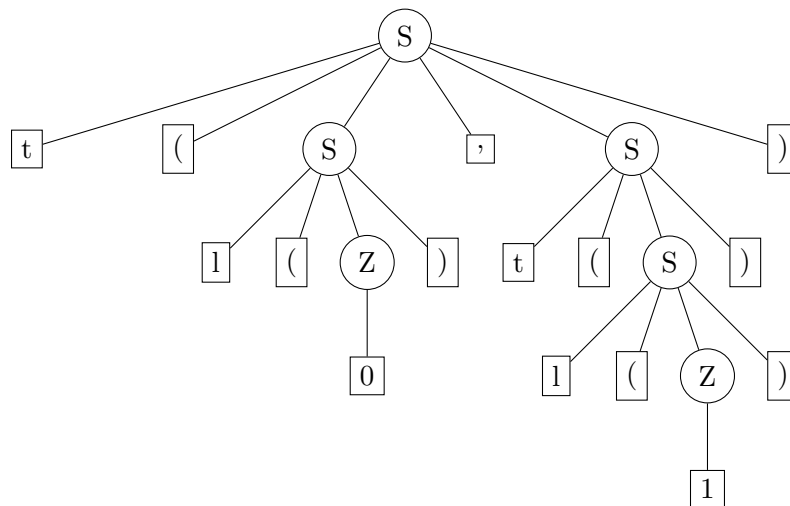


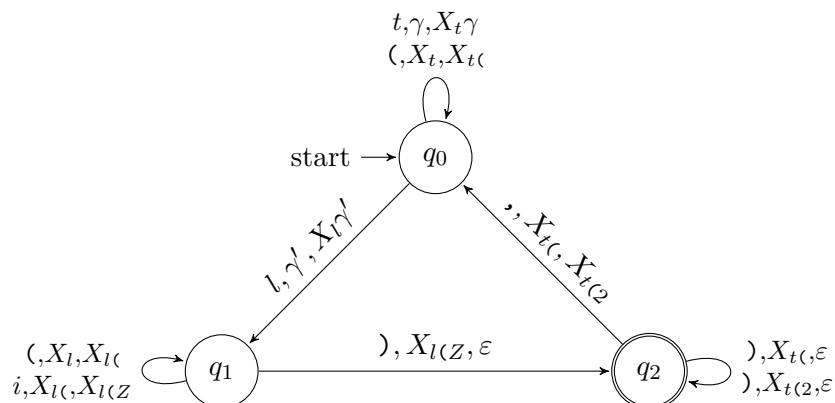
Hausaufgabe 11

Aufgabe 5

a)



c)



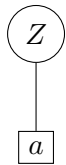
```

1: function PARSE TREE(text)
2:    $a \leftarrow$  first symbol of text
3:   initialise Stack  $S$ 
4:   while  $a \neq \text{EOF}$  do
5:     if  $a \in \{0, 1\}$  then
6:        $T \leftarrow \text{NUMTREE}(a)$ ; PUSH( $S, T$ )
7:     else if  $a \in \{\mathfrak{t}, \mathfrak{l}, (, ,\}$  then
8:       PUSH( $S, a$ )
9:     else if  $a = )$  then
10:       $T \leftarrow \text{REDUCE}(S)$ ; PUSH( $S, T$ )
11:    end if
12:  end while
13:   $T \leftarrow \text{POP}(S)$ 
14:  if  $T$  is a tree and  $S = \emptyset$  then
15:    return  $T$ 
16:  end if
17: end function
18:
19: function REDUCE( $S$ )
20:   $T_1 \leftarrow \text{POP}(S)$ 
21:  if  $T_1$  is a NumTree then
22:     $X_{\mathfrak{c}} \leftarrow \text{POP}(S)$ 
23:     $X_1 \leftarrow \text{POP}(S)$ 
24:    if  $X_{\mathfrak{c}} = ($  and  $X_1 = 1$  then
25:      return LEAF( $T$ )
26:    end if
27:  else if  $T_1$  is a S-Tree then
28:     $k \leftarrow \text{POP}(S)$ 
29:    if  $k = ,$  then
30:       $T_2 \leftarrow \text{POP}(S)$ 
31:       $X_{\mathfrak{c}} \leftarrow \text{POP}(S)$ 
32:       $X_{\mathfrak{t}} \leftarrow \text{POP}(S)$ 
33:      if  $T_2$  is a S-Tree and  $X_{\mathfrak{c}} = ($  and  $X_{\mathfrak{t}} = \mathfrak{t}$  then
34:        return BINARYTREE( $T_2, T_1$ )
35:      end if
36:    else if  $k = ($  then
37:       $X_{\mathfrak{t}} \leftarrow \text{POP}(S)$ 
38:      if  $X_{\mathfrak{t}} = \mathfrak{t}$  then
39:        return SINGLETREE( $T_1$ )
40:      end if
41:    end if
42:  end if
43: end function

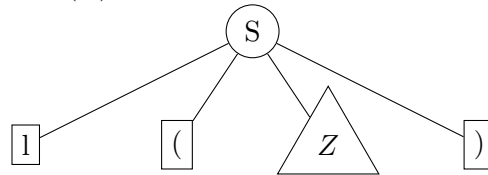
```

