Definition: Sei M eine NTM.

$C_2 = q_2, w_2\underline{a_2}u_2$ ist **Nachfolgekonfiguration** von $C_1 = q_1, w_1\underline{a_1}u_1$, symbolisch $C_1 \vdash_M C_2$, wenn es einen Übergang $(q_1, a_1)\Delta(q_2, b)$ gibt s.d.

- $(q_1, a_1), (q_2, b)$ die drei Bedingungen einer Nachfolgekonfiguration für deterministische TM erfüllen.

$\Rightarrow$ Bei NTM kann eine Konfiguration mehrere Nachfolgekonfigurationen besitzen.

$\Rightarrow$ Für gegebenen Input kann eine NTM unterschiedliche Rechnungswege besitzen.

$\Rightarrow$ Menge der Rechnungen einer NTM von einer Konfiguration C_0 kann als gerichteter Baum, R_{C_0}, betrachtet werden:

- C_0 ist Wurzel von R_{C_0};
- Söhne eines Knoten C sind die Nachfolgekonfigurationen von C.

$\Rightarrow$ Eine Rechnung von M ist ein Ast in R_{C_0} mit Startkonfiguration C_0.

Beispiel: Betrachte

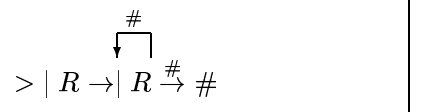
$$L = \{\ |^n\ \mid\ n \text{ ist nicht prim und } n \geq 2\}.$$

L kann mittels folgender NTM M akzeptiert werden:

- M rät nicht-deterministisch zwei Zahlen;
- multipliziert diese miteinander; und
- vergleicht das Ergebnis mit dem Eingabewort.

$\Longrightarrow M : > R\ Guess\ R\ Guess\ Mult\ Compare$, wo

- $Guess$ ist eine NTM die eine Zahl $n \geq 2$ rät, d.h. $s, \underline{\#} \vdash^*_{Guess} h, \# \mid^n \underline{\#}$:



- $Mult$ multipliziert (determiniert) Zahlen m und n, d.h. $s, \# \mid^m \# \mid^n \underline{\#} \vdash^*_{Mult} h, \# \mid^{m \cdot n} \underline{\#}$.
- $Compare$ vergleicht Zahlen m und n und hält genau dann, wenn beide gleich sind.

Satz: Jede Sprache, die von einer NTM akzeptiert wird, wird auch von einer Standard-TM akzeptiert.

Beweis: Sei L eine Sprache über Σ_0^* mit $\# \notin \Sigma_0$, und sei $M = (K, \Sigma, \Delta, s)$ eine NTM die L akzeptiert.

Wir konstruieren eine Standard-TM M' s.d.:

- M' durchläuft systematisch alle Rechnungen von M und sucht nach einer Haltekonfiguration von M.
- M' soll genau dann halten, wenn sie eine Haltekonfiguration von M findet.

Sei $C_0 = s, \#w\underline{\#}$ die Startkonfiguration von M und R_{C_0} der Rechnungsbaum von M mit Wurzelknoten C_0.

$\Longrightarrow R_{C_0}$ kann im Allgemeinen unendlich viele Knoten besitzen, aber jeder Knoten hat nur *endlich viele direkte Nachfolgeknoten*, nämlich maximal

$$r = \max\{|\Delta(q, a)| \mid q \in K, a \in \Sigma\}$$

viele.

M' durchsucht R_{C_0} mittels *iterative deeping*:

- M' verfolgt zunächst alle Äste der Tiefe 0, d.h., M' überprüft zunächst die Wurzel von R_{C_0};
- dann alle Äste bis Tiefe 1, bis Tiefe 2, etc.;
- M' hält sobald eine Haltekonfiguration gefunden ist.

M' kann als 3-Band-TM realisiert werden:

- Das erste Band enthält während der ganzen Rechnung unverändert das Eingabewort w.

  (Da die Rechnung immer wieder neu mit $s, \#w\underline{\#}$ beginnt , wird das Eingabewort immer wieder benötigt.)

