Theoretische Informatik 2

SoSe 2024

Berechenbarkeit und Komplexität

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

Universität Bremen

Das (spezielle) Halteproblem

- Das spezielle Halteproblem ist das Problem:
  - Gegeben eine Turingmaschine M, hält M auf ihrer eigenen Kodierung $\langle M \rangle$?
  - Formal: $H_s = \{\langle M \rangle \langle M \rangle \ : \ M$ Turingmaschine und M hält auf $\langle M \rangle\}$.

Satz

Das spezielle Halteproblem ist unentscheidbar.

Beweis: Diagonalisierung

(Many-one) Reduktionen

- Eine Reduktion von $L_1 \subseteq \Sigma^*$ auf $L_2 \subseteq \Sigma^*$ ist eine (totale) berechenbare Funktion

  $$f : \Sigma^* \to \Sigma^*,$$

  so dass für alle $w \in \Sigma^*$:

  $$w \in L_1 \Leftrightarrow f(w) \in L_2.$$

- L_1 ist auf L_2 reduzierbar, $L_1 \leq L_2$, falls es eine Reduktion von L_1 nach L_2 gibt.
- Anschaulich: L_1 ist ein kleines Problem im Vergleich zu L_2, L_1 ist nicht schwerer als L_2.

Reduktionen

Lemma

Seien $L_1, L_2 \subseteq \Sigma^*$ *mit* $L_1 \leq L_2$.

- L_2 *entscheidbar* $\Rightarrow$ L_1 *entscheidbar.*
- L_1 *unentscheidbar* $\Rightarrow$ L_2 *unentscheidbar.*

Lemma

Seien $L_1, L_2 \subseteq \Sigma^*$ *mit* $L_1 \leq L_2$.

- L_2 *semi-entscheidbar* $\Rightarrow$ L_1 *semi-entscheidbar*
- L_1 *nicht semi-entscheidbar* $\Rightarrow$ L_2 *nicht semi-entscheidbar.*

Das Halteproblem und das Wortproblem

- Das Halteproblem ist das Problem:
  - Gegeben eine Turingmaschine M und ein Wort w, hält M auf w?
  - Formal: $H = \{\langle M \rangle\, w \;:\; M \text{ Turingmaschine und } M \text{ hält auf } w\}$.
- Das Wortproblem ist das Problem:
  - Gegeben eine Turingmaschine M und ein Wort w, akzeptiert M das Wort w?
  - Formal: $W = \{\langle M \rangle\, w \;:\; M \text{ Turingmaschine und } w \in L(M)\}$.

Satz

Es gilt $H_s \leq H \leq W$.

Korollare

Satz

Es gilt $H_s \leq H \leq W$.

Also ist

- das Halteproblem unentscheidbar und das Wortproblem unentscheidbar.
- Beide Probleme sind semi-entscheidbar.
- Die Komplemente beider Probleme sind nicht semi-entscheidbar.

Heute: der Satz von Rice

Satz von Rice

Jede nichttriviale semantische Eigenschaft von Turingmaschinen ist unentscheidbar.

- Semantische Eigenschaft einer Maschine M: hängt nur von $L(M)$ ab.
- Nichttrivial: Es gibt Maschine mit der Eigenschaft und Maschine ohne die Eigenschaft.
- Beispiele:
  - $L(M) = \varnothing$.
  - $L(M)$ ist unendlich.
- Beispiele für nicht semantische Eigenschaften:
  - M hat 42 Zustände.

Formal: Eigenschaften von Sprachen

- Eigenschaft der semi-entscheidbaren Sprachen über Alphabet Σ:

$$\mathcal{P} \subseteq \{L \subseteq \Sigma^* \; : \; L \text{ semi-entscheidbar}\}$$

  - Erinnerung: L semi-entscheidbar $\Leftrightarrow$ es ex. TM M mit $L(M) = L$.

- Beispiele:
  - $\mathcal{P}_\varnothing = \varnothing$ und $\mathcal{P}_{all} = \{L \subseteq \Sigma^* \; : \; L \text{ semi-entscheidbar}\}$ sind die trivialen Eigenschaften.
    - Für jede TM M gilt $L(M) \notin \mathcal{P}_\varnothing$.
    - Für jede TM M gilt $L(M) \in \mathcal{P}_{all}$.
  - $\mathcal{P}_{fin} = \{L \subseteq \Sigma^* \; : \; L \text{ ist endlich}\}$.
  - $\mathcal{P}_{reg} = \{L \subseteq \Sigma^* \; : \; L \text{ ist regulär}\}$.

