

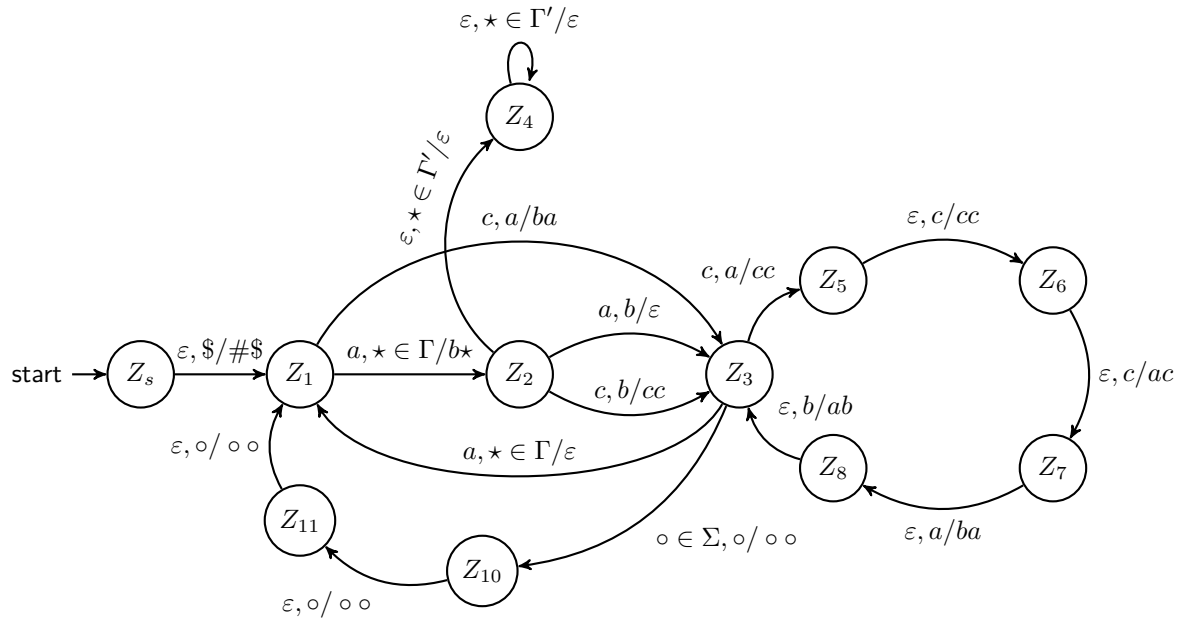
Informatik I: – Blatt 6

Rasmus Diederichsen

17. Juni 2014

Aufgabe 6.1

Es sei $\Gamma' = \Gamma \cup \{\$\}$.



Aufgabe 6.2

Schritt 1

$$\forall Z \in \mathcal{Z} : S \rightarrow R_{(Z_{start}, \#, Z)}$$

Es resultieren die Regeln

$$S \rightarrow R_{(Z_1, \#, Z_1)} \quad S \rightarrow R_{(Z_1, \#, Z_2)}$$

Schritt 2

$$\forall Z_0, \textcircled{Z_1} \xrightarrow{a, b/c} \textcircled{Z_2} : R_{(Z_1, b, Z_0)} \rightarrow aR_{(Z_2, c, Z_0)}$$

Es resultieren die Regeln

$R_{(Z_1, b, Z_1)} \rightarrow bR_{(Z_1, b, Z_1)}$	$R_{(Z_1, b, Z_2)} \rightarrow bR_{(Z_1, b, Z_2)}$
$R_{(Z_2, b, Z_1)} \rightarrow cR_{(Z_2, b, Z_1)}$	$R_{(Z_2, b, Z_2)} \rightarrow cR_{(Z_2, b, Z_2)}$
$R_{(Z_2, b, Z_1)} \rightarrow bR_{(Z_1, c, Z_1)}$	$R_{(Z_2, b, Z_2)} \rightarrow bR_{(Z_1, c, Z_2)}$

Schritt 3

$$\forall \text{Übergänge } \textcircled{Z_1} \xrightarrow{a, b/\varepsilon} \textcircled{Z_2} : R_{(Z_1, b, Z_2)} \rightarrow a$$

Es resultieren die Regeln

$R_{(Z_1, b, Z_1)} \rightarrow b$	$R_{(Z_1, \#, Z_1)} \rightarrow b$
$R_{(Z_1, c, Z_1)} \rightarrow c$	

$$\forall Z, Z', \text{Übergänge } \textcircled{Z_1} \xrightarrow{a, b/cd} \textcircled{Z_2} : R_{(Z_1, b, Z)} \rightarrow aR_{(Z_2, c, Z')}R_{(Z', d, Z)}$$

