Quickies //alternative Normalform kontext-frei?

Q.1 //Tabellenerklärung CYK-Algorithmus-Tabelle?

Zur Orientierung in der CYK-Tabelle: Oben stehen in Zeile 0 die Terminalsymbole (Buchstaben) und die Tabelle ist links-aligned. Die Tabelle mit den Nicht-Terminalsymbol-Mengen beginnt mit Zeile 1 und Spalte 1.

Das Wortproblem kann für jedes zusammenhängende Teilwort leicht aus der Tabelle abgelesen werden. Für das Teilwort vom i-ten bis zum (i+j-1)-ten Buchstaben (Wortlänge j) nehme man den Tabelleneintrag in Zeile j und Spalte i und prüfe, ob dort das Startsymbol enthalten ist.

Warum: Gehe ich von der entsprechenden Tabellenzelle senkrecht nach oben, komme ich zum Wortanfang. Gehe ich von der entsprechenden Tabellenzelle diagonal nach oben rechts, komme ich zum Wortende.

Q.2 //Beispiele für CNF und GNF?

Beispiele	Ja, GNF	Nein, ¬ GNF
Ja, CNF	$S \to a$	$S \to SS$
Nein, ¬ CNF	$S \to aS$	$S \to Sa$

Aufgabe 1 //Kellerautomat Normalform?

1.1 //Beispiele?

- Jeder DEA ist in Kellerautomat-Normalform.
  In Übungsblatt 2 Aufgabe 2 steht ein Beispiel für einen DEA.

- Der in der Vorlesung besprochene Kellerautomat zur Erkennung der Sprache $L_1=\{w\$w^R \mid w \in (\Sigma \setminus \{\$\})^*\}$ der Palindrome mit Mittelkennzeichnung ist ein deterministischer Kellerautomat in Normalform.

- Der nachfolgende Kellerautomat zur Erkennung der Sprache $L_2=\{a^n b^{2n} \mid n \in \mathbb{N}\}$ ist nicht in Normalform: (PDA)$(Q,\Sigma,\Sigma_\#,\delta,q_0,\#)$ mit $Q:=\{q_0,q_1\} \wedge \Sigma := \{a,b\} \wedge \Sigma_\# := \{a,b,\#\}$
  $\forall A_\# \in \{a,\#\}:\ \delta(q_0,a,A_\#) = \{(q_0, aaA_\#)\}$
  $\forall q \in \{q_0,q_1\}:\ \delta(q,b,a) = \{(q_1,\epsilon)\}$
  $\forall q \in \{q_0,q_1\}:\ \delta(q,\epsilon,\#) = \{(q,\epsilon)\}$

- Der nachfolgende Kellerautomat zur Erkennung der Sprache $L_3=\{12345\}$ mit nur einem Wort ist nicht in Normalform:
  (PDA)$(Q,\Sigma,\Sigma_\#,\delta,q_0,\#)$ mit $Q:=\{q_0,q_1\} \wedge \Sigma := \{1,2,3,\ldots,8,9,0\} \wedge \Sigma_\# := \Sigma \cup \{\#\}$
  $\delta(q_0,\epsilon,\#) = \{(q_1, 12345)\}$
  $\forall x \in \Sigma:\ \delta(q_1,x,x) = \{(q_1,\epsilon)\}$

1.2 //Normalform äquivalent?

„⊇“: klar. Jede von einem PDA in Normalform erkannte Sprache kann auch von einem beliebigen PDA (nämlich dem gleichen) erkannt werden.

„⊆“: Sei $\delta(q,x,A) \ni (q', a_1 a_2 \ldots a_n)$ eine Übergangsregelung eines PDA nicht in Normalform. Dann ergänze die Knoten $q'_2, q'_3, \ldots q'_n$ und ersetze diese Regel durch:
$\delta^{NEU}(q,x,A) \ni (q'_n, a_n A)$
$\forall i \in \{3,4,\ldots,n\}:\ \delta^{NEU}(q'_i,\epsilon,a_i) \ni (q'_{i-1}, a_{i-1} a_i)$
$\delta^{NEU}(q'_2,\epsilon,a_2) \ni (q', a_1 a_2)$
Indem man diese Prozedur auf alle Übergangsregelungen anwendet, die nicht schon in Normalform sind, erhält man einen PDA in Normalform, der die gleiche Sprache erkennt.

