Theoretische Informatik 2

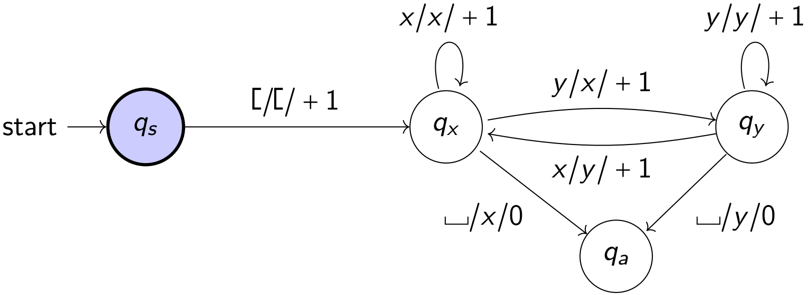
SoSe 2024

Berechenbarkeit und Komplexität

Prof. Dr. Sebastian Siebertz
AG Theoretische Informatik
MZH, Raum 3160
siebertz@uni-bremen.de

Universität Bremen

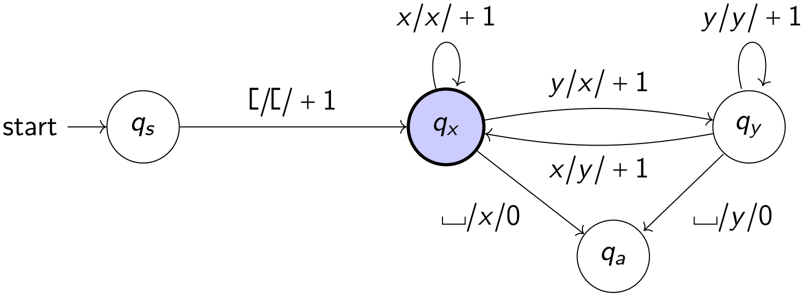
Turingmaschinen Wiederholung



[	y	x	y	x	y	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

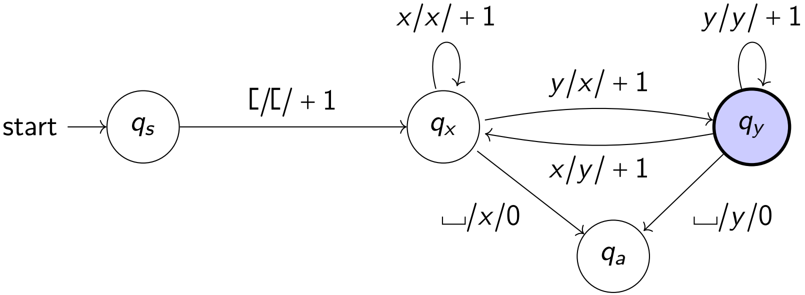
Turingmaschinen Wiederholung



[	y	x	y	x	y	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	…

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

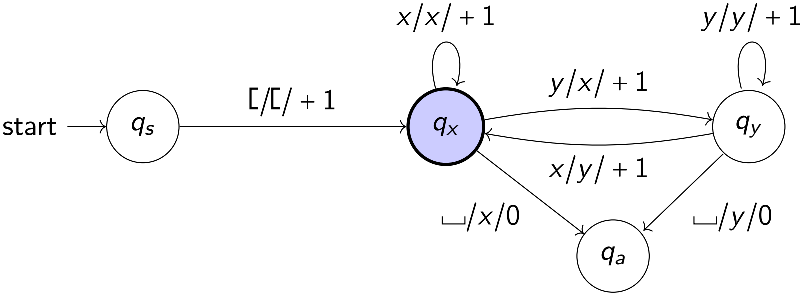
Turingmaschinen Wiederholung



[	x	$\boxed{x}$	y	x	y	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

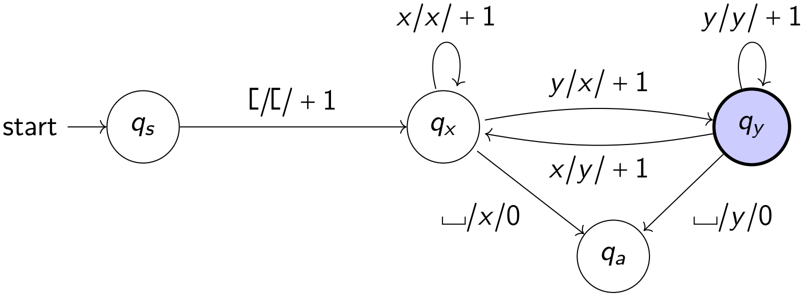
Turingmaschinen Wiederholung



[	x	y	y	x	y	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

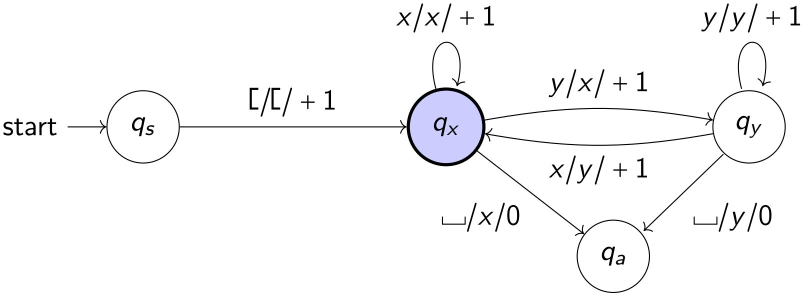
Turingmaschinen Wiederholung



$[$	x	y	x	$\boxed{x}$	y	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\dots$