Dabei werden folgende Ableitungsbäume wiedergegeben:

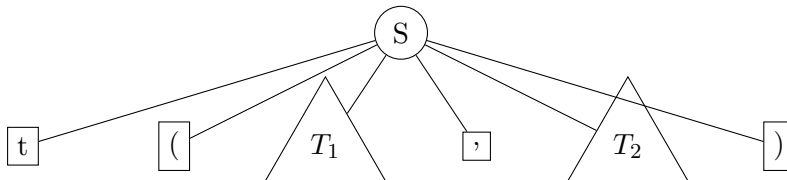
NumTree(a):



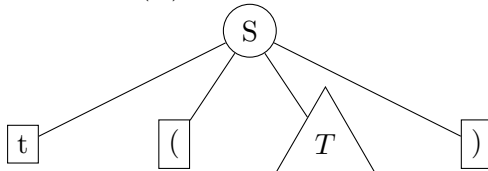
Leaf(Z):



BinaryTree(T_1, T_2):



SingleTree(T):



Aufgabe 6

a)

Wir markieren folgende Nichtterminale in angegebener Reihenfolge: C, A, D, G

Wir können keine weiteren Nichtterminale markieren, da sie eine zirkuläre Abhängigkeit haben:

Um S zu markieren, müssten wir B oder F markiert haben.

Um B zu markieren, müssten wir S oder F markiert haben.

Um F zu markieren, müssten wir B markiert haben.

b) Insbesondere ist die Sprache damit leer, da S kein terminierendes Nichtterminal ist.

Aufgabe 7

a

A,B,C	S	A	S,B	A	S,B
	b	B,C		A	S
		c	C	S	A
			a	A,B,C	S
				b	B,C
					c
					C

c

C			S	A
c	C		S	A
	a	A,B,C	S	A
		c	C	
			c	C

Damit gilt $abcabc \in L(\mathcal{G})$ und $ccacc \notin L(\mathcal{G})$.

Aufgabe 8

Sei eine kontextfreie Grammatik $\mathcal{G} = (N, \Sigma, P, S)$ gegeben. Wenn $|L(\mathcal{G})| = \infty$, so muss ein Wort $z \in L(\mathcal{G})$ mit $|z| \geq 2^{|N|+1}$ existieren, sodass der Ableitungsbaum von z mindestens Höhe $h \geq |N|+1$ hat, also eins der endlich vielen Nichtterminale mehr als einmal in der Ableitung von z vorkommt (Idee Pumping-Lemma). Andererseits haben wir nun endlich viele Nichtterminale und damit endlich viele Ableitungen, in denen jedes Nichtterminal höchstens ein mal vorkommt. Weiter gilt auch, dass wenn ein $z \in L(\mathcal{G})$ mit $|z| \geq n = 2^{|N|+1}$ existiert (wobei n eben genau das n aus dem Pumping-Lemma ist), dass dann das Pumping Lemma uns einer Zerlegung liefert, welche wir insbesondere beliebig oft "pumpen" können, um unendlich viele verschiedene Wörter in $L(\mathcal{G})$ zu erhalten.

Insgesamt haben wir also gezeigt, dass:

$$\exists z \in L(\mathcal{G}) : |z| \geq 2^{|N|+1} \quad \Longleftrightarrow \quad |L(\mathcal{G})| = \infty$$

Nun können wir wie folgt entscheiden, ob $L(\mathcal{G})$ zu einer gegebenen Grammatik \mathcal{G} unendlich ist: Es ist Σ endlich, also ist $\Sigma^n \Sigma^*$ ein regulärer Ausdruck, welcher alle Wörter über dem Eingabealphabet mit mindestens Länge $n := 2^{|N|+1}$ erkennt. Dann können wir einen DFA \mathcal{A} erstellen mit $L(\mathcal{A}) = L(\Sigma^n \Sigma^*)$. Weiter können wir aus der gegebenen Grammatik \mathcal{G} einen PDA \mathcal{B} mit $L(\mathcal{B}) = L(\mathcal{G})$ erstellen, welcher mit leerem Zustand akzeptiert. Dann können wir den PDA \mathcal{C} erstellen, welcher die Produktkonstruktion der beiden darstellt. Also $L(\mathcal{C}) = L(\mathcal{A}) \cap L(\mathcal{B})$. Zu diesem können wir nun wieder eine kontextfreie Grammatik \mathcal{G}' erstellen, sodass $L(\mathcal{G}') = L(\mathcal{C})$. Ferner ist also $L(\mathcal{G}') = \{z \mid z \in L(\mathcal{G}) \wedge |z| \geq n\}$. Wenn wir also mit dem Markierungsalgorithmus das Leerheitsproblem auf \mathcal{G}' lösen, so wissen wir, ob die Sprache $L(\mathcal{G})$ ein Wort der Länge $\geq n$ enthält. Nach dem obigen Beweis ist dann $L(\mathcal{G}') \neq \emptyset \implies |L(\mathcal{G})| = \infty$.

Somit können wir mit dem angegebenen Algorithmus testen, ob die Sprache einer gegebenen kontextfreien Grammatik unendlich ist.