Formal: Semantische Eigenschaften von Maschinen

- Sei $\mathcal{P} \subseteq \{L \subseteq \Sigma^* \ : \ L \text{ semi-entscheidbar}\}$ eine Eigenschaft von semi-entscheidbaren Sprachen.
- Eine Turingmaschine M hat die semantische Eigenschaft $\mathcal{P}$, wenn

$$L(M) \in \mathcal{P}.$$

- Beispiele: Sei M eine Turingmaschine
  - M hat nicht die Eigenschaft $\mathcal{P}_{\emptyset}$ aber die Eigenschaft $\mathcal{P}_{all}$.
  - M hat die Eigenschaft $\mathcal{P}_{fin} = \{L \subseteq \Sigma^* \ : \ L \text{ ist endlich}\}$ genau dann, wenn $L(M)$ endlich ist.
  - M hat die Eigenschaft $\mathcal{P}_{reg} = \{L \subseteq \Sigma^* \ : \ L \text{ ist regulär}\}$ genau dann, wenn $L(M)$ regulär ist.

Formal: Semantische Eigenschaften von Maschinen

- Die folgenden Eigenschaften sind **keine** semantischen Eigenschaften:
  - M hat 42 Zustände.
  - M terminiert auf jeder Eingabe.
  - M verwirft das Wort 101010.
    - Es gibt M' mit $L(M) = L(M')$ und M' terminiert nicht auf 101010.

Der Satz von Rice

Satz von Rice

Sei $\mathcal{P} \subseteq \{L \subseteq \Sigma^* \ : \ L \text{ semi-entscheidbar}\}$ eine nichttriviale Eigenschaft von semi-entscheidbaren Sprachen über einem Alphabet Σ. Dann ist die Frage:

- Gegeben eine Turingmaschine M, gilt $L(M) \in \mathcal{P}$?
- Formal: $\langle \mathcal{P} \rangle = \{\langle M \rangle \ : \ M \text{ Turingmaschine mit } L(M) \in \mathcal{P}\}$.

unentscheidbar.

- Turingmaschinen = Programme: keine nichttriviale semantische Eigenschaft von Programmen kann automatisch getestet werden!
- Weitreichende Konsequenzen z.B. für Programmverifikation: automatische Prüfung auf Korrektheit ist unmöglich!

Beweis Satz von Rice

Beweis.

- $\langle \mathcal{P} \rangle = \{\langle M \rangle \ : \ M$ Turingmaschine mit $L(M) \in \mathcal{P}\}$.
- Wir zeigen $H \leq \langle \mathcal{P} \rangle$.
  - Annahme: $\varnothing \notin \mathcal{P}$ (sonst zeigen wir stattdessen $\bar{H} \leq \langle \mathcal{P} \rangle$).
- Da das Halteproblem unentscheidbar ist, folgt $\langle \mathcal{P} \rangle$ unentscheidbar.
- Suchen: total berechenbare Funktion f, die zu jeder Eingabe $\langle M \rangle\, w$ für das Halteproblem eine Maschine M' berechnet, so dass

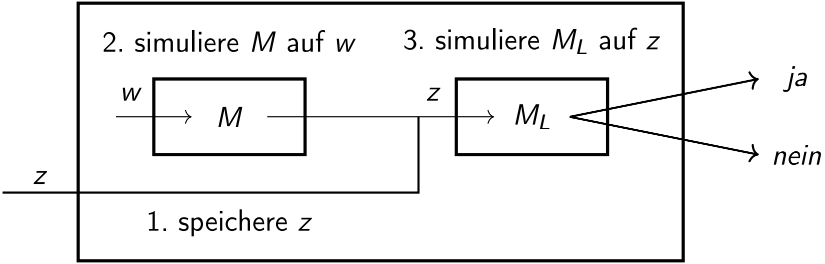
$$M \text{ hält auf } w \Leftrightarrow L(M') \in \mathcal{P}.$$