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

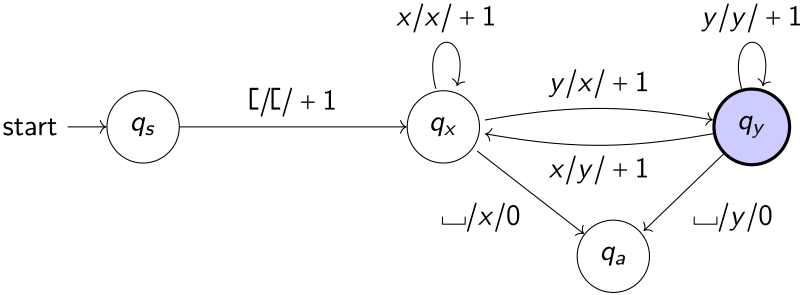
Turingmaschinen Wiederholung

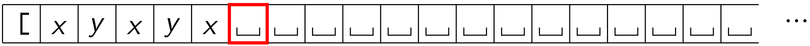


[	x	y	x	y	$\boxed{y}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

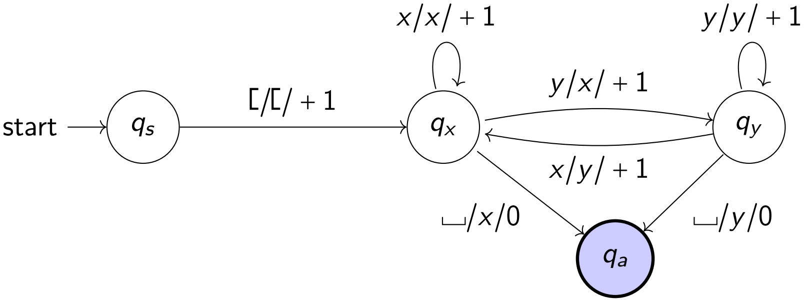
Turingmaschinen Wiederholung





- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

Turingmaschinen Wiederholung



[	x	y	x	y	x	y	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

- Die Maschine soll ein x an die erste Position schreiben und das Eingabewort dafür eine Position nach rechts schieben.

Turingmaschinen Wiederholung

- Eine deterministische Einbandturingmaschine ist ein Tupel $M = (Q, \Sigma, \Gamma, [, \sqcup, \delta, q_s, q_a, q_r)$, wobei
  - Q endliche Zustandsmenge ist,
  - Σ das Eingabealphabet ist,
  - Γ das Arbeitsalphabet ist mit $\Sigma \subseteq \Gamma$,
  - $[\in \Gamma \setminus \Sigma$ der linke Endmarker ist,
  - $\sqcup \in \Gamma \setminus \Sigma$ das Blanksymbol ist,
  - $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\}$ die Übergangsfunktion ist,wobei gilt, dass
    1. für alle $p \in Q$ ein $q \in Q$ und $d \in \{0, +1\}$ existieren, so dass $\delta(p, [) = (q, [, d)$ und
    2. für alle $a \in \Gamma$ gilt $\delta(q_a, a) = (q_a, a, 0)$ und $\delta(q_r, a) = (q_r, a, 0)$,
  - $q_s \in Q$ der Anfangszustand ist,
  - $q_a \in Q$ der akzeptierende Zustand ist und
  - $q_r \in Q$ der verwerfende Zustand ist, $q_a \neq q_r$.

Turingmaschinen Wiederholung

- $\delta(p,a) = (q,b,d)$ für $p,q \in Q, a,b \in \Gamma$ und $d \in \{-1,0,+1\}$ bedeutet, dass die Maschine,
  - wenn sie im Zustand p
  - an der aktuellen Kopfposition k
  - das Symbol a liest,
  - so wechselt sie in den Zustand q,
  - schreibt das Symbol b an der Kopfposition auf das Band und
  - bewegt den Kopf auf Position $k + d$.
    - $d = -1 \to$ nach links
    - $d = 0 \to$ stehenbleiben
    - $d = +1 \to$ nach rechts

Äquivalente Varianten der Turingmaschine

- Es gibt zahlreiche unterschiedliche Definitionen und Varianten von Turingmaschinen.
- Das Modell der Turingmaschinen ist robust unter kleinen Veränderungen.
  - Es ergibt sich immer die gleiche Klasse von erkennbaren und entscheidbaren Sprachen.

Äquivalente Varianten der Turingmaschine

- Häufig betrachtete Varianten:
  - Statt q_a und q_r: Menge von akzeptierenden Zuständen F.
  - Lese- und Schreibkopf muss sich in jedem Berechnungsschritt bewegen.
  - Speicherband in beide Richtungen unendlich.
  - Mehrere Spuren zur Berechnung: Mehrspur-Turingmaschinen.
  - Mehrere Speicherbänder mit unabhängigen Lese- und Schreibköpfen: Mehrband-Turingmaschinen.
  - Nichtdeterministische Maschinen.

    $\rightarrow$ Extrem wichtig in der Komplexitätstheorie.

Mehrband-Turingmaschinen

- Mehrband-Turingmaschinen: mehrere Bänder mit mehreren unabhängigen Lese- und Schreibköpfen.

…	␣	␣	␣	1	0	1	#	0	0	0	#	1	0	#	1	1	#	0	0	#	0	$\boxed{0}$	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	0	0	#	0	0	#	0	0	#	1	1	#	1	0	0	#	0	$\boxed{␣}$	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	␣	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

- Einseitig oder beidseitig unbeschränkte Bänder.