- Das zweite Band enthält einen Zähler der den aktuellen Pfad des Rechnungsbaumes angibt.
  - Der Zähler kann z.B. durch Zahl im r-adischen System repräsentiert werden, wobei hier oBdA angenommen wird, daß jeder Knoten in R_{C_0} genau r Söhne besitzt.
    - $d_1 \dots d_n$ bedeutet, daß von C_0 aus der d_1-te Nachfolgeknoten C_{d_1} der Tiefe 1 gewählt wurde, dann der d_2-te von C_{d_1} der Tiefe 2, etc.
  - Am Beginn der Rechnung enthält Band 2 die Zahl 0 (= Wurzel C_0).
    - Addition der Zahl in Band 2 mit 1 im r-adischen System ergibt nächste zu betrachtende Rechnung.

- Band 3 simuliert Rechnung von M entsprechend dem Zähler $d_1 \dots d_n$ auf Band 2.
  - Falls Haltekonfiguration erreicht wird, stoppt M'; sonst wird Zähler auf Band 2 um eins erhöht.

Zusammenfassend gilt somit:

M' hält bei Input $w \iff R_{C_0}$ enthält eine Haltekonfiguration.

$\iff M$ hält bei Input w.

■

Beachte:

- die simulierende Maschine M' benötigt im Extremfall exponentiell mehr Schritte als die NTM M um ein Wort zu akzeptieren!

Turing-Maschinen und Sprachklassen

Problem:

- Welcher Klassen von Sprachen werden durch Turing-Maschinen beschrieben?

Erinnerung:

- In der Automatentheorie wurden bereits Zusammenhänge zwischen Sprachklassen und Automaten studiert.
- Die **Chomsky-Hierarchy** besteht aus folgenden vier Sprachklassen:

  Typ 3 $\subset$ Typ 2 $\subset$ Typ 1 $\subset$ Typ 0

- Es gilt:
  - Eine Sprache L ist vom Typ i wenn sie von einer Grammatik G vom Typ i erzeugt wird, d.h. falls $L = L(G)$.
  - Grammatik $G = (V, T, R, S)$:
    - V: Menge der Variablen
    - T: Menge der Terminale
    - R: Menge der Regeln
    - S: Startvariable

- Klassifikation der Grammatiken:
  - Typ 0: keine Einschränkungen
  - Typ 1 (kontextsensitiv):
    - Regeln der Form $P \to Q$ mit $|P| \leq |Q|$, oder $S \to \varepsilon$;
  - Typ 2 (kontextfrei):
    - Typ-1-Regeln der Form $P \to Q$, wo P eine einzelne Variable ist;
  - Typ 3 (regulär):
    - Typ-2-Regeln wo die rechten Seiten entweder
      - Terminalzeichen oder
      - ein Terminalzeichen gefolgt von einer Variablen sind.

- Es gilt:
  - L ist vom Typ 3 $\Longleftrightarrow$ L wird von einem endichen Automaten akzeptiert.
  - L ist vom Typ 2 $\Longleftrightarrow$ L wird von einem (nichtdeterministischen) Kellerautomaten (=*Push-Down-Automat*) akzeptiert.
- Wir werden zeigen:
  - L ist vom Typ 1 $\Longleftrightarrow$ L wird von einer *linear-beschränkten Turing-Maschine* akzeptiert.
  - L ist vom Typ 0 $\Longleftrightarrow$ L wird von einer (Standard) Turing-Maschine akzeptiert.

Definition: Ein **linear beschränkter Automat (LBA)** ist eine NTM $M = (K, \Sigma, \Delta, s)$ s.d. folgende zusätzliche Eigenschaften gelten:

- $\$, \S \in \Sigma$ ($ ist Anfangsmarkierung, § ist Endmarkierung des benutzten Bandstücks);
- $\forall q \in K\ \exists q_1, q_2 \in K$ s.d. $\Delta(q, \$) = \{(q_1, R)\}$ und $\Delta(q, \S) = \{(q_2, L)\}$;
- $\forall q, q' \in K,\ \forall a \in \Sigma$ gilt $(q', \$) \notin \Delta(q, a)$ und $(q', \S) \notin \Delta(q, a)$;
- Startkonfigurationen haben die Form $C_{start} = s, \$wa\S$ mit $w \in (\Sigma \setminus \{\#, \$, \S\})^*$ und $a \in (\Sigma \setminus \{\#, \$, \S\})$, oder $w = \varepsilon$ und $a = \#$.
  - Für $a \neq \#$ ist wa der Input von M, und für $a = \#$ ist ε der Input von M.