Beweis Satz von Rice

- $\mathcal{P}$ nichttrivial: es existiert $L \in \mathcal{P}$ und Maschine M_L mit $L(M_L) = L$.
- Zu Eingabe $\langle M \rangle\, w$:
$$f(\langle M \rangle\, w) = \left\langle M' \right\rangle,$$
wobei M' die Maschine ist, die auf Eingabe z
    - z auf einem unbeschriebenen Band speichert,
    - M auf w simuliert
    - wenn M auf w hält, so simuliert M' die Maschine M_L auf z und akzeptiert genau dann, wenn M_L akzeptiert.
- $\langle M' \rangle$ ist aus $\langle M \rangle\, w$ und der festen Maschine M_L berechenbar, also ist f eine totale berechenbare Funktion.

Beweis Satz von Rice

$M' = f(\langle M \rangle w)$



M hält nicht auf w

$\Rightarrow$ die Simulation von M auf w terminiert nicht

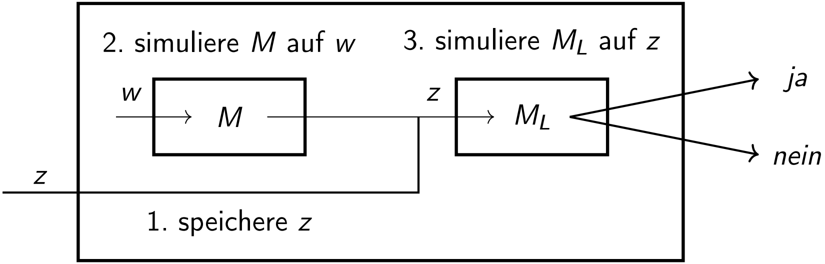
$\Rightarrow$ die Eingabe z wird (für beliebiges z) von M' nicht akzeptiert

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

$\Rightarrow$ $L(M') \notin \langle \mathcal{P} \rangle$, da $\varnothing \notin \mathcal{P}$.

Beweis Satz von Rice

$M' = f(\langle M \rangle w)$



M hält auf w

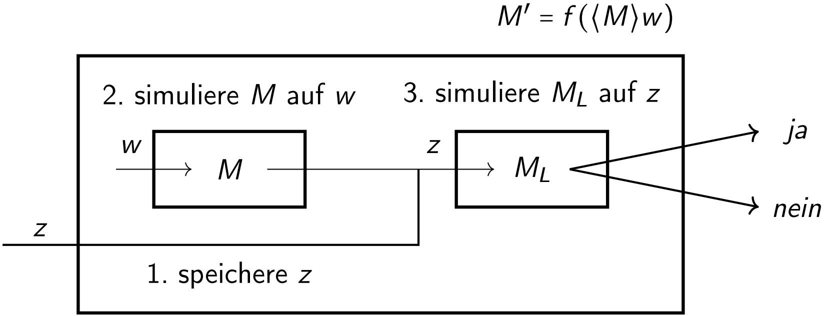
$\Rightarrow$ die Simulation von M auf w terminiert

$\Rightarrow$ M_L auf Eingabe z wird simuliert und M' akzeptiert genau dann, wenn $z \in L$

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

$\Rightarrow$ $L(M') \in \mathcal{P}$.

Beweis Satz von Rice



- M hält auf $w \Leftrightarrow L(M') \in \mathcal{P}$.
- Reduktion gefunden ✓

Der Satz von Rice

- Trotzdem kann für einige Programme bewiesen werden, dass sie gewisse semantische Eigenschaften haben.
- Beispiel: Für $M = (Q, \Sigma, \Gamma, [, \sqcup, \delta, q_s, q_a, q_r)$ mit $q_s = q_r$ gilt $L(M) = \varnothing$.
- Im Allgemeinen ist es aber nach dem Satz von Rice unentscheidbar, ob für eine gegebene Maschine M gilt $L(M) = \varnothing$.

Der Satz von Rice, zweiter Teil

- Eigenschaft $\mathcal{P}$ heißt nach oben abgeschlossen, falls für alle semi-entscheidbaren $L \subseteq L' \subseteq \Sigma^*$ mit $L \in \mathcal{P}$ auch gilt $L' \in \mathcal{P}$.
- Beispiele:
  - $\mathcal{P}_1 = \{\varnothing\}$ ist **nicht** nach oben abgeschlossen.
  - $\mathcal{P}_5 = \{L \subseteq \Sigma^* \ : \ L$ semi-entscheidbar und $|L|$ endlich$\}$ ist **nicht** nach oben abgeschlossen.
  - $\mathcal{P}_4 = \{L \subseteq \Sigma^* \ : \ L$ semi-entscheidbar und $|L|$ unendlich$\}$ ist nach oben abgeschlossen.