Mehrband-Turingmaschinen

- Deterministisch: Übergangsfunktion

  $$\delta: \quad Q \times \Gamma^k \to Q \times \Gamma^k \times \{-1, 0, +1\}^k$$

  - $\delta(p, a_1, \ldots, a_k) = (q, b_1, \ldots, b_k, d_1, \ldots d_k)$:
    - Wenn M in Zustand p
    - auf den k Bändern an der jeweiligen Kopfposition die Zeichen $a_1, \ldots, a_k$ liest,
    - so wechselt sie in Zustand q,
    - schreibt die Zeichen $b_1, \ldots, b_k$ auf die Bänder
    - und bewegt sich auf Band i in Richtung d_i.

Nichtdeterministische Mehrband-Turingmaschinen

- Nichtdeterministisch: in einem Zustand
  kann es verschiedene Möglichkeiten geben die Berechnung fortzusetzen!

- Übergangsrelation

  $\Delta \subseteq Q \times \Gamma^k \ \times \ Q \times \Gamma^k \times \{-1, 0, +1\}^k$

  - $(p, a_1, \ldots, a_k, q, b_1, \ldots, b_k, d_1, \ldots, d_k) \in \Delta$:
    - Wenn M in Zustand p
    - auf den k Bändern an der jeweiligen Kopfposition die Zeichen $a_1, \ldots, a_k$ liest,
    - so kann sie in Zustand q wechseln,
    - die Zeichen $b_1, \ldots, b_k$ auf die Bänder schreiben
    - und sich auf Band i in Richtung d_i bewegen.
  - Es kann weitere Tupel
    $(p, a_1, \ldots, a_k, q', b'_1, \ldots, b'_k, d'_1, d'_k) \in \Delta$ geben.

Beispiel: Partitionsproblem

- Eingabe: Multimenge M von natürlichen Zahlen (Zahlen dürfen mehrfach vorkommen).
- Frage: kann M in zwei Teilmengen M_1, M_2 partitioniert werden, so dass
$$\sum_{m \in M_1} m = \sum_{m \in M_2} m?$$

- Kodiere Multimenge $M = \{m_1, \ldots, m_k\}$ als
$$\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*.$$
- Entscheidungsproblem:
$$\mathcal{P} = \{\mathrm{code}(M) \;:\; M \subseteq \mathbb{N} \text{ positive Instanz des Partitionsproblems}\}.$$

Beispiel: Partitionsproblem

- Beispielinstanz: $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Positiv, denn $\{5, 4, 2, 3, 3, 4, 1\} = \{5, 3, 2, 1\} \cup \{4, 4, 3\}$ und $\sum_{m \in \{5,3,2,1\}} m = \sum_{m \in \{4,4,3\}} m = 11$
- Beispielinstanz: $M' = \{5, 4, 2, 3, 3, 4\}$.
  - Negativ, denn $\sum_{m \in \{5,4,2,3,3,4\}} m$ ist ungerade

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

...	␣	␣	␣	$\boxed{1}$	0	1	#	1	0	0	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	...
...	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	...
...	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	...
...	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	...

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	$\boxed{0}$	1	#	1	0	0	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	**1**	#	1	0	0	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	**␣**	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	**␣**	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	**␣**	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	$\boxed{\#}$	1	0	0	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	$\boxed{1}$	0	0	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
    - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
    - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
    - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	$\boxed{0}$	0	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\dots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	0	$\boxed{0}$	#	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	0	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	0	0	$\boxed{\#}$	1	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	0	0	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\# \dots \#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

...	␣	␣	␣	1	0	1	#	0	0	0	#	$\boxed{1}$	0	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	...
...	␣	␣	␣	0	0	0	#	1	0	0	#	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	...
...	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	...
...	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	...

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	0	0	#	1	$\boxed{0}$	#	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	0	0	#	0	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	0	0	#	1	0	**[#]**	1	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	0	0	#	0	0	**[␣]**	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	**[␣]**	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	**[␣]**	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	0	0	#	1	0	#	$\boxed{1}$	1	#	1	1	#	1	0	0	#	1	␣	␣	…
…	␣	␣	␣	0	0	0	#	1	0	0	#	0	0	#	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Berechnung einer nichtdeterministischen 4-Band Turingmaschine:
  - $M = \{5, 4, 2, 3, 3, 4, 1\}$.
  - Eingabe: $\mathrm{code}(M) = \mathrm{bin}(m_1)\#\mathrm{bin}(m_2)\#\ldots\#\mathrm{bin}(m_k) \in \Sigma^*$.
  - Phase 1: rate M_1 und M_2

…	␣	␣	␣	1	0	1	#	0	0	0	#	1	0	#	1	1	#	0	0	#	0	0	0	#	1	$\boxed{␣}$	␣	…
…	␣	␣	␣	0	0	0	#	1	0	0	#	0	0	#	0	0	#	1	1	#	1	0	0	#	0	$\boxed{␣}$	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Beispiel: Partitionsproblem

- Phase 2: Laufe zurück und addiere die Zahlen auf Band 1 auf Band 3 und die Zahlen auf Band 2 auf Band 4.