Gemäß $\forall X, Y : R_{(Z_1, \#, X)} \rightarrow aR_{(Z_2, b, Y)}R_{(Y, \#, X)}$ (es gibt im Graphen nur einen solchen Übergang) resultieren die Regeln

$R_{(Z_1, \#, Z_1)} \rightarrow aR_{(Z_2, b, Z_1)}R_{(Z_1, \#, Z_1)}$
$R_{(Z_1, \#, Z_1)} \rightarrow aR_{(Z_2, b, Z_2)}R_{(Z_2, \#, Z_1)}$
$R_{(Z_1, \#, Z_2)} \rightarrow aR_{(Z_2, b, Z_1)}R_{(Z_1, \#, Z_2)}$
$R_{(Z_1, \#, Z_2)} \rightarrow aR_{(Z_2, b, Z_2)}R_{(Z_2, \#, Z_2)}$

Der gesamte Regelsatz ist also

$S \rightarrow R_{(Z_1, \#, Z_1)}$	$S \rightarrow R_{(Z_1, \#, Z_2)}$
$R_{(Z_1, b, Z_1)} \rightarrow bR_{(Z_1, b, Z_1)}$	$R_{(Z_1, b, Z_2)} \rightarrow bR_{(Z_1, b, Z_2)}$
$R_{(Z_2, b, Z_1)} \rightarrow cR_{(Z_2, b, Z_1)}$	$R_{(Z_2, b, Z_2)} \rightarrow cR_{(Z_2, b, Z_2)}$
$R_{(Z_2, b, Z_1)} \rightarrow bR_{(Z_1, c, Z_1)}$	$R_{(Z_2, b, Z_2)} \rightarrow bR_{(Z_1, c, Z_2)}$
$R_{(Z_1, b, Z_1)} \rightarrow b$	$R_{(Z_1, \#, Z_1)} \rightarrow b$
$R_{(Z_1, c, Z_1)} \rightarrow c$	
$R_{(Z_1, \#, Z_1)} \rightarrow aR_{(Z_2, b, Z_1)}R_{(Z_1, \#, Z_1)}$	
$R_{(Z_1, \#, Z_1)} \rightarrow aR_{(Z_2, b, Z_2)}R_{(Z_2, \#, Z_1)}$	
$R_{(Z_1, \#, Z_2)} \rightarrow aR_{(Z_2, b, Z_1)}R_{(Z_1, \#, Z_2)}$	
$R_{(Z_1, \#, Z_2)} \rightarrow aR_{(Z_2, b, Z_2)}R_{(Z_2, \#, Z_2)}$	

Regeln, die tatsächlich etwas produzieren, sind aquamarin markiert. Regeln, die etwas produzieren könnten, von der Startvariable aus aber nicht erreichbar sind, sind grau hinterlegt. Wir kürzen ab

$$\begin{aligned}
R_1 &:= R_{(Z_1, \#, Z_1)} \\
R_2 &:= R_{(Z_1, c, Z_1)} \\
R_3 &:= R_{(Z_1, b, Z_1)} \\
R_4 &:= R_{(Z_2, b, Z_1)}
\end{aligned}$$

und destillieren hieraus

$$\begin{aligned}
S &\rightarrow R_1 \\
R_1 &\rightarrow b \mid aR_4R_1 \\
R_2 &\rightarrow c \\
R_4 &\rightarrow cR_4 \mid bR_2
\end{aligned}$$

und notieren, dass R_3 nicht erreichbar ist.

Aufgabe 6.3

Der (vermutlich falsche) Algorithmus läuft in $\mathcal{O}(|V|^2)$ da für jeden Knoten die Suche gestartet wird und in deren Verlauf alle anderen Knoten maximal zweimal besucht werden.

```

1 class DeCycle {
2     Graph g;
3     DeCycle(Graph g)

