine partiell rekursive Funktion, falls es eine Maschine M gibt, die f berechnet.

Beispiel

$f : \{0,1\}^* \longrightarrow \{0,1,\#\}^*$ mit $f(x) = \#x$ ist eine rekursive Funktion, siehe früheres Beispiel.

Eine Turingmaschine für Palindrome

Beispiel

Sei $M = (Q, \Sigma, \delta, s)$ mit
$Q = \{s, q_0, q_1, q, q_0^1, q_1^1\}$, $\Sigma = \{0, 1, \triangleright, \#\}$ und

δ	0	1	$\triangleright$	$\#$
s	$(q_0, \triangleright, \rightarrow)$	$(q_1, \triangleright, \rightarrow)$	$(s, \triangleright, \rightarrow)$	$(h_y, \#, \perp)$
q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$-$	$(q_0^1, \#, \leftarrow)$
q_1	$(q_1, 0, \rightarrow)$	$(q_1, 1, \rightarrow)$	$-$	$(q_1^1, \#, \leftarrow)$
q_0^1	$(q, \#, \leftarrow)$	$(h_n, 1, \perp)$	$(h_y, \triangleright\perp)$	$-$
q_1^1	$(h_n, 1, \perp)$	$(q, \#, \leftarrow)$	$(h_y, \triangleright, \perp)$	$-$
q	$(q, 0, \leftarrow)$	$(q, 1, \leftarrow)$	$(s, \triangleright, \rightarrow)$	$-$

Die '$-$' Einträge können beliebig gewählt werden.

M entscheidet die Menge aller Palindrome über $\{0, 1\}$.

Begriffstrennung

Trennen Sie zwischen der intuitiven umgangssprachlichen Beschreibung und der formalen Definition.

Begriffstrennung

Trennen Sie zwischen der intuitiven umgangssprachlichen Beschreibung und der formalen Definition.

Rechts neben dem Kopf
stehen nur Leerzeichen.

Begriffstrennung

Trennen Sie zwischen der intuitiven umgangssprachlichen Beschreibung und der formalen Definition.

Rechts neben dem Kopf stehen nur Leerzeichen.	In Konfiguration (q, u, v) ist $v \in \{\#\}^*$.

Begriffstrennung

Trennen Sie zwischen der intuitiven umgangssprachlichen Beschreibung und der formalen Definition.

Rechts neben dem Kopf stehen nur Leerzeichen.

In Konfiguration (q, u, v) ist $v \in \{\#\}^*$.

Bei Eingabe x läuft M in eine Endlosschleife .

Begriffstrennung

Trennen Sie zwischen der intuitiven umgangssprachlichen Beschreibung und der formalen Definition.

Rechts neben dem Kopf stehen nur Leerzeichen.	In Konfiguration (q, u, v) ist $v \in \{\#\}^*$.
Bei Eingabe x läuft M in eine Endlosschleife .	$M(x) = \nearrow$.

Begriffstrennung

Trennen Sie zwischen der intuitiven umgangssprachlichen Beschreibung und der formalen Definition.

Rechts neben dem Kopf stehen nur Leerzeichen.	In Konfiguration (q, u, v) ist $v \in \{\#\}^*$.
Bei Eingabe x läuft M in eine Endlosschleife .	$M(x) = \nearrow$.
Was ist eigentlich eine Turingband?	