…	␣	␣	$\boxed{␣}$	1	0	1	#	0	0	0	#	1	0	#	1	1	#	0	0	#	0	0	0	#	1	␣	␣	…
…	␣	␣	$\boxed{␣}$	0	0	0	#	1	0	0	#	0	0	#	0	0	#	1	1	#	1	0	0	#	0	␣	␣	…
…	␣	␣	␣	1	1	0	1	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…
…	␣	␣	␣	1	1	0	1	$\boxed{␣}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

- Phase 3: Überprüfe ob die beiden Zahlen auf Band 3 und Band 4 gleich sind.

Die Maschine für Phase 1

$$
\begin{tikzcd}[row sep=huge]
\text{start} \arrow[r] & q_1 \arrow[loop above, "{\begin{bmatrix} 1 & / & 1 & / & +1 \\ \sqcup & / & 0 & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}, \begin{bmatrix} 0 & / & 0 & / & +1 \\ \sqcup & / & 0 & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}, \begin{bmatrix} \# & / & \# & / & +1 \\ \sqcup & / & \# & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}}"] \arrow[d, leftrightarrow, "{\begin{bmatrix} \# & / & \# & / & +1 \\ \sqcup & / & \# & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}}"] \\
\text{start} \arrow[r] & q_2 \arrow[loop below, "{\begin{bmatrix} 1 & / & 0 & / & +1 \\ \sqcup & / & 1 & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}, \begin{bmatrix} 0 & / & 0 & / & +1 \\ \sqcup & / & 0 & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}, \begin{bmatrix} \# & / & \# & / & +1 \\ \sqcup & / & \# & / & +1 \\ \sqcup & / & \sqcup & / & 0 \\ \sqcup & / & \sqcup & / & 0 \end{bmatrix}}"]
\end{tikzcd}
$$

Nichtdeterministische Mehrband-Turingmaschine mit beidseitig unbeschränkten Bändern

- Eine nichtdeterministische k-Band-Turingmaschine mit beidseitig unbeschränkten Bändern ist ein Tupel

$$M = (Q, \Sigma, \Gamma, \sqcup, \Delta, Q_s, F),$$

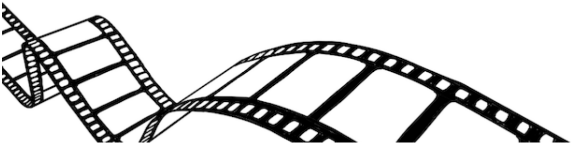
wobei

  - Q eine endliche Zustandsmenge ist,
  - Σ das Eingabealphabet ist, Γ das Arbeitsalphabet ist mit $\Sigma \subseteq \Gamma$ und $\sqcup \in \Gamma \setminus \Sigma$ das Blanksymbol ist,
  - $\Delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{-1, 0, +1\}^k$ die Übergangsrelation ist,
  - $Q_s \subseteq Q$ die Menge der Anfangszustände ist und
  - $F \subseteq Q$ die Menge der akzeptierenden Zustände ist.

Beidseitig unendlich beschränkte Bänder

- Beidseitig unendliches Wort über Alphabet Σ: Abbildung $w : \mathbb{Z} \to \Sigma$.
- Menge aller beidseitig unendlichen Wörter (also aller Abbildungen von $\mathbb{Z}$ nach Σ): $\Sigma^{\mathbb{Z}}$.
- Für ein beidseitig unendliches Wort w über Σ, $z \in \mathbb{Z}$ und $a \in \Sigma$ definiere

$$w[z \mapsto a] : i \mapsto \begin{cases} a & \text{falls } i = z \\ w(i) & \text{sonst.} \end{cases}$$



Konfigurationen

- Sei $M = (Q, \Sigma, \Gamma, \textvisiblespace, \Delta, Q_s, F)$ eine nichtdeterministische k-Band Turingmaschine.
- Konfiguration von M:

$$\alpha = (q, w_1, \dots, w_k, p_1, \dots, p_k) \in Q \times \Gamma^{\mathbb{Z}} \times \dots \times \Gamma^{\mathbb{Z}} \times \mathbb{Z} \times \dots \times \mathbb{Z},$$

so dass für alle $1 \le i \le k$ Positionen $\ell_i < r_i \in Z$ existieren mit $w(j) = \textvisiblespace$ für alle $j \in \mathbb{Z}$ mit $j < \ell_i$ oder $j > r_i$.

…	␣	␣	$\boxed{\textvisiblespace}$	1	0	1	#	0	0	0	#	1	0	#	1	1	#	0	0	#	0	0	0	#	1	␣	␣	…

…	␣	␣	$\boxed{\textvisiblespace}$	0	0	0	#	1	0	0	#	0	0	#	0	0	#	1	1	#	1	0	0	#	0	␣	␣	…

…	␣	␣	␣	1	1	0	1	$\boxed{\textvisiblespace}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

…	␣	␣	␣	1	1	0	1	$\boxed{\textvisiblespace}$	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	␣	…

Konfigurationen

- Konfiguration $(q, w_1, \dots, w_k, p_1, \dots, p_k)$, wobei

$$w_i = \dots \sqcup \sqcup \sqcup \; v_i \sqcup \sqcup \sqcup \dots \text{ für ein } v_i \in \Gamma^*,$$

bedeutet:

  - M befindet sich in Zustand q,
  - das $i$te Band ist mit

$$\dots \sqcup \sqcup \sqcup \; v_i \sqcup \sqcup \sqcup \dots$$

    beschriftet und
  - der $i$te Kopf befindet sich an Position p_i auf dem $i$ten Band.