Der Satz von Rice, zweiter Teil

Satz von Rice, zweiter Teil

Sei $\mathcal{P} \subseteq \{L \subseteq \Sigma^* \ : \ L \text{ rekursiv aufzählbar}\}$ eine nichttriviale nicht nach oben abgeschlossene Eigenschaft von semi-entscheidbaren Sprachen.
Dann ist die Frage:

- Gegeben eine Turingmaschine M, gilt $L(M) \in \mathcal{P}$?
- Formal: $\langle \mathcal{P} \rangle = \{\langle M \rangle \ : \ M \text{ Turingmaschine mit } L(M) \in \mathcal{P}\}$.

nicht semi-entscheidbar.

Beweis.

- Wir zeigen $\bar{H} \leq \langle \mathcal{P} \rangle$.
- Das Komplement des Halteproblems ist nicht semi-entscheidbar, also folgt die gewünschte Aussage.

Beweis Satz von Rice, zweiter Teil

- Suchen: total berechenbare Funktion f, die zu jeder Eingabe $\langle M \rangle w$ für das Halteproblem eine Maschine M' berechnet, so dass

$$M \text{ hält nicht auf } w \Leftrightarrow L(M') \in \mathcal{P}.$$

- Äquivalent:

$$M \text{ hält auf } w \Leftrightarrow L(M') \notin \mathcal{P}.$$

- $\mathcal{P}$ nichttrivial und nicht nach oben abgeschlossen:

  $\Rightarrow$ es existieren M_0 und M_1, so dass
  $L(M_0) \subseteq L(M_1)$ und $L(M_0) \in \mathcal{P}$ und $L(M_1) \notin \mathcal{P}$.

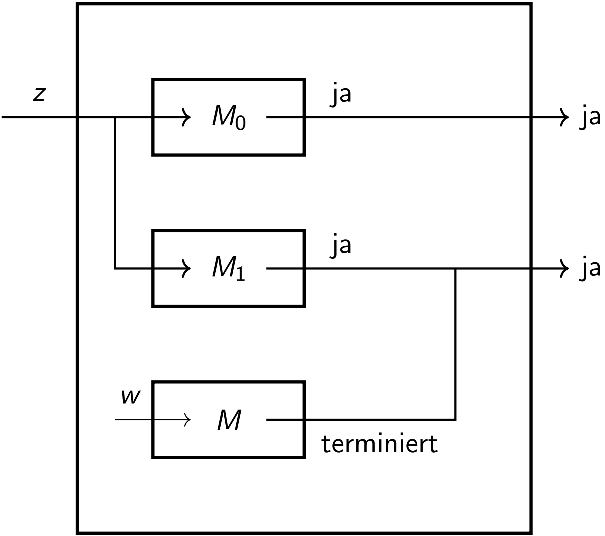
Beweis Satz von Rice, zweiter Teil

- Berechne zu $\langle M \rangle\, w$ Kodierung $\langle M' \rangle$, so dass

$$M \text{ auf } w \text{ hält} \Leftrightarrow L(M') \notin \mathcal{P}.$$

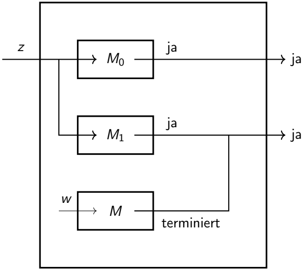
- M' Maschine, die zu Eingabe z
  - auf dem ersten Band M_0 auf z simuliert
  - auf dem zweiten Band M_1 auf z simuliert
  - auf dem dritten Band M auf w simuliert.
  - Simulation parallel, immer einen Schritt auf dem ersten Band, einen Schritt auf dem zweiten Band, einen Schritt auf dem dritten Band, usw.
  - M' akzeptiert genau dann, wenn
    - M_0 akzeptiert z, oder
    - M_1 akzeptiert z und M hält auf w.