Informell:

- LBA arbeiten immer im Band zwischen den Marken $ und §;
- die Marken bleiben immer unverändert; und
- es werden nie Marken von der Maschine geschrieben.

Satz: Sei L eine Typ 0 Sprache. Dann gibt es eine TM M die L akzeptiert.

Beweis: Sei $G = (V, T, R, S)$ eine Typ 0 Grammatik die L erzeugt.

Wir konstruieren eine 2-Band TM M die L akzeptiert.

M arbeitet wie folgt:

1. Band 1 enthält ein Eingabewort w, und M erzeugt auf Band 2 das Startsymbol S von G:

$$s, \underline{\#}^{\#w\#} \vdash_M^* q, \#\underline{S}^{\#w\#}$$

2. Danach rät M auf Band 2 eine Ableitung in G: Zu $S \Longrightarrow_G^* u$ rechnet M

$$q, \#\underline{S}^{\#w\#} \vdash_M^* q', \#u\underline{\#}^{\#w\#}$$

Dabei geht M folgendermaßen vor:

- M wählt jeweils nicht-deterministisch eine Regel $P \to Q \in R$ sowie ein Vorkommnis von P auf Band 2 und ersetzt es durch Q.
  - Falls Q länger oder kürzer ist als P, verschiebt es dabei den Rest des Wortes rechts vom gewählten Vorkommnis von P entsprechend.

3. M vergleicht u und w: falls $u = w$, hält M, sonst hält M nie.

Zusammenfassend:

M hält bei Input w.
$\iff$ Es gibt eine Ableitung $S \Longrightarrow_G^* w$.
$\iff w \in L(G)$. ■

Satz: Sei L eine Typ 1 Sprache. Dann gibt es einen LBA M der L akzeptiert.

Beweis: Sei $G = (V, T, R, S)$ eine Typ 1 Grammatik die L erzeugt.

Wir konstruieren einen LBA $M' = (K, \Sigma, \Delta, s)$ der L akzeptiert, analog zur NTM M vom letzten Beweis.

Falls $\varepsilon \in L$, dann enthält G die Regel $S \to \varepsilon$.
$\Longrightarrow$ Wir setzen $\Delta(s, \#) = \{(h, \#)\}$.
Sonst sei $\Delta(s, \#) = \emptyset$.

M' hat 1 Band. Um ein Äquivalent zu 2 Bändern zu haben, legt M' zunächst eine zweite Spur an: Für alle $a \in T \setminus \{\#, \$, \S\}$ sei

$$\delta(s, a) = \{(s, \tfrac{a}{\#})\} \qquad \delta(s, \$) = \{(q_1, R)\}$$
$$\delta(s, \tfrac{a}{\#}) = \{(s, L)\} \qquad \delta(q_1, \tfrac{a}{\#}) = \{(q, \tfrac{a}{S})\}$$

$\Longrightarrow s, \$av\underline{b}\S \vdash_{M'}^* q, \$\overset{avb}{\underline{S}\ldots}\S$, mit $w = avb$.

Ab jetzt arbeitet M' wie M oben:

- Auf Spur 2 wird eine Rechnung $S \Longrightarrow_G^* u$ simuliert, wobei $|u| \leq |w|$.
- Nach nicht-deterministischer Zeit vergleiche $u = w$.
- M' hält falls $u = w$, sonst hält M' nie. ■

Satz: Sei L eine Sprache, die von einer TM akzeptiert wird. Dann ist L vom Typ 0.

Beweis: Sei $M = (K, \Sigma, \Delta, s)$ eine TM die L akzeptiert.

OBdA machen wir folgende Annahmen:

- M verwendet Grenzmarken \$ und § (wie LBA); allerdings wird § bei Bedarf nach rechts verschoben.
- Die Startkonfiguration von M für Input wa ist gegeben durch $s, \$w\underline{a}\S$.