Übergänge

$$(p, a_1, \ldots, a_k, q, b_1, \ldots, b_k, d_1, \ldots, d_k) \in \Delta$$

- wenn M im Zustand p
- auf dem $i$ten Band an der aktuellen Kopfposition p_i
- das Symbol a_i liest,
- so hat sie die Möglichkeit in Zustand q zu wechseln,
- auf dem $i$ten Band das Symbol b_i zu schreiben und
- den Kopf auf dem $i$ten Band auf Position $p_i + d_i$ zu bewegen.

Formal: Folgekonfigurationen

- Sei

$$\alpha = (p, w_1, \ldots, w_k, p_1, \ldots, p_k) \in Q \times \Gamma^{\mathbb{Z}} \times \ldots \times \Gamma^{\mathbb{Z}} \times \mathbb{Z} \times \ldots \times \mathbb{Z}$$

  eine Konfiguration mit

$$w_1(p_1) = a_1, \ldots, w_k(p_k) = a_k$$

  und

- $(p, a_1, \ldots, a_k, q, b_1, \ldots, b_k, d_1, \ldots, d_k) \in \Delta$.
- Dann ist

$$\beta = (q, w_1[p_1 \mapsto b_1], \ldots, w_k[p_k \mapsto b_k], p_1 + d_1, \ldots, p_k + d_k)$$

  eine Folgekonfiguration von α.

- Wir schreiben $\alpha \vdash_M \beta$.

Start- und Stoppkonfigurationen

- Eine Konfiguration $\alpha_1 = (q_s, w_1, \ldots, w_k, p_1, \ldots, p_k)$ heißt Startkonfiguration von M auf $w \in \Sigma^*$ falls
  - $q_s \in Q_s$,
  - $w_1(i) = w(i)$ für alle $i \in \mathbb{Z}$ mit $1 \leq i \leq |w|$ und $w_1(i) = \sqcup$ für alle $i \in \mathbb{Z}$ mit $i \leq 0$ oder $i \geq |w| + 1$,
  - $w_j(i) = \sqcup$ für alle $2 \leq j \leq k$ und alle $i \in \mathbb{Z}$, und
  - $p_i = 1$ für alle $1 \leq i \leq k$.
- Eine Konfiguration α ist eine Stoppkonfiguration, falls α keine Folgekonfiguration hat.
- Eine Stoppkonfiguration $(q, w_1, \ldots, w_k, p_1, \ldots, p_k)$ heißt akzeptierend, falls $q \in F$ und verwerfend falls $q \notin F$.

Berechnungen

- Berechnung von M auf w: maximale (endliche oder unendliche) Konfigurationsfolge

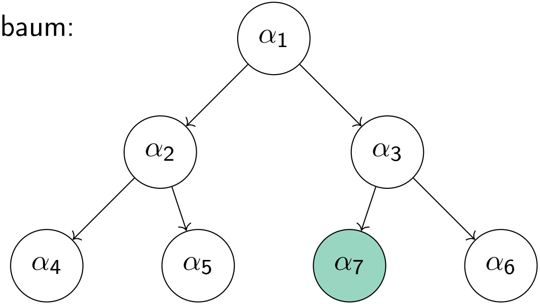
$$\alpha_1 \vdash_M \alpha_2 \vdash_M \dots,$$

  wobei α_1 eine Startkonfiguration von M auf w ist.

- Berechnung ist terminierend, wenn sie in endlich ist.
- Terminierende Berechnung $\alpha_1 \vdash_M \alpha_2 \vdash_M \dots \vdash_M \alpha_m$ ist akzeptierend, falls α_m akzeptierend ist und verwerfend, falls α_m verwerfend ist.
- Die Maschine M akzeptiert w, wenn es eine akzeptierende Berechnung von M auf w gibt.

Berechnungsbaum

- Berechnungsbaum:



  - Wurzel: Startkonfiguration
  - Jeder Knoten hat Folgekonfigurationen als Kinder
  - *M* akzeptiert *w*, wenn mindestens eine Konfiguration im Berechnungsbaum akzeptierend ist.

NTM-erkennbare Sprachen

- Die von einer nichtdeterministischen Turingmaschine M erkannte Sprache ist die Menge $L(M) = \{w \in \Sigma^* \ : \ M \text{ akzeptiert } w\}$.
- Eine Sprache $L \subseteq \Sigma^*$ heißt NTM-erkennbar, wenn es eine nichtdeterministische Turingmaschine M gibt mit $L = L(M)$.
- Eine Sprache $L \subseteq \Sigma^*$ heißt NTM-entscheidbar, wenn es eine nichtdeterministische Turingmaschine M gibt, so für jedes Wort $w \in \Sigma^*$ jede Berechnung von M auf w terminiert und so dass M das Wort w akzeptiert genau dann, wenn $w \in L$.

Ein Startzustand ist ausreichend

Lemma

Sei $M = (Q, \Sigma, \Gamma, \sqcup, \Delta, Q_s, F)$ *eine nichtdeterministische* k*-Band Turingmaschine. Dann existiert eine nichtdeterministische* k*-Band Turingmaschine* $M' = (Q', \Sigma, \Gamma, \sqcup, \Delta', Q'_s, F')$ *mit* $|Q'_s| = 1$*, so dass*

- $L(M) = L(M')$ *und*
- *falls für* $w \in \Sigma^*$ *alle Berechnungen von* M *auf* w *endlich sind, so sind auch alle Berechnungen von* M' *auf* w *endlich.*

Ein Startzustand ist ausreichend

Beweis.