Beweis Satz von Rice, zweiter Teil



Beweis Satz von Rice, zweiter Teil

- M' akzeptiert genau dann, wenn
  - M_0 akzeptiert z, oder
  - M_1 akzeptiert z und M hält auf w.



- M hält nicht auf w

  $\Rightarrow$ M' akzeptiert z genau dann, wenn M_0 z akzeptiert.

  $\Rightarrow$ $L(M') = L(M_0)$

- M hält auf w

  $\Rightarrow$ M' akzeptiert z genau dann, wenn M_0 oder M_1 z akzeptieren.

  - $L(M_0) \subseteq L(M_1)$

  $\Rightarrow$ M' akzeptiert z genau dann, wenn $z \in L(M_1)$

  $\Rightarrow$ $L(M') = L(M_1)$.

Beweis Satz von Rice, zweiter Teil

- Wir haben gezeigt:

$$
\begin{aligned}
& M \text{ hält auf } w \\
\Rightarrow\ & L(M') = L(M_1) \\
\Rightarrow\ & L(M') \notin \mathcal{P}
\end{aligned}
$$

  und

$$
\begin{aligned}
& M \text{ hält nicht auf } w \\
\Rightarrow\ & L(M') = L(M_0) \\
\Rightarrow\ & L(M') \in \mathcal{P}.
\end{aligned}
$$

- Also

$$M \text{ auf } w \text{ hält} \Leftrightarrow L(M') \notin \mathcal{P}$$

  wie gewünscht.

Der Satz von Rice

Korollar

Das Leerheitsproblem für Turingmaschinen, d.h., die Frage ob für eine gegebene Turingmaschine M gilt $L(M) = \varnothing$, ist nicht semi-entscheidbar.

Korollar

Das Äquivalenzproblem von Turingmaschinen, d.h., die Frage ob für zwei gegebene Turingmaschinen M_1, M_2 gilt $L(M_1) = L(M_2)$, ist nicht semi-entscheidbar.

Beweis.

- Sei $M_\varnothing$ eine Turingmaschine mit $L(M_\varnothing) = \varnothing$.
- Es gilt $L(M) = \varnothing \Leftrightarrow L(M) = L(M_\varnothing)$.

Entscheidungsprobleme für Grammatiken

Repräsentation	Wortproblem	Leerheitsproblem	Äquivalenzproblem
Typ-0-Grammatik, TM	semi-entscheidbar, unentscheidbar	nicht semi-entscheidbar	nicht semi-entscheidbar
Typ-1-Grammatik, linear beschr. NTM	PSpace-vollständig	nicht semi-entscheidbar	nicht semi-entscheidbar
Typ-2-Grammatik, PDA	Polynomialzeit	Polynomialzeit	nicht semi-entscheidbar
det. PDA	Linearzeit	Polynomialzeit	entscheidbar
Typ-3-Grammatik, NEA, reg. Ausdr.	Linearzeit	Linearzeit	PSpace-vollständig
DEA	Linearzeit	Linearzeit	Polynomialzeit

Entscheidungsprobleme für Grammatiken

Satz

Für die Typ-2 Grammatiken ist nicht semi-entscheidbar, ob eine gegebene Grammatik G alle Wörter erzeugen kann, ob also $L(G) = \Sigma^*$ gilt.

Korollar

Das Äquivalenzproblem für Typ-2 Grammatiken ist nicht semi-entscheidbar.

Beweis.

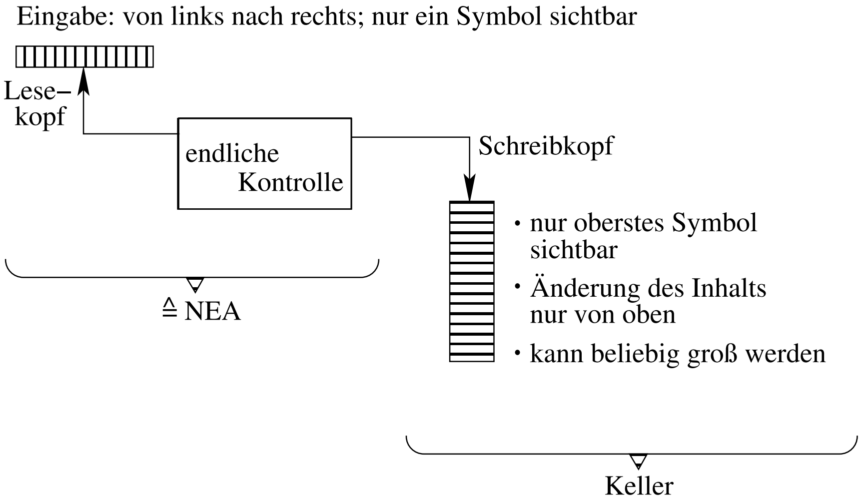
- $L(G) = \Sigma^* \Leftrightarrow L(G) = L(G_{all})$ für eine beliebige feste Typ-2 Grammatik G_{all}, die alle Wörter erzeugen kann.