```

```

4      {
5          this.g = g;
6      }
7      void deCycle()
8      {
9          for(Vertex v : g.getVertices()) v.visited = false;
10         for(Vertex v : g.getVertices()) traverse(v, new ArrayList<
            Vertex>());
11     }
12     List<Vertex> getTargets(List<Arc>)
13     {
14         /* ... map Arcs to their targets */
15     }
16     void print()
17     {
18         /* output graph */
19     }
20     void traverse(Vertex v, List<Vertex> path)
21     {
22         path.add(v);
23         if(v.visited)
24             contractCycle(path.sublist(path.indexOf(v), path.length()));
25         else
26         {
27             v.visited = true;
28             for (Vertex neighbour : getTargets(g.getOutgoingArcs(v)))
29                 traverse(neighbour, new ArrayList<Vertex>(path));
30         }
31         v.visited = false; // reset for next search
32     }
33     static void main(String[] argv)
34     {
35         Graph g = new RandomGraph(Integer.parseInt(argv[0]));
36         DeCycle d = new DeCycle(g);
37         d.deCycle();
38         d.print();
39     }
40 }

```

Aufgabe 6.4

Die Ausgangsgrammatik ist

G

$$\begin{array}{lll}
S \rightarrow D & C \rightarrow eEd & E \rightarrow C \\
S \rightarrow dA & C \rightarrow B & \\
A \rightarrow aS & D \rightarrow CdDeB & \\
A \rightarrow S & D \rightarrow A & \\
A \rightarrow E & D \rightarrow d & \\
B \rightarrow db & E \rightarrow B & \\
B \rightarrow C & E \rightarrow BB &
\end{array}$$

Schritt 1: Erzeuge Regeln $A_x \rightarrow x$, ersetze a, \dots, e in G durch A_a, \dots, A_e .

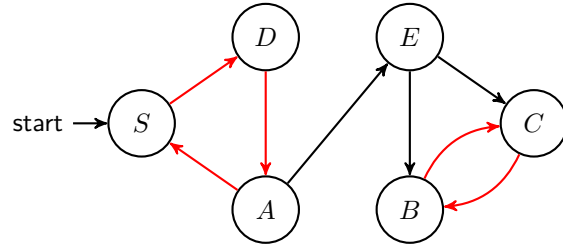
G'

$$\begin{array}{lll}
S \rightarrow D & C \rightarrow A_eEA_d & E \rightarrow C \\
S \rightarrow A_dA & C \rightarrow B & A_a \rightarrow a \\
A \rightarrow A_aS & D \rightarrow CA_dDA_eB & A_b \rightarrow b \\
A \rightarrow S & D \rightarrow A & A_d \rightarrow d \\
A \rightarrow E & D \rightarrow d & A_e \rightarrow e \\
B \rightarrow A_dA_b & E \rightarrow B & \\
B \rightarrow C & E \rightarrow BB &
\end{array}$$

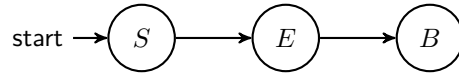
Schritt 2: Entferne Regeln der Form $V \rightarrow V$. Folgende Regeln sind zu eliminieren:

$S \rightarrow D$	$C \rightarrow A_eEA_d$	$E \rightarrow C$
$S \rightarrow A_dA$	$C \rightarrow B$	$A_a \rightarrow a$
$A \rightarrow A_aS$	$D \rightarrow CA_dDA_eB$	$A_b \rightarrow b$
$A \rightarrow S$	$D \rightarrow A$	$A_d \rightarrow d$
$A \rightarrow E$	$D \rightarrow d$	$A_e \rightarrow e$
$B \rightarrow A_dA_b$	$E \rightarrow B$	
$B \rightarrow C$	$E \rightarrow BB$	

Wir erstellen den zugehörigen Graphen.



Es existieren Zyklen $S \rightarrow D \rightarrow A \rightarrow S$ und $B \rightarrow C \rightarrow B$ bzw. $C \rightarrow B \rightarrow C$. Der kontrahierte Graph ist



Wir eliminieren die Senken beginnend mit B und transformieren so die Regelmengende $\{S \rightarrow E, E \rightarrow B\}$.

$E \rightarrow B$ wird zu

$$E \rightarrow A_d A_b$$

$$E \rightarrow A_e E A_d$$

$S \rightarrow E$ wird zu

$$S \rightarrow BB$$

$$S \rightarrow A_d A_b$$

$$S \rightarrow A_e E A_d$$

} vorherige Elimination

Die neue Grammatik ist

G''

$$S \rightarrow A_d S \mid A_a S \mid BB \mid A_d A_b \mid A_e E A_d \mid B A_d S A_e B \mid d$$

$$B \rightarrow A_d A_b \mid A_e E A_d$$

$$E \rightarrow A_d A_b \mid A_e E A_d \mid BB$$

$$A_a \rightarrow a$$

$$A_b \rightarrow b$$

$$A_d \rightarrow d$$

$$A_e \rightarrow e$$

Schritt 3: Kürzung zu langer Regeln. Wir führen für Regeln $V \rightarrow V_1, \dots, V_k$ mit $k > 2$ Regeln $V \rightarrow V_1 U_1, U_1 \rightarrow V_2 U_2, \dots, U_k \rightarrow V_{k-1} V_k$ ein und erhalten so