Aufgabe 2 //deterministischer Kellerautomat?

$\delta(q_0, a, a) = \delta(q_0, b, b) = \{(q_0, \varepsilon)\}$

$\delta(q_0, \varepsilon, K) = \{(q_0, X) \mid X$ steht in der Tabelle$\}$

K = →	S	A	B
Tabelle:	aS, ε, AB	a, aa, aBa, aBBa	ε, bB

2.1 //ist Automat deterministisch?

Der Automat ist nicht-deterministisch, weil $\delta(q_0, \varepsilon, S) \geq 2$

2.2 //Äquivalente Grammatik?

Grammatik G = ({S, A, B}, {a, b}, P, S) mit

$S \rightarrow aS \mid AB \mid \varepsilon$

$A \rightarrow a \mid aa \mid aBa \mid aBBa$

$B \rightarrow bB \mid \varepsilon$

2.3 //Konfigurationssequenz?

$(q_0, aaba, S) \vdash (q_0, aaba, aS) \vdash (q_0, aba, S) \vdash (q_0, aba, aBa) \vdash (q_0, ba, Ba) \vdash (q_0, ba, ba) \vdash (q_0, a, a) \vdash (q_0, \varepsilon, \varepsilon)$

// Aufgabe 3 auf der nächsten Seite

Aufgabe 4 //reguläres Pumping-Lemma?

$$L_1 = \{a^n b^m \mid n,m \in \mathbb{N} \wedge n-m=42\} = \{a^{m+42} b^m \mid m \in \mathbb{N}\}$$

$$L_2 = \{a^{[n/2]} b^n \mid n \in \mathbb{N}\} = \{ \underbrace{a^m b^{2m}}_{2|n \;\wedge\; m=n/2}, \underbrace{a^m b^{2m+1}}_{2\nmid n \;\wedge\; m=(n-1)/2}, \underbrace{b}_{n=1} \mid m \in \mathbb{N}\}$$

$$L_3 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

Annahme: L_1 regulär, p Pumpingzahl

Setze $\vec{w} := a^{p+42} b^p \in L_1 \wedge |\vec{w}| = 2p+42 \geq p$

Pumping-Garantie: $\exists \vec{x}\vec{y}\vec{z} = \vec{w}$ mit $1 \leq |\vec{y}| \leq |\vec{x}\vec{y}| \leq p \wedge \vec{x}\vec{z} \in L_1$

Dann: $\vec{x}\vec{y} \in a^*$. Also $\exists 1 \leq i \leq p$: $\vec{y} = a^i$

Dann: $L_1 \ni \vec{x}\vec{z} = a^{p+42-i} b^p \notin L_1$ WIDERSPRUCH. Also L_1 nicht regulär.

Annahme: L_2 regulär, p Pumpingzahl

Setze $\vec{w} := a^{2p} b^{4p} \in L_2 \wedge |\vec{w}| = 6p \geq p$ // $a^{2p} b^{4p+1} \in L_2$

Pumping-Garantie: $\exists \vec{x}\vec{y}\vec{z} = \vec{w}$ mit $1 \leq |\vec{y}| \leq |\vec{x}\vec{y}| \leq p \wedge \vec{x}\vec{z} \in L_2$

Dann: $\vec{x}\vec{y} \in a^*$. Also $\exists 1 \leq i \leq p$: $\vec{y} = a^i$

Dann: $L_2 \ni \vec{x}\vec{z} = a^{2p-i} b^{4p} \notin L_2$ WIDERSPRUCH. Also L_2 nicht regulär.