- Füge neuen Startzustand ein: $Q' = Q \cup \{q'_s\}$ (Annahme $q'_s \notin Q$) und setze $Q'_s = \{q'_s\}$.
- Füge "ε-Transition" von q'_s zu den alten Startzuständen ein:

$$\Delta' := \Delta \cup \{(q'_s, a_1, \dots, a_k, q_s, a_1, \dots, a_k, 0, \dots, 0) \ : \\ a_1, \dots, a_k \in \Gamma, q_s \in Q_s\}.$$

- Jede Berechnung von M kann zu Berechnung von M' erweitert werden.
- Umgekehrt wird jede Berechnung von M' zu einer Berechnung von M durch Weglassen der ersten Konfiguration.

Ein Band ist ausreichend

Lemma

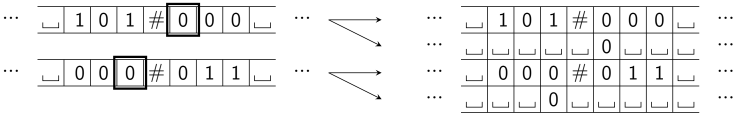
Sei $M = (Q, \Sigma, \Gamma, \sqcup, \Delta, Q_s, F)$ *eine nichtdeterministische* k*-Band Turingmaschine. Dann existiert eine nichtdeterministische* 1*-Band Turingmaschine* $M' = (Q', \Sigma, \Gamma', \sqcup, \Delta', Q'_s, F')$*, so dass*

- $L(M) = L(M')$ *und*
- *falls für* $w \in \Sigma^*$ *alle Berechnungen von* M *auf* w *endlich sind, so sind auch alle Berechnungen von* M' *auf* w *endlich.*

Ein Band ist ausreichend

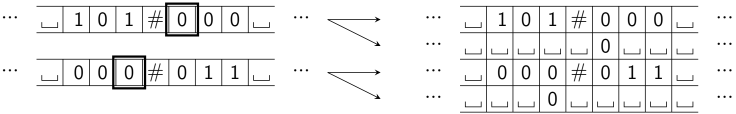
Beweis.

- Wir erweitern das Bandalphabet um die Beschriftungen der k Bänder in einem einzigen Zeichen zu simulieren.



- $\Gamma' = \Gamma^{2k}$.
- Benutzten ein $x \in \Sigma$ als Markersymbol für die Kopfpositionen.
- Zeichen sind Tupel aus Γ mit $2k$ Einträgen.
  - Eintrag $2i - 1$ des Tupels kodiert Beschriftung von Band i
  - Eintrag $2i$ kodiert Kopfposition von Band i.

Ein Band ist ausreichend



- Anschaulich hat M' auf einem Band $2k$ Spuren.
  - Der $i$te Eintrag eines Tupels aus Γ' wird $i$te Spur genannt.
  - M' wird Mehrspur-Maschine genannt.
- M' simuliert nacheinander die Berechnung von M auf jedem Band.
  - Für $i = 1, \dots, k$ sucht sie zunächst den Marker x auf der $2i$-ten Spur.
  - Sie liest das Zeichen auf der Spur $2i-1$ darüber und verändert es so wie M es auf Band i verändern würde.
  - Sie verschiebt den Marker auf Spur $2i$ so wie M den Kopf auf Band i verändern würde.

Ein Band ist ausreichend

- Die Details sind technisch.
  - Welche Zustände benötigt M'?
  - Wie sucht M' den Marker auf einer Spur?
  - Wie werden die Spuren im Detail verändert?
  - Würde auch $\Gamma' = \{0, 1, \sqcup\}$ ausreichen?

Einseitig unendliches Arbeitsband ist ausreichend

- Ähnliche können wir ein beidseitig unendliches Band durch ein einseitig beschränktes Band simulieren.
- Benutzten 4 Spuren
  - Spur 1: Bandinhalt von Positionen $0, 1, \ldots$
  - Spur 2: Kopfposition falls sich Kopf auf Position $0, 1, \ldots$ befindet.
  - Spur 3: Bandinhalt von Positionen $-1, -2, \ldots$ gespiegelt!
    - Eintrag von Position $-i$ an Position i.
  - Spur 4: Kopfposition falls sich Kopf auf Position $-1, -2, \ldots$ befindet.

Deterministische Maschinen sind ausreichend

Lemma

Sei M eine nichtdeterministische Turingmaschine. Dann existiert eine deterministische Turingmaschine M', so dass

- $L(M) = L(M')$ *und*
- *falls für $w \in \Sigma^*$ alle Berechnungen von M auf w endlich sind, so ist auch die Berechnung von M' auf w endlich.*

- Fundamental wichtiges Ergebnis!

Konfigurationen kodieren

- Konfigurationen sind unendliche Objekte

$$\alpha = (q, w_1, \ldots, w_k, p_1, \ldots, p_k) \in Q \times \Gamma^{\mathbb{Z}} \times \ldots \times \Gamma^{\mathbb{Z}} \times \mathbb{Z} \times \ldots \times \mathbb{Z}.$$

- Trotzdem können sie als endliche Wörter kodiert werden.
    - Für alle $1 \le i \le k$ existieren Positionen $\ell_i < r_i \in Z$ mit $w(j) = \sqcup$ für alle $j \in \mathbb{Z}$ mit $\ell_i \le j \le r_i$.
    - Die unendlich vielen $\sqcup$ Symbole müssen nicht explizit gespeichert werden.