G'''

$$S \rightarrow A_d S \mid A_a S \mid BB \mid A_d A_b \mid A_e U_1 \mid B U_2$$

$$U_1 \rightarrow E A_d$$

$$U_2 \rightarrow A_d U_3$$

$$U_3 \rightarrow S U_4$$

$$U_4 \rightarrow A_e B$$

$$B \rightarrow A_d A_b \mid A_e U_1$$

$$E \rightarrow A_d A_b \mid A_e U_1 \mid BB$$

$$A_a \rightarrow a$$

$$A_b \rightarrow b$$

$$A_d \rightarrow d$$

$$A_e \rightarrow e$$

Diese Grammatik ist nun in Chomsky Normal Form.

Aufgabe 6.5

Man nehme an, das Pumping Lemma gilt, n bezeichne die zugehörige Wortmindestlänge. Sei $z = 1^n 2 1^n 2 1^n$. Es existiert nach Voraussetzung eine Zerlegung $uvwxy$ von z , sodass $|vx| \geq 1$, $|vwx| \leq n$ und $uv^iwx^iy \in L \forall i$

Die Fälle, in denen v, x verschiedene Symbole beinhalten (also sowohl 1en als auch 2en), braucht man gar nicht zu betrachten, da auf diese Weise Teilwörter der Form $(1^k 2)^i$ o.Ä. entstehen können, was natürlich nicht für alle i in L enthalten ist.

Es gibt vier Möglichkeiten, aus welchen Zeichen vwx bestehen kann.

1. $vwx = 1^k, k \leq n$. In diesem Fall kann durch Pumpen eine beliebige Anzahl 1en erzeugt werden, $uv^iwx^iy = 1^l 2 1^n 2 1^n$ für $l \neq n \notin L$.
2. $vwx = 1^k 2, k \leq n - 1$. In diesem Fall können durch Pumpen beliebig viele 1en erzeugt werden falls $|v| > 0$ und eventuell 2en, falls $|x| > 0$. Daher $uv^iwx^iy \notin L$.
3. $vwx = 1^k 2 1^l, 1 \leq k + l \leq n - 1$. In diesem Fall können durch Pumpen beliebig viele 1en erzeugt werden, eventuell auch beliebig viele 2en, falls $|v| > 1$ oder $|x| > 1$. Daher $uv^iwx^iy \notin L$, da der letzte Teil 1^n nicht mitgepumpt wird.
4. $vwx = 2 1^k, k \leq n - 1$. In diesem Fall können durch Pumpen beliebig viele 1en erzeugt werden, eventuell 2en, falls $|v| \geq 1$. Mithin $uv^iwx^iy \notin L$.

In keinem Fall ist $uv^iwx^iy \in L$, was einen Widerspruch zur Annahme darstellt.

Aufgabe 6.6

Bei der Umwandlung eines NDKA-AdLK in eine kontextfreie Grammatik können Regeln der folgenden Arten generiert werden

1. $V \rightarrow \Sigma$
2. $V \rightarrow V$
3. $V \rightarrow \Sigma \times V$
4. $V \rightarrow \Sigma \times V \times V$
5. $(V \rightarrow \varepsilon)$

Dieses Problem ist nur dann lösbar, wenn die Grammatik nicht linksrekursiv ist (also kein Regeln der Form $A \rightarrow AB$ enthält).

Um die Grammatik in GNF zu überführen, geht man wie folgt vor:

```
1 entferne Kreise in Regeln  $V_i \rightarrow V_j$ 
2 while exists Regel  $A \rightarrow V_i$ 
3   forall Regeln  $V_i \rightarrow *$ 
4     kreierte Regel  $A \rightarrow \text{rhs}(V_i)$ 
5   end
6   entferne  $A \rightarrow V_i$ 
7   entferne Regel  $V_i \rightarrow *$  falls !exists Regel  $R := V \rightarrow V^*$  mit  $V_i$  in
   rhs(R)
8 end
9 while exist Regel  $A \rightarrow V_1V_2$ 
10   forall Regeln  $V_1 \rightarrow *$  kreierte Regel  $A \rightarrow \text{rhs}(V_1)V_2$ 
11   entferne Regel  $A \rightarrow V_1V_2$ 
12 end
```