$\Longrightarrow L$ sei eine Sprache über $T \subseteq \Sigma$ mit $\$, \S, \# \in \Sigma \setminus T$.

Wir konstruieren eine Grammatik $G = (V, T, R, S)$ mit $L = L(G)$.

Idee: G simuliert die Arbeitsweise von M.

- G erzeugt zunächst "zweispurig" ein beliebiges Wort $w \in T^*$, d.h. Zeichenketten der Form $\$\overset{w}{w}\S$.
  - Die obere Spur bleibt während Ableitung unverändert;
  - die untere Spur simuliert M, indem das Arbeitsband von M nachgebildet wird.

  $\Longrightarrow$ Dazu werden 3-Tupel-Variablen der Form $\overset{b}{\underset{q}{a}} \in \Sigma \times \Sigma \times K$ verwendet:
  - b ist Teil des Eingabewortes;
  - a ist aktuelle Kopfposition;
  - q ist aktueller Zustand.

- Wenn M einen Haltezustand erreicht, dann ist $w \in L$.

  $\Longrightarrow$ Die untere Spur wird gelöscht und das Anfangs erzeugte Wort w wird behalten.

Spezifikation von $G = (V, T, R, S)$:

Als Variablenmenge von G definieren wir

$$\begin{aligned} V &= \{S, A_1\} \cup \big(\Sigma \times (K \cup \{h, h_L\})\big) \cup \\ &\quad \big(\Sigma \times \Sigma \times (K \cup \{h\})\big) \cup (\Sigma \times \Sigma) \cup \{\$, \S\} \end{aligned}$$

Die Regelmenge R besteht aus folgenden Elementen:

1. Regeln zur Generierung des Inputs sowie der Repräsentation der Anfangskonfiguration:

   $$S \rightarrow \$A_1 \mid \$\overset{\#}{\underset{s}{\#}}\S \quad \text{(letztere Regel ist für Input } \varepsilon\text{)}$$

   $$A_1 \rightarrow \overset{a}{a}A_1 \mid \overset{a}{\underset{s}{a}}\S \qquad \forall a \in T$$

   Damit läßt sich bisher ableiten

   $$S \Longrightarrow_G^* \$\overset{w\,a}{\underset{s}{w\,a}}\S$$

   für Anfangskonfiguration $s, \$w\underline{a}\S$ von M.

2. Regeln zur Simulation des Arbeitsbandes von M:

   Für $q \in K$, $q' \in K \cup \{h\}$ und $a \in (T \cup \{\#\})$ enthält R folgende Regeln:

   - falls $(q, a)\Delta(q', a')$:

     $$\begin{matrix} b \\ a \\ q \end{matrix} \rightarrow \begin{matrix} b \\ a' \\ q' \end{matrix} \qquad \forall b \in (T \cup \{\#\});$$

   - falls $(q, a)\Delta(q', R)$:

     $$\begin{matrix} b & d \\ a & c \\ q & \end{matrix} \rightarrow \begin{matrix} b & d \\ a & c \\ & q' \end{matrix} \qquad \forall b, c, d \in (T \cup \{\#\});$$

     $$\begin{matrix} b \\ a \\ q \end{matrix}\S \rightarrow \begin{matrix} b & \# \\ a & \# \\ & q' \end{matrix}\S \qquad \forall b \in (T \cup \{\#\});$$

   - falls $(q, a)\Delta(q', L)$:

     $$\begin{matrix} d & b \\ c & a \\ & q \end{matrix} \rightarrow \begin{matrix} d & \\ c & b \\ q' & a \end{matrix} \qquad \forall b, c, d \in (T \cup \{\#\});$$

     $$\$\begin{matrix} b \\ a \\ q \end{matrix} \rightarrow \begin{matrix} \$ & \\ q' & b \\ & a \end{matrix} \qquad \forall b \in (T \cup \{\#\});$$

   - falls $(q, \$)\Delta(q', R)$:

     $$\begin{matrix} \$ & d \\ q & c \end{matrix} \rightarrow \$\begin{matrix} d \\ c \\ q' \end{matrix} \qquad \forall c, d \in (T \cup \{\#\}).$$