Beweis des Satzes

- Wir reduzieren das Komplement des Halteproblems $\bar{H}$ auf das Universalproblem $L(G) = \Sigma^*$ für kontextfreie Grammatiken G.
- Kontextfreie Grammatiken sind äquivalent zu Kellerautomaten/Pushdown-Automaten und die Umwandlung ist berechenbar.
- Wir zeigen: Zu jedem Wort $\langle M \rangle\, w$ können wir einen Pushdown-Automaten $\mathcal{A}$ berechnen, so dass

$$M \text{ nicht auf } w \text{ hält} \Leftrightarrow \mathcal{A} \text{ alle Wörter akzeptiert.}$$

Kellerautomaten/Pushdown-Automaten



Beweis des Satzes

- Es gelte $M = (Q, \Sigma, \Gamma, [, \sqcup, \delta, q_s, q_a, q_r)$.
- Kodieren Konfigurationen von M als endliche Wörter: Wenn

$$(q, v, k) \in Q \times \Gamma^\omega \times \mathbb{N}$$

eine Konfiguration von M ist und $n \geq k$ minimal mit $v(i) = \sqcup$ für alle $i > n$, so repräsentieren wir dies als endliches Wort

$$\alpha = v_1 \dots v_{k-1} q v_k \dots v_n \sqcup \in (Q \cup \Gamma)^*.$$

- Alphabet des Pushdown-Automaten $\mathcal{A}$: $\Sigma' = Q \cup \Gamma \cup \{\vdash\}$.

Beweis des Satzes

- $\mathcal{A}$ soll für jedes Wort $v \in \Sigma'^*$ testen, ob

$$v = \alpha_1 \vdash \alpha_2 \vdash \ldots \vdash \alpha_n,$$

  wobei

  - α_1 die Kodierung der Startkonfiguration von M auf w ist,
  - $\alpha_i \vdash_M \alpha_{i+1}$ für $1 \leq i < n$ gilt und
  - α_n eine terminierende Konfiguration ist (egal ob akzeptierend oder verwerfend).

- Er soll akzeptieren, wenn dies ***nicht*** der Fall ist.

Beweis des Satzes

M terminiert auf w

$\Rightarrow$ es existiert akzeptierende Konfigurationsfolge $\alpha_1 \vdash_M \alpha_2 \vdash_M \ldots \vdash_M \alpha_n$

$\Rightarrow$ $\mathcal{A}$ verwirft $v = \alpha_1 \vdash \alpha_2 \vdash \ldots \vdash \alpha_n$

$\Rightarrow$ Es gilt $L(\mathcal{A}) \neq \Sigma'^*$.

M terminiert auf w nicht

$\Rightarrow$ es existiert keine akzeptierende Konfigurationsfolge

$\Rightarrow$ $\mathcal{A}$ akzeptiert alle Wörter $v \in \Sigma'^*$

$\Rightarrow$ Es gilt $L(\mathcal{A}) = \Sigma'^*$.

Konstruktion von $\mathcal{A}$

- $\mathcal{A}$ soll akzeptieren, wenn
  - v nicht die Form $\alpha_1 \vdash \alpha_2 \vdash \ldots \vdash \alpha_n$ hat für Konfigurationen $\alpha_1, \ldots, \alpha_n$, oder
  - α_1 nicht die Startkonfiguration von M auf w ist, oder
  - α_n keine terminierende Konfiguration ist (egal ob akzeptierend oder verwerfend),oder
  - $\alpha_i \not\vdash_M \alpha_{i+1}$ für ein $1 \leq i < n$.
- Die ersten 3 Eigenschaften sind regulär $\rightarrow$ einfach.
- Die letzte Eigenschaft benötigt Nichtdeterminismus und den Keller.

