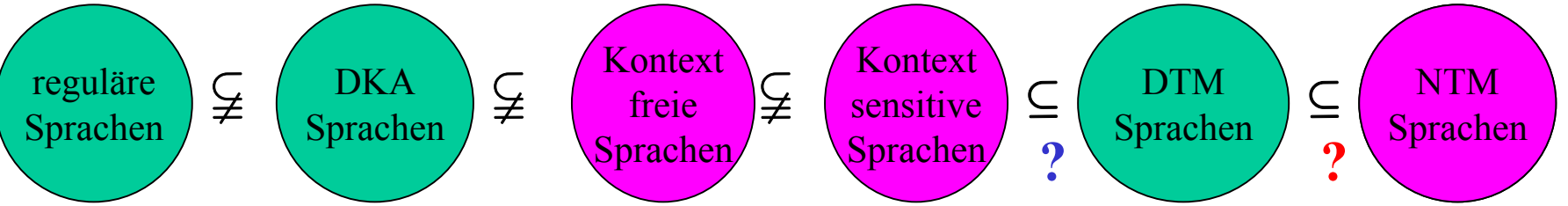


DEA / NEA	DKA	NKA	LBA	DTM	NTM
nkslineare Grammatik rechtslin. Gr.)	LR(k) Gr.	kontext freie Gr.	kontext sensitive Gr.	?	Typ 0 Grammatik

eguläre Ausdrücke
Myhill-Nerode

Satz:
Die DTM-Sprachen sind genau das Gleiche wie die NTM-Sprachen.

Beweis: Es reicht zu zeigen:

Für jede k-Band NTM $M=(\Sigma,\Gamma,\#,Q,s,F,\Delta)$ gibt es eine (k+1)-Band DTM M', sodass $x \in L(M)$ genau dann wenn $x \in L(M')$.

Idee: Simuliere "alle" möglichen Abläufe von M bei Eingabe x.
Für jede einzelnen Ablauf verwende einen „Leitstring", um die nicht-deterministischen Wahlmöglichkeiten deterministisch zu treffen.

Für „Situation" $S = (q,a,A_1,\cdots,A_k) \in Q\times\Sigma\times\Gamma^k$ sei W_S die Anzahl der anwendbaren Regeln in Δ.

$$W = \max\{ W\_S \mid S \in Q\times\Sigma\times\Gamma^k \}$$

Für jede Situation S, gib den in S anwendbaren Regeln in Δ eine feste Ordnung.

Sei $LS = \{1,\cdots,W\}^*$ die Menge der "Leitstrings"

Verwendung von Leitstring $I = (i_1,i_2,\cdots,i_k) \in LS$ bedeutet:

> Simuliere M mit Eingabe x für k Schritte.
> Verwende im j-ten Schritt die i_j-te anwendbare Regel
> (falls so eine Regel nicht existiert, stop)

Die deterministische TM M' verwendet Band k+1 fürs Speichern des aktuellen Leitstrings.

M' generiert einen Leitstring nach dem anderen auf Band k+1.

Für jeden Leitstring I:

M' verwendet Leitstring I für Eingabe x

Wenn dabei M Eingabe x akzeptiert,

dann akzeptiert M' ebenfalls Eingabe x.

Wenn nicht, dann löscht M' die Arbeitsbänder 1 bis k,

und bewegt Eingabelesekopf zum Eingabeanfang.

DEA / NEA	DKA	NKA	LBA	DTM NTM
nkslineare Grammatik echtslin. Gr.)	LR(k) Gr.	kontext freie Gr.	kontext sensitive Gr.	Typ 0 ammatik

eguläre Ausdrücke
Myhill-Nerode

Berechnen von Funktionen durch Turingmaschinen

Erkläre ein Band von det. TM M zum „Ausgabeband“.

Wenn M mit Eingabe x stoppt, dann ist der Inhalt dieses Bandes „die Ausgabe von M bei Eingabe x“.

Eine partielle Funktion $f : \Sigma^* \to \Gamma^*$ heißt (Turing-)berechenbar, wenn es eine det. TM M gibt, sodass für jedes $x \in \Sigma^*$
M bei Eingabe x mit Ausgabe f(x) hält, falls f(x) definiert,
und M bei Eingabe x nicht hält, falls f(x) nicht definiert.

Eine partielle Funktion $f : \mathbb{N}^k \to \mathbb{N}$ heißt (Turing-)berechenbar, falls die Funktion

$$bin(x_1)\$bin(x_2)\$\cdots\$bin(x_k) \mapsto bin(\ f(x_1,\cdots,x_k)\ )$$

(Turing-)berechenbar ist. (bin(x) .. binäre Kodierung von x)

Achtung: f Turing-berechenbar ist etwas anderes als das genaue Verhalten von f zu wissen.

$$f(n) = \begin{cases} 1 & \text{falls in der Dezimaldarstellung von } \pi \text{ irgendwo mindesten } n \text{ Zifferen } 7 \text{ hintereinander vorkommen} \\ 0 & \text{sonst.} \end{cases}$$

Diese Funktion ist auf jeden Fall berechenbar, aber man weiß nicht, was sie ist.

Church-Turing These:

Jede „intuitiv“ berechenbare Funktion ist Turing-berechenbar.

Church-Turing These:

Jede „intuitiv“ berechenbare Funktion ist Turing-berechenbar.

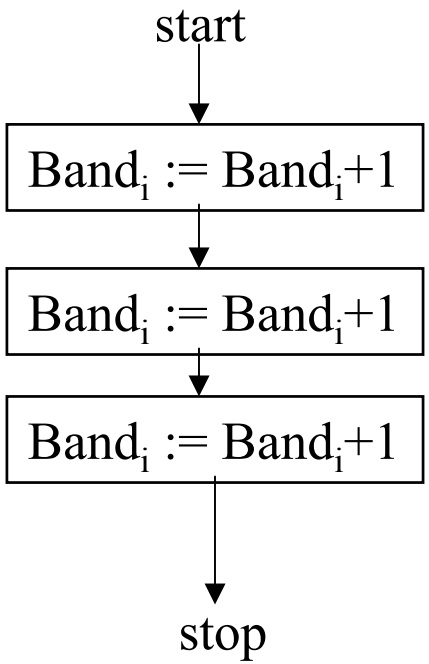
Church-Turing These:

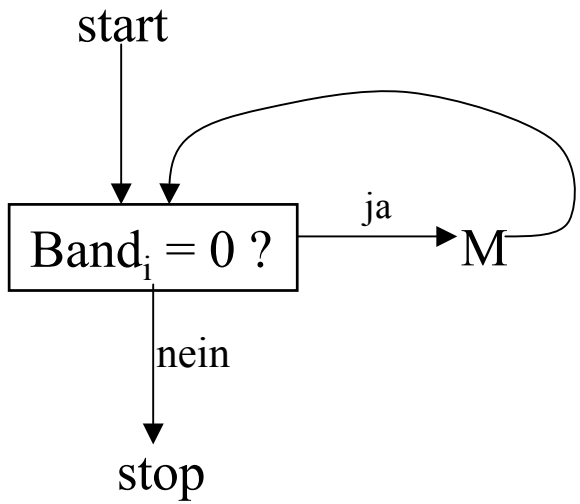
Kann eine Funktion nicht durch eine Turingmaschine berechnet werden, dann ist sie **überhaupt** nicht berechenbar.

Praktisches Konstruieren von TMen aus Teilmaschinen

$Band_i := Band_i+1$

$Band_i = 0$?





while $Band_i=0$ **do** M