Der Algorithmus für $\overline{L(G)}$

Die Grobform des Algorithmus für $\overline{L(G)}$ ist also diese:

$k := 0;$
$r(0) := 1;$
REPEAT $k := k + 1;$
berechne $r(k)$; (dabei ist $r(k-1)$ als bekannt vorausgesetzt)
UNTIL $r(k) = r(k-1)$;
IF erreichbar(x, k) THEN ablehnen;
sonst: akzeptieren.

Hierbei bedeutet $r(k)$ die Anzahl der Satzformen der Länge maximal $n = |x|$, die in höchstens k Schritten erreichbar sind.
Die Prozedur erreichbar wird auf Folie 43.5 beschrieben.

Zunächst geben wir aber den Algorithmus für berechne$(r(k))$ an.

Der Algorithmus berechne($r(k)$)

Der Algorithmus berechne($r(k)$) geht so:

```
counter:= 0;
FOR α := erste Satzform der Länge ≤ n TO
         letzte Satzform der Länge ≤ n DO
  IF erreichbar(α, k) THEN
          INCREASE counter;
RETURN r(k) :=counter;
```

Hier kann $r(k-1)$ wieder als bekannt vorausgesetzt werden

In der letzten Prozedur auf der nächsten Folie zeigen wir noch, wie erreichbar(α, k) realisiert wird, wobei festgestellt werden soll, ob die Satzform α in höchstens k Schritten erreichbar ist. Dabei muss sowohl im positiven, wie auch im negativen Fall auf mindestens einem Berechnungsweg eine (korrekte!) Antwort gegeben werden!

Der Algorithmus erreichbar(α, k)

Hier nun der Algorithmus erreichbar(α, k):

```
result:=false;          Hier ist r(k − 1) bekannt!
Zykle durch alle Satzformen der Länge ≤ n.
Errate die in k − 1 Schritten erreichbaren.
IF β erraten mit β = α
        oder β ⊢ α THEN result:=true;
IF r(k − 1) erreichbare Satzformen erraten THEN
    RETURN result;
ELSE Abbruch!
```

Dieser Algorithmus realisiert das auf Folie 38.2 angekündigte *Durchzykeln* und kann dadurch in reinem Nichtdeterminismus die korrekte Antwort auf die Erreichbarkeitsfrage geben, und das sowohl im JA- wie auch im NEIN-Fall.

Abschluss

Der Rest sind einfache technische Überlegungen:

Können alle angegebenen Algorithmen vom LBA auf dem ihm zur Verfügung stehenden Platz ausgeführt werden?

Die Anwort ist JA.

Man kann zum Beispiel mehrere Spuren vorsehen, etwa um sich die Zahl $r(k-1)$ zu merken, während $r(k)$ berechnet wird.

Die Überprüfung, dass das in allen Einzelschritten so funktioniert, ist eine empfohlene Aufgabe, mit der man das eigene Verständnis kontrollieren kann und soll!

Damit ist der Beweis abgeschlossen.

1.5 Tabellen

Welche Sprachklassen haben wir betrachtet?
Und mit welchen Mitteln haben wir sie beschrieben?

Zunächst haben wir die vier Chomsky-Klassen eingeführt: Typ-0 bis Typ-3 Grammatiken beschreiben entsprechende Sprachklassen.

Für Typ-3 haben wir zusätzlich die (nicht-) deterministischen endlichen Automaten, sowie reguläre Ausdrücke als äquivalente Beschreibungsmittel identifiziert.

Typ-2 wird durch kontextfreie Grammatiken und durch allgemeine Kellerautomaten (PDA) definiert. Die Teilklasse der durch DPDAs erkennbaren Sprachen bildet eine *Zwischenklasse.*

Schließlich haben wir mit dem Satz von Kuroda gesehen, dass Typ-1 und Typ-0 durch Turingmaschinen – in Variationen – zu beschreiben sind. Als Zwischenklasse erhalten wir die Klasse REC.

Tabelle *Beschreibungsarten*

Aus diesen Beschreibungen können wir die folgende Tabelle zusammenstellen:

Chomsky?	Sprachklasse	Grammatik	Maschinentyp	Sonstiges
Ja	Typ-3	regulär	DEA/NEA	Reguläre Ausdrücke
Nein	det. kontextfrei	LR(k)-Grammatik	DPDA	
Ja	Typ-2	kontextfrei	PDA	
Ja	Typ-1	nichtverkürzend	LBA	kontextsensitiv
Nein	Entscheidbare Sprachen	—	Turingmasch. mit JA/NEIN-Antw.	REC
Ja	Typ-0	beliebig	TM mit JA/?	r.e.

Determinismus vs. Nichtdeterminismus

Wir haben vier Maschinenmodelle betrachtet, von denen jedes eine deterministische und eine nichtdeterministische Variante hat, nämlich den endlichen Automat (Typ-3), den Kellerautomat (Typ-2), die linear beschränke Turingmaschine (Typ-1) und die unbeschränkte Turingmaschine (Typ-0).

Sowohl bei endlichen Automaten wie auch bei den unbeschränkten Turingmaschinen sieht man, dass die nichtdeterministische Variante durch die deterministische simuliert werden kann.

Also gilt $DEA = NEA$ und $(N)TM = DTM$.

Aus den Abschlusseigenschaften schließt man, dass $PDA \neq DPDA$ gilt, denn Typ-2 ist nicht abgeschlossen gegen Komplement.

Offen bleibt die Frage, ob $LBA = DLBA$ oder $LBA \neq DLBA$ gilt!

Tabelle *(Nicht-) Determinismus*

Die folgende Tabelle fasst diese Resultate zusammen:

Klassen	Nichtdeterm.	Deterministisch	äquivalent?
Typ-3	NEA	DEA	JA
Typ-2/DCFL	PDA	DPDA	NEIN
Typ-1/?	LBA	„DLBA“	offen
Typ-0	(N)TM	DTM	JA