Konfigurationen kodieren

- Wähle $\Gamma' = \Gamma \cup Q \cup \{[,]\}$: Markersymbole [,] und Zustände der Maschine als neue Symbole (endlich viele!)
- Kodiere $\alpha = (q, w_1, \ldots, w_k, p_1, \ldots, p_k)$, wobei

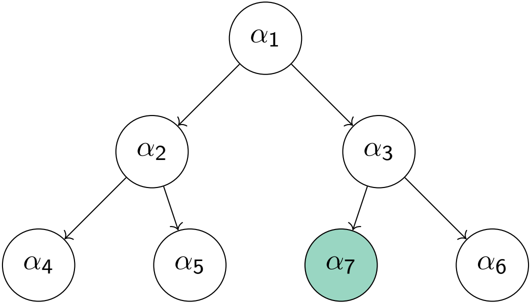
$$w_i = \ldots \sqcup \sqcup \sqcup\, v_i(1) v_i(2) \ldots v_i(\ell_i) \sqcup \sqcup \sqcup \ldots \text{ für ein } v_i \in \Gamma^*,$$

als
$$\begin{aligned} \mathrm{code}(\alpha) = {} & [v_1(1) \ldots\ldots v_1(p_1 - 1) {\color{red}q} v_1(p_k) \ldots v_1(\ell_i)] \\ & [v_2(1) \ldots v_2(p_2 - 1) {\color{red}q} v_2(p_k) \ldots\ldots v_2(\ell_2)] \\ & \ldots \\ & [v_k(1) \ldots\ldots v_k(p_2 - 1) {\color{red}q} v_k(p_k) \ldots v_k(\ell_k)], \end{aligned}$$

d.h., die Beschriftungen der Bänder werden durch Marker getrennt hintereinander geschrieben und der aktuelle Zustand wird vor die jeweilige Kopfposition geschrieben.

Berechnungsbäume revisited

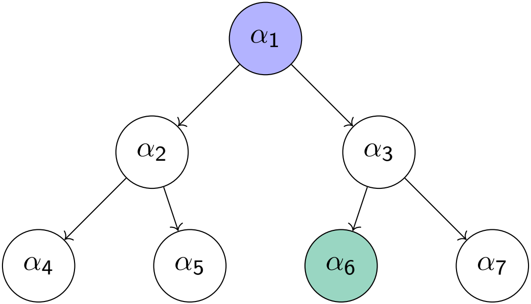
- Berechnungsbaum:
  - Wurzel: Startkonfiguration
  - Jeder Knoten hat Folgekonfigurationen als Kinder
  - *M* akzeptiert *v*, wenn eine Konfiguration im Berechnungsbaum akzeptierend ist.



Berechnungsbäume revisited

- Der Berechnungsbaum kann unendlich tief sein, wenn eine Berechnung nicht terminiert.
- Zu jeder Konfiguration gibt es nur endlich viele mögliche Folgekonfigurationen, abhängig von Q und Γ.
  - Der Berechnungsbaum hat endlichen Verzweigungsgrad.
- Jede Konfiguration wird mit einer Breitensuche gefunden.

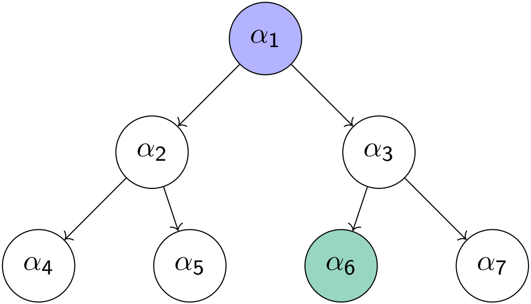
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_1]$$

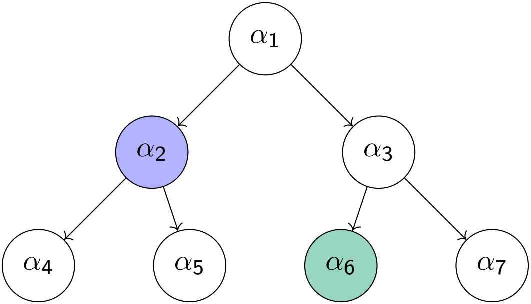
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_1, \alpha_2, \alpha_3]$$

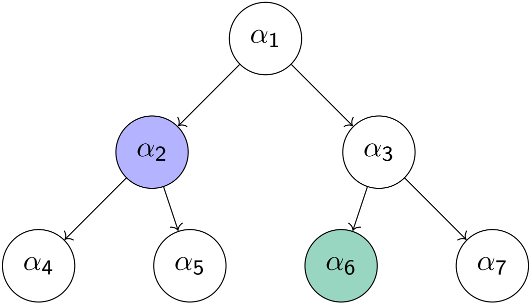
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_2, \alpha_3]$$

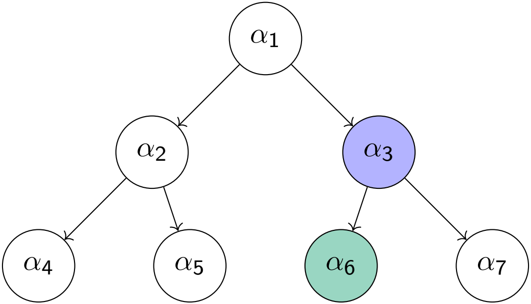
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_2, \alpha_3, \alpha_4, \alpha_5]$$