Es gilt für alle $w \in T^*$:

- M akzeptiert $w \iff$ es gibt $x_1, x_2 \in \Sigma^*$ und ein $b \in \Sigma$ s.d. $s, \$w\S \vdash_M^* h, \$x_1\underline{b}x_2\S$.
- Letzteres gilt genau dann, wenn

  $$w = ua = u_1cu_2$$

  und es gibt eine Ableitung in G der Form

  $$S \Longrightarrow_G^* \$\begin{matrix} u & a \\ u & a \\ & s \end{matrix}\S \Longrightarrow_G^* \$\begin{matrix} u_1 & & c \\ x_1 & b & u_2 \\ & h & x_2 \end{matrix}\#\cdots\#\S.$$

3. G enthält Regeln um die bisher erzeugten Zeichenketten in das Eingabewort umzuwandeln:
   - Löschen der unteren Zeilen:

     $$\begin{matrix} a & \\ b & c \\ h & d \end{matrix} \rightarrow \begin{matrix} a & \\ b & c \\ & h \end{matrix} \qquad \begin{matrix} a & \\ b & h_L \end{matrix} \rightarrow \begin{matrix} a \\ h_L \end{matrix} c$$

     $$\begin{matrix} a \\ b \\ h \end{matrix}\S \rightarrow \begin{matrix} a \\ h_L \end{matrix}\S \qquad \$\begin{matrix} a \\ h_L \end{matrix} \rightarrow \$\,a$$

   - Löschen der Endmarken und Blanks:

     $$\$ \rightarrow \varepsilon \qquad \S \rightarrow \varepsilon \qquad \# \rightarrow \varepsilon$$

Zusammenfassend:

M akzeptiert $w \iff G$ erzeugt w. ■

Satz: Sei L eine Sprache, die von einem LBA akzeptiert wird. Dann ist L vom Typ 1.

Beweis: Sei $M = (K, \Sigma, \Delta, s)$ ein LBA der L akzeptiert. Wir konstruieren eine Typ 1 Grammatik $G = (V, T, R, S)$ mit $L = L(G)$.

Die Konstruktion von G ist analog wie im letzten Beweis.

Unterschiede:

- Da LBA die Endmarke $\S$ nicht verschieben, ersetzen wir die Regel $\begin{matrix} b \\ a \\ q \end{matrix}\S \rightarrow \begin{matrix} b & \# \\ a & \# \\ & q' \end{matrix}\S$ durch

  $$\begin{matrix} b \\ a \\ q \end{matrix}\S \rightarrow \begin{matrix} b & \S \\ a & q' \end{matrix}$$

  und fügen

  $$\begin{matrix} b & \S \\ a & q \end{matrix} \rightarrow \begin{matrix} b \\ a \\ q' \end{matrix}\S$$

  für $(q, a)\Delta(q', L)$ im Fall $a = \S$ hinzu.

- Falls M das Wort ε akzeptiert, nehmen wir die Regel $S \to \varepsilon$ hinzu, sonst nicht.

  $\Rightarrow$ Die Regel $S \to \begin{smallmatrix}\$\\ \#\\ s\end{smallmatrix}\begin{smallmatrix}\#\\ \#\\ \S\end{smallmatrix}$ wird nicht benützt.

  $\Rightarrow$ Spur 1 enthält zwischen \$ und § nie ein #.

  $\Rightarrow$ Die nicht-Typ-1 Regel $\# \to \varepsilon$ wird nicht benötigt.

- Die nicht-Typ-1 Regeln $\$ \to \varepsilon$ und $\S \to \varepsilon$ können hier ebenfalls nicht verwendet werden.
  - Als Ersatz erweitern wir die Variablen-Tupel und vermerken \$ und § in einer vierten und fünften Spur:
    - Oben auf erstem Zeichen soll \$ stehen;
    - oben auf letztem Zeichen soll § stehen.

