Name: Matrikelnummer:

Probeklausur: Compilerbau

Aufgabe 1 Gegeben sei folgendes Fab4 Programm.

Last.fab4

```
1  constr True 0;
2  constr False 0;
3  constr Nil 0;
4  constr Cons 2;
5
6  last xs = case xs of
7   Cons y ys ->(case ys of
8                  Nil -> y
9                  Cons z zs -> last ys);
10 square x = x*x;
11
12 main = last (Cons (square 1) (Cons 2 Nil)))
```

Reduzieren Sie schrittweise den Ausdruck `main` in normaler Auswertungsreihenfolge.

Lösung

```
    main
→   last (Cons (square 1) (Cons 2 Nil)))
→   case (Cons (square 1) (Cons 2 Nil)) of
      Cons y ys ->(case ys of
                     Nil -> y
                     Cons z zs -> last ys)
→   case (Cons 2 Nil) of
           Nil -> (square 1)
           Cons z zs -> last (Cons 2 Nil))
→   last (Cons 2 Nil)
→   case (Cons 2 Nil) of
      Cons y ys ->(case ys of
                     Nil -> y
                     Cons z zs -> last ys)
→   case (Nil) of
          Nil -> 2
          Cons z zs -> last Nil
→   2
```

Aufgabe 2 Gegeben sei die Grammatik $\mathcal{G} = (\mathcal{T}, \mathcal{N}, S, \mathcal{R})$ mit:

- $\mathcal{T}$= { `and, or, not, true, false, (, )`}
- $\mathcal{N}$= {A,B,C}
- S =A
- $\mathcal{R}$={

$$A \rightarrow B \mid A \texttt{ or } B \quad (1)$$
$$B \rightarrow C \mid B \texttt{ and } C \quad (2)$$
$$C \rightarrow \texttt{(} A \texttt{)} \mid \texttt{not } C \mid \texttt{true} \mid \texttt{false} \quad (3)$$

}

a) Leiten Sie den Satz:
`true and (not false or false)`
mit der Grammatik ab.

Lösung

$$\begin{aligned}
& A \\
\rightarrow\ & B \\
\rightarrow\ & B \texttt{ and } C \\
\rightarrow\ & C \texttt{ and } C \\
\rightarrow\ & \texttt{true and } C \\
\rightarrow\ & \texttt{true and (}A\texttt{)} \\
\rightarrow\ & \texttt{true and (}A \texttt{ or } B\texttt{)} \\
\rightarrow\ & \texttt{true and (}B \texttt{ or } B\texttt{)} \\
\rightarrow\ & \texttt{true and (}C \texttt{ or } B\texttt{)} \\
\rightarrow\ & \texttt{true and (not } C \texttt{ or } B\texttt{)} \\
\rightarrow\ & \texttt{true and (not false or } B\texttt{)} \\
\rightarrow\ & \texttt{true and (not false or } C\texttt{)} \\
\rightarrow\ & \texttt{true and (not false or false)}
\end{aligned}$$

b) Zeichnen Sie für Ihre Ableitung den Syntaxbaum.

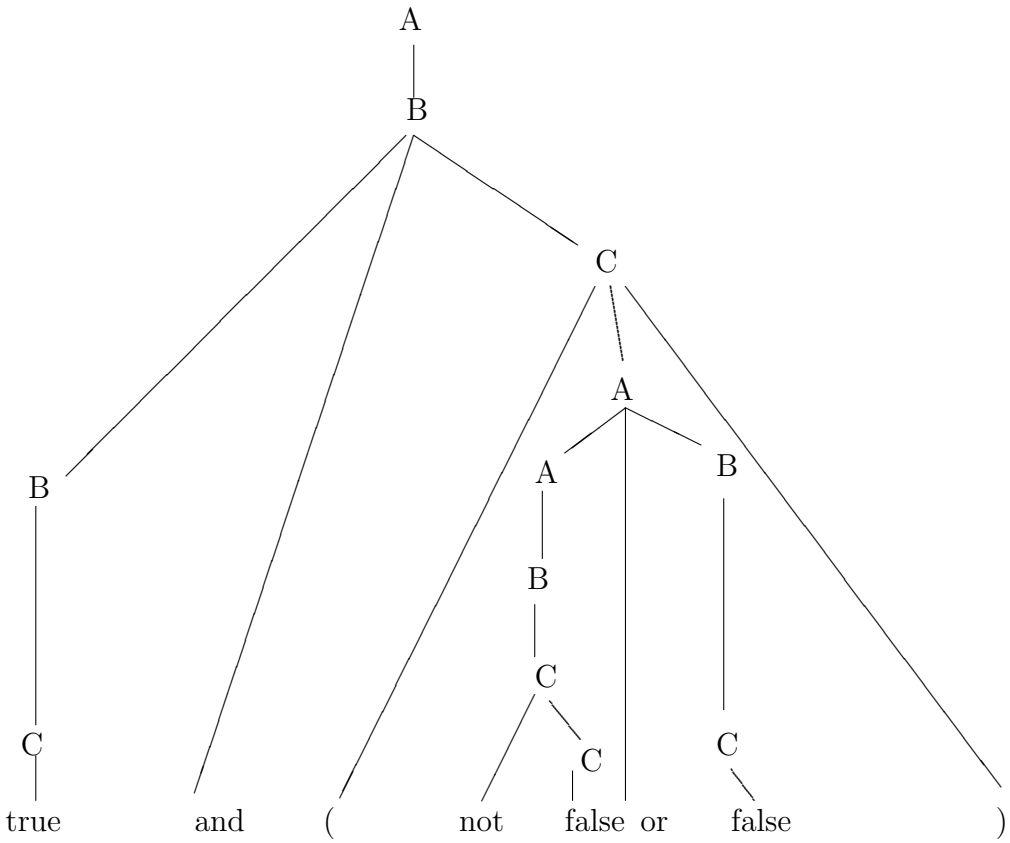


Abbildung 1: Syntaxbaum.

