Rückrichtung, Beweisidee

Für die zweite Richtung ergibt sich das folgende Problem:

Wir wollen mit einer Grammatik, die ja Wörter *erzeugt*, eine Maschine, die Wörter *erkennt*, simulieren.

Am Anfang unseres Prozesses müssen wir also das Wort erraten, das wir eigentlich erzeugen wollen – dann können wir die Rechnung des LBA durch die Grammatik in geeigneter Weise nachempfinden, und am Ende können wir das anfangs geratene Wort wieder erzeugen.

Das bedeutet allerdings, dass wir uns das zu Beginn geratene Wort die ganze Zeit über „merken“ müssen!

Idee: Nutze die Technik der *Spuren* in den Variablen, d.h. Menge V kann z.B. durch $V \cup (V \times \Sigma)$ ersetzt werden. (Details gleich...)

1. Phase der Worterzeugung

Wir gehen aus vom (nichtdeterm.) LBA M mit $L = T(M)$, und wollen eine Grammatik G vom Typ-1 konstruieren, für die gilt: $L(G) = T(M)$, also $L(G) = L$.

In der ersten Phase erzeugen wir in der 1. Spur ein (geratenes) Zielwort $x_1 x_2 \ldots x_n$, und in der 2. Spur gleichzeitig die hierzu passende Anfangskonfiguration des LBA in geeigneter Kodierung:

$$S \Rightarrow^* \binom{x_1}{(z_0,x_1)} \binom{x_2}{x_2} \binom{x_3}{x_3} \cdots \binom{x_{n-1}}{x_{n-1}} \binom{x_n}{\hat{x}_n}$$

Der Einfachheit halber können wir generell annehmen, dass nur Wörter der Länge mindestens 2 in der Simulation erzeugt werden müssen – Wörter der Länge 1 hinterher einzeln hinzuzufügen ist einfach! Also sei ab hier generell $n \geq 2$.

Danach soll in der zweiten Phase die 1. Spur unverändert bleiben, während wir in der 2. Spur eine Rechnung des LBA simulieren.

2. und 3. Phase

Wann endet die 2. Phase?

Die 2. Phase kann enden, wenn der LBA ohne Akzeptierung stoppt. In diesem Fall endet auch unser Erzeugungsprozess ohne Resultat.

Oder der LBA akzeptiert: Dann wissen wir, dass das anfangs geratene Wort $x_1 \dots x_n$ zu L gehört, d.h. in einer dritten Phase wollen wir die 2. Spur komplett entfernen und das Wort $x_1 \dots x_n$ erzeugen.

Wir setzen nun den Plan um, indem wir die Grammatik G wie folgt konstruieren:

$G = (V, \Sigma, P, S)$ mit $V = \{S, A\} \cup (\Sigma \times \Delta)$. Dabei ist

$$\Delta = \Gamma \cup (Z \times \Gamma)$$

Die Regelmenge P entwickeln wir für die drei Phasen getrennt auf den nächsten Folien.

Regeln der 1. Phase

Für jedes $a \in \Sigma$ gehören die folgenden drei Regeln zu P:

$$S \to A(a, \hat{a})$$

$$A \to A(a, a)$$

$$A \to (a, (z_0, a))$$

Man kann sich leicht vergewissern, dass diese Regeln uns in die Lage versetzen, für jedes Wort $x = x_1 \dots x_n$ aus Σ^* mit $n \geq 2$ die gewünschte Ausgangsposition

$$\binom{x_1}{(z_0,x_1)} \binom{x_2}{x_2} \binom{x_3}{x_3} \dots \binom{x_{n-1}}{x_{n-1}} \binom{x_n}{\hat{x}_n}$$

zu erzeugen (und auch nur solche Situationen!)

Regeln der 2. Phase

Nun sollen in der 2. Phase Rechenschritte des LBA auf der 2. Spur nachempfunden werden, ohne die 1. Spur zu verändern.

Sei also z.B. $(z', c, N) \in \delta(z, a)$. Dann brauchen wir diese Regel:

$$\binom{b}{(z,a)} \quad \to \quad \binom{b}{(z',c)}$$

Wenn aber $(z', c, R) \in \delta(z, a)$ ist, dann benötigen wir:

$$\binom{b}{(z,a)}\binom{b'}{a'} \quad \to \quad \binom{b}{c}\binom{b'}{(z',a')}$$

Den Fall $(z', c, L) \in \delta(z, a)$ sollte man jetzt selbst einfügen können.

Zu beachten ist bei diesen Übergängen, dass sie jeweils für alle $b \in \Sigma$, $b' \in \Sigma$ und $a' \in \Gamma$ gebraucht werden. Die Regelmenge wird also im allgemeinen recht groß. Wichtig für uns ist aber nur, dass sie endlich bleibt!

Regeln der 3. Phase

Am Schluss müssen wir die 1. Spur als Ergebnis deklarieren. Dass der Schluss erreicht ist, erkennen wir daran, dass ein akzeptierender Endzustand in der 2. Spur zu finden ist:

$$\binom{b}{(z,a)} \quad \to \quad b$$

Diese Regeln gibt es also für alle $z \in E$, sowie $b \in \Sigma$ und $a \in \Gamma$.

Auch alle anderen Zeichen aus der 1. Spur müssen „nach unten“ geholt werden:

$$\binom{b}{a} \quad \to \quad b$$

für alle $b \in \Sigma$ und $a \in \Gamma$.

Mit dieser Definition der Grammatik G ist es dann nicht allzu schwierig, die Gleichheit $L(G) = T(M)$ nachzuweisen.

Turingmaschinen und Typ-0

Wenn man den Beweis des Satzes von Kuroda durchgeht und darauf achtet, wo die Einschränkungen der Typ-1 Grammatiken (nichtverkürzende Regeln) und des LBA (beschränktes Band) verwendet werden, stellt man fest, dass der gesamte Beweis 1:1 übernommen werden kann, wenn man auf beide Einschränkungen verzichtet. Damit hat man also den gleichen Satz für Typ-0 Grammatiken und beliebige Turingmaschinen:

Satz: Die Klasse der von Turingmaschinen akzeptierten Sprachen ist gleich der Klasse der Typ-0 Sprachen.

Man muss hierbei allerdings genau darauf achten, welches Verhalten die betreffenden Turingmaschinen insbesondere im Fall der Ablehnung des Inputs, d.h. wenn die Eingabe x nicht zur Sprache $T(M)$ gehören soll, aufweist. (Vgl. nächste Folie!)