Konstruktion von $\mathcal{A}$

- Wenn
$$(p, w, k) \in Q \times \Gamma^\omega \times \mathbb{N}$$
eine Konfiguration ist und
$$\delta(p, w_k) = (q, b, d) \in Q \times \Gamma \times \{-1, 0, 1\},$$
dann ist die *Folgekonfiguration* von α die Konfiguration
$$(q, w[k \mapsto b], k + d).$$
- Das Infix *cpa* von α wird ersetzt durch das Infix *qcb*, *cqb* oder *cbq*.
- Alle anderen Zeichen bleiben unverändert.
- Wenn dies nicht der Fall ist, soll $\mathcal{A}$ das feststellen.

Konstruktion von $\mathcal{A}$

- $\alpha_i \not\vdash_M \alpha_{i+1}$

$\Rightarrow$ es existiert j, so dass das

$$\alpha_i(j)\alpha_i(j+1)\alpha_i(j+2) \quad \longrightarrow \quad \alpha_{i+1}(j)\alpha_{i+1}(j+1)\alpha_{i+1}(j+2)$$

nicht entsprechend δ.

- $\mathcal{A}$ rät α_i und die Position j.
- Er merkt sich die Zeichen $\alpha_i(j)\alpha_i(j+1)\alpha_i(j+2)$ und die Position j.
- Er läuft zur nächsten Konfiguration α_{i+1}, markiert durch $\vdash$, und zählt von dort $j-1$ Zeichen.
- Er verifiziert, dass $\alpha_{i+1}(j)\alpha_{i+1}(j+1)\alpha_{i+1}(j+2)$ nicht entsprechend δ ist und akzeptiert in diesem Fall.

Konstruktion von $\mathcal{A}$

- Phase 1: laufe über das Eingabewort. Bei $\vdash$ wechsele nichtdeterministisch in Phase 2.
  - Entspricht raten von α_i.
- Phase 2: laufe über das Eingabewort und lege für jedes gelesene Zeichen ein Symbol auf den Stack. Wechsele nichtdeterministisch zu Phase 3.
  - Entspricht raten und speichern von j.
- Phase 3: speichere die nächsten 3 Zeichen $\alpha_i(j)\alpha_i(j+1)\alpha_i(j+2)$ und laufe bis zum nächsten gelesenen $\vdash$.
- Phase 4: entferne für jedes gelesene Zeichen ein Symbol vom Stack. Wenn der Stack leer sind die nächsten 3 Zeichen $\alpha_{i+1}(j)\alpha_{i+1}(j+1)\alpha_{i+1}(j+2)$. Überprüfe ob diese δ entsprechen.

Zusammenfassung

- $\mathcal{A}$ akzeptiert, genau dann, wenn
  - v nicht die Form $\alpha_1 \vdash \alpha_2 \vdash \ldots \vdash \alpha_n$ hat für Konfigurationen $\alpha_1, \ldots, \alpha_n$, oder
  - α_1 nicht die Startkonfiguration von M auf w ist, oder
  - α_n keine terminierende Konfiguration ist (egal ob akzeptierend oder verwerfend), oder
  - $\alpha_i \nvdash_M \alpha_{i+1}$ für ein $1 \leq i < n$.
- Es gilt

$$M \text{ hält nicht auf } w \Leftrightarrow \mathcal{A} \text{ akzeptiert alle Wörter.}$$

Entscheidungsprobleme für Grammatiken

Repräsentation	Wortproblem	Leerheitsproblem	Äquivalenzproblem
Typ-0-Grammatik, TM	semi-entscheidbar, unentscheidbar	nicht semi-entscheidbar	nicht semi-entscheidbar
Typ-1-Grammatik, linear beschr. NTM	PSpace-vollständig	nicht semi-entscheidbar	nicht semi-entscheidbar
Typ-2-Grammatik, PDA	Polynomialzeit	Polynomialzeit	nicht semi-entscheidbar
det. PDA	Linearzeit	Polynomialzeit	entscheidbar
Typ-3-Grammatik, NEA, reg. Ausdr.	Linearzeit	Linearzeit	PSpace-vollständig
DEA	Linearzeit	Linearzeit	Polynomialzeit