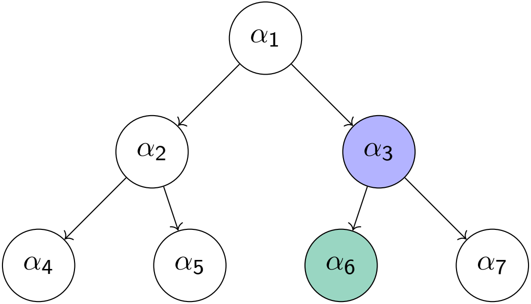
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_3, \alpha_4, \alpha_5]$$

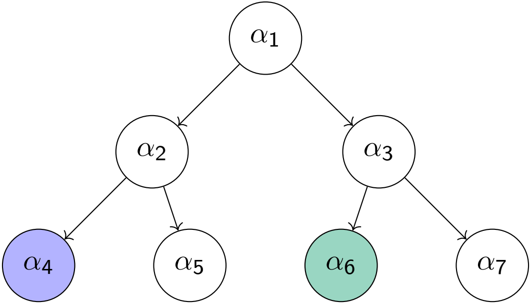
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7]$$

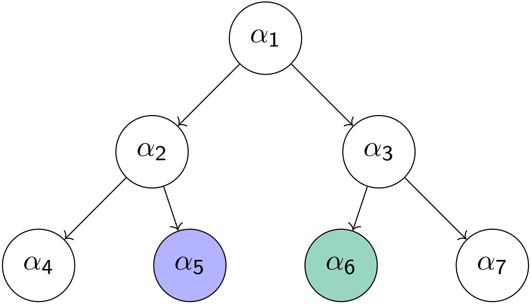
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_4, \alpha_5, \alpha_6, \alpha_7]$$

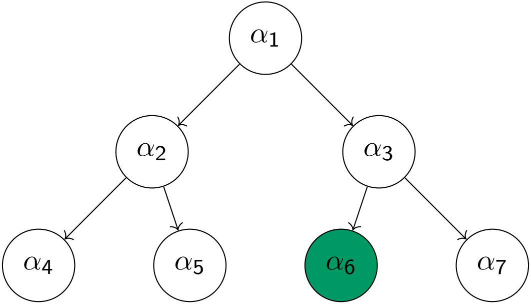
Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_5, \alpha_6, \alpha_7]$$

Breitensuche



- Liste von Einträgen, deren Nachfolger noch untersucht werden müssen:

$$[\alpha_6, \alpha_7]$$

Simulation der Breitensuche auf Berechnungsbaum

- M' simuliert Breitensuche auf Berechnungsbaum auf 4 Bändern.
  - Band 1: Eingabe w.
  - Band 2: Liste der zu besuchenden Knoten für Breitensuche.
    - Initialisierung: Berechne Kodierung $\mathrm{code}(\alpha_0)$ der Startkonfiguration α_0.
    - In jedem Schritt: kopiere die erste Konfiguration α des zweiten Bandes auf das dritte Band und lösche α auf dem zweiten Band.
  - Band 3: Aktuell betrachtete Konfiguration α.
  - Band 4: Nachfolgekonfigurationen β von α
    - Berechne alle Nachfolgekonfigurationen β von α.
    - Kopiere den Inhalt des vierten Bandes an das Ende des zweiten Bandes.
    - Lösche die Inhalte des dritten und vierten Bandes.
    - Gehe zum nächsten Schritt.

Simulation der Breitensuche auf Berechnungsbaum

- Wenn M' akzeptierende Konfiguration berechnet: wechsele in den akzeptierenden Zustand und terminiere.
- Wenn das zweite Band leer ist:
  - Der Berechnungsbaum ist endlich und die Breitensuche terminiert ohne eine akzeptierende Konfiguration zu finden.
  - Wechsele in den verwerfenden Zustand und terminiere.

Simulation der Breitensuche auf Berechnungsbaum

- $w \in L(M)$
  - $\Leftrightarrow$ es existiert akzeptierende Konfiguration in Berechnungsbaum
  - $\Leftrightarrow$ akzeptierende Konfiguration wird durch Breitensuche gefunden
  - $\Leftrightarrow$ M' akzeptiert w

$\Rightarrow$ $L(M) = L(M')$

Simulation der Breitensuche auf Berechnungsbaum

- Für $w \in \Sigma^*$ sind alle Berechnungen von M auf w endlich

$\Leftrightarrow$ der Berechnungsbaum von M auf w ist endlich

$\Rightarrow$ die Berechnung von M' auf w ist endlich.

Deterministische Maschinen sind ausreichend

- Die technischen Details sind wieder aufwändig.
- Die wichtigen Einsichten für die Konstruktion zusammengefasst:
  - Konfigurationen können als endliche Wörter kodiert werden.
  - In einem endlich verzweigten Baum wird durch eine Breitensuche jeder Knoten nach endlicher Zeit besucht, auch wenn der Baum unendlich tief ist.
  - Eine deterministische Turingmaschine kann eine Breitensuche des Berechnungsbaums einer nichtdeterministischen Turingmaschine simulieren.

Deterministische Maschinen sind ausreichend

Satz

Sei $L \subseteq \Sigma^*$ eine Sprache.

- L ist erkennbar $\Leftrightarrow$ L ist NTM-erkennbar.
- L ist entscheidbar $\Leftrightarrow$ L ist NTM-entscheidbar.

- Der Berechenbarkeitsbegriff ist robust unter den diskutierten Änderungen des Maschinenmodells.
- Wähle das Modell, das für eine gegebene Aufgabe am besten geeignet ist.