$\Longrightarrow$ Die initialen Regeln von G sind damit:

$$S \to \begin{smallmatrix}\$\\ a\\ a\end{smallmatrix} A_1 \mid \begin{smallmatrix}\S\\ \$\\ a\\ a\\ s\end{smallmatrix} \quad \text{(letztere Regel ist für Input } a\text{)}$$

$$A_1 \to \begin{smallmatrix}a\\ a\end{smallmatrix} A_1 \mid \begin{smallmatrix}\S\\ a\\ a\\ s\end{smallmatrix}$$

Damit läßt sich anfangs ableiten

$$S \Longrightarrow_G^* \begin{smallmatrix}\$\\ a\\ a\end{smallmatrix}\begin{smallmatrix}w\\ w\end{smallmatrix}\begin{smallmatrix}\S\\ b\\ b\\ s\end{smallmatrix} \qquad \text{bzw.} \qquad S \Longrightarrow_G^* \begin{smallmatrix}\S\\ \$\\ a\\ a\\ s\end{smallmatrix}.$$

- Die Simulation von M verläuft dann analog wie im vorigen Beweis.

  Bei Erreichen eines Haltezustand erzeugt G ein terminales Wort mittels folgender Regeln:

$$\begin{smallmatrix}a\\ b\\ h\end{smallmatrix}\begin{smallmatrix}c\\ d\end{smallmatrix} \to \begin{smallmatrix}a\\ b\\ h\end{smallmatrix}\begin{smallmatrix}c\\ d\end{smallmatrix} \qquad \begin{smallmatrix}a\\ b\end{smallmatrix}\begin{smallmatrix}c\\ h_L\end{smallmatrix} \to \begin{smallmatrix}a\\ h_L\end{smallmatrix} c$$

$$\begin{smallmatrix}\S\\ a\\ b\\ h\end{smallmatrix} \to \begin{smallmatrix}a\\ h_L\end{smallmatrix} \qquad \begin{smallmatrix}\$\\ a\\ b\end{smallmatrix}\begin{smallmatrix}c\\ h_L\end{smallmatrix} \to ac$$

$$\begin{smallmatrix}a\\ b\\ h\end{smallmatrix}\begin{smallmatrix}\S\\ c\\ d\end{smallmatrix} \to \begin{smallmatrix}a\\ h_L\end{smallmatrix} c \qquad \begin{smallmatrix}\S\\ \$\\ a\\ b\\ h\end{smallmatrix} \to a$$

■

Satz: Sei L eine Typ 1 Sprache. Dann ist das Wortproblem "$w \in L$?" entscheidbar.

Beweis: Da L vom Typ 1 ist, gibt es einen LBA M der L akzeptiert.

M arbeitet nur auf dem Bandstück zwischen den Grenzmarken \$ und §.

$\Rightarrow$ Es gibt nur endlich viele Konfigurationen die M annehmen kann.

$\Rightarrow$ Nach einer bestimmten Anzahl von Schritten kommt M in eine Schleife.

Man kann nun eine TM M' angeben, die den Rechnungsbaum von M mittels *iterative deepening* systematisch durchsucht, aber in keine Schleife kommt (Äste die sich wiederholen werden ignoriert).

$\Rightarrow$ Der Ableitungsbaum von M' ist endlich.

$\Rightarrow$ M' gibt "Y" aus falls ein Haltezustand von M gefunden wird, sonst "N". ■

Korollar: Für jede Typ 2 oder Typ 3 Sprache L ist das Wortproblem "$w \in L$?" entscheidbar.

Bemerkung:

- Bisherige Turing-Maschinen haben immer ein **vorgegebenes Programm**.
- Die Flexibilität von Rechnern ist aber, daß sie **universell** sind, d.h. daß sie beliebige Programme ausführen können.
- Man kann eine **universelle Turing-Maschine** U konstruieren, die als Eingabe das Programm einer beliebigen TM M hat, zusammen mit einem Wort w, und die die Arbeit von M auf w simuliert.
- Dazu verwendet man eine geeignete Kodierung die jeder TM bzw. jedem Input eine Zahl zuordnet ("gödelisierung"). Diese Zuordnung ist **umkehrbar eindeutig**.
- Wir werden später eine konkrete universelle TM angeben, zusammen mit einer geeigneten gödelisierung.