Annahme: L_3 regulär, p Pumpingzahl

Setze $\vec{w} := a^p b^p c^p \in L_3 \wedge |\vec{w}| = 3p \geq p$

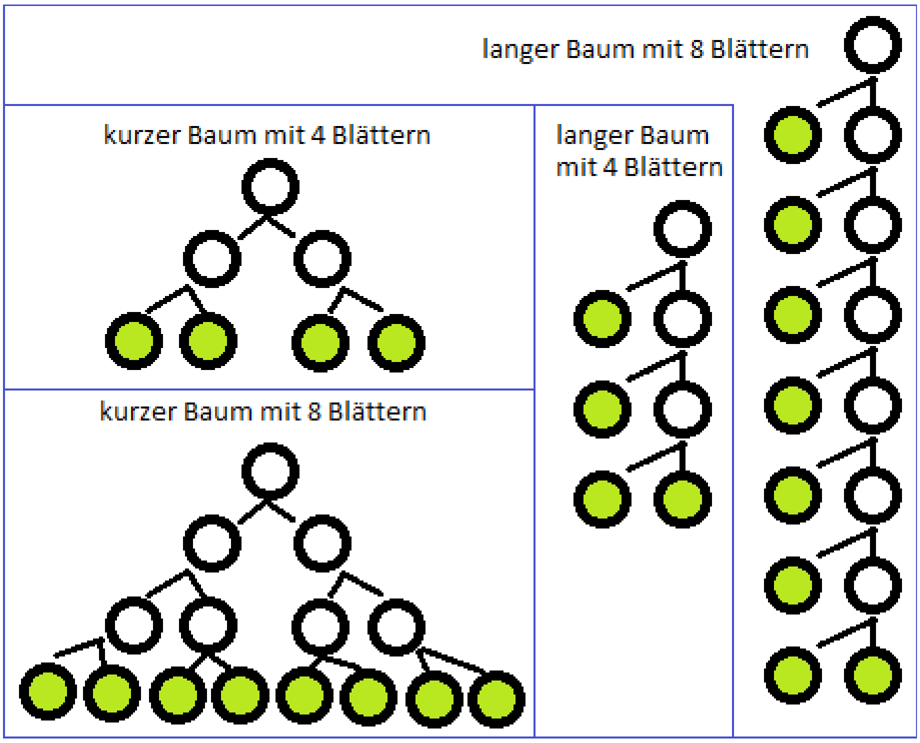
Pumping-Garantie: $\exists \vec{x}\vec{y}\vec{z} = \vec{w}$ mit $1 \leq |\vec{y}| \leq |\vec{x}\vec{y}| \leq p \wedge \vec{x}\vec{z} \in L_3$

Dann: $\vec{x}\vec{y} \in a^*$. Also $\exists 1 \leq i \leq p$: $\vec{y} = a^i$

Dann: $L_3 \ni \vec{x}\vec{z} = a^{p-i} b^p c^p \notin L_3$ WIDERSPRUCH. Also L_3 nicht regulär.

Aufgabe 3

3.1 //Baumschule?



3.2 //Datenstruktur Bäume

```
struct Baum {
 Content content;
 Baum knotenLinks = null;
 Baum knotenRechts = null;
}
```

3.3 //Baumcontent ausgeben

```
//liest vom Baum erst den linken Zweig, dann den eigenen Knoten, dann rechts.
function readBaum_LCR(Baum baum) returns Content_Kette {
 Content_Kette result = new Content_Kette();
 if (baum.knotenLinks != null) {
   Content_Kette resultLinks = readBaum_LCR(baum.knotenLinks);
   result.append(resultLinks);
 }
 result.append(baum.content);
 if (baum.knotenRechts != null) {
   Content_Kette resultRechts = readBaum_LCR(baum.knotenRechts);
   result.append(resultRechts);
 }
 return result;
}
```

3.4 //Baumcontent ausgeben

readBaum_LCR kann rückwärts ausgegeben werden, indem man erst den rechten Ast abgeht, dann den Content ausgibt, dann den linken Ast abgeht.