Lösung

Siehe Abbildung 1.

c) Berechnen Sie die First und Follow Mengen für die die verschiedenen Alternativen in den Produktionen der Grammatik:
A, B, C, `not` C, `(`A`)`, A `or` B, B `and` C.

Lösung

σ	First(σ)	Follow(σ)
A	`(`, `not`, `true`, `false`	`or`, `)`
B	`(`, `not`, `true`, `false`	`and`, `)`, `or`
C	`(`, `not`, `true`, `false`	`and`, `)`, `or`
`not`C	`not`	
`(`C	`(`	
B `and` C	`(`, `not`, `true`, `false`	
A `or` B	`(`, `not`, `true`, `false`	

d) Hat die Grammatik die LL(1) Eigenschaft?

Lösung

Nein! z.B. für Regel (1) ist der Schnitt der First-Mengen beider Regelalternativen nicht leer:
First(B)$\cap$First(A `or` B)$\neq \emptyset$

e) Warum können Sie für die Grammatik keinen rekursiv absteigenden Parser schreiben?

Lösung

Die Regeln (1) und (2) sind linksrekursiv. Rekursiv absteigende Parser terminieren nicht für linksrekursive Grammatiken.

f) Transformieren Sie die Grammatik so, daß für sie direkt ein rekursiv absteigender Parser geschrieben werden kann.

Lösung

Elemination von Linksrekursionen:

$$A \rightarrow B\ A2 \quad (4)$$
$$A2 \rightarrow \texttt{or}\ B \mid \epsilon \quad (5)$$
$$B \rightarrow C\ B2 \quad (6)$$
$$B2 \rightarrow \texttt{and}\ C \mid \epsilon \quad (7)$$
$$C \rightarrow \texttt{(}\ A\ \texttt{)} \mid \texttt{not}\ C \mid \texttt{true} \mid \texttt{false} \quad (8)$$

}

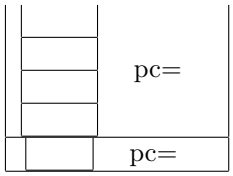
Aufgabe 3 Gegeben sei folgender Startzustand der G-Maschine.

Stack:

Heap:

0 (last)	`(NGlobal 1 0)`
1 (take)	`(NGlobal 2 22)`
2 (repeat)	`(NGlobal 1 60)`
3 (main)	`(NGlobal 0 68)`

Dump:



pc = 78
`PUSHGLOBAL main`

Der Code der Maschine sei:

0	`PUSH 0`	28	`CASEJUMP {1=4, 0=0}`	56	`JUMP 0`
1	`EVAL`	29	`SPLIT 0`	57	`UPDATE 2`
2	`CASEJUMP {3=0}`	30	`PACK 2 0`	58	`POP 2`
3	`SPLIT 2`	31	`SLIDE 0`	59	`UNWIND`
4	`PUSH 1`	32	`JUMP 24`	60	`PUSH 0`
5	`EVAL`	33	`SPLIT 0`	61	`PUSHGLOBAL repeat`
6	`CASEJUMP {2=0, 3=4}`	34	`PUSH 1`	62	`MKAP`
7	`SPLIT 0`	35	`EVAL`	63	`PUSH 1`
8	`PUSH 0`	36	`CASEJUMP {2=0, 3=4}`	64	`PACK 3 2`
9	`SLIDE 0`	37	`SPLIT 0`	65	`UPDATE 1`
10	`JUMP 6`	38	`PACK 2 0`	66	`POP 1`
11	`SPLIT 2`	39	`SLIDE 0`	67	`UNWIND`
12	`PUSH 3`	40	`JUMP 14`	68	`PUSHINT 42`
13	`PUSHGLOBAL last`	41	`SPLIT 2`	69	`PUSHGLOBAL repeat`
14	`MKAP`	42	`PUSH 1`	70	`MKAP`
15	`SLIDE 2`	43	`PUSHINT 1`	71	`PUSHINT 1000`
16	`JUMP 0`	44	`EVAL`	72	`PUSHGLOBAL take`
17	`SLIDE 2`	45	`PUSH 4`	73	`MKAP`
18	`JUMP 0`	46	`EVAL`	74	`MKAP`
19	`UPDATE 1`	47	`SUB`	75	`PUSHGLOBAL last`
20	`POP 1`	48	`PUSHGLOBAL take`	76	`MKAP`
21	`UNWIND`	49	`MKAP`	77	`UNWIND`
22	`PUSHINT 0`	50	`MKAP`	78	`PUSHGLOBAL main`
23	`EVAL`	51	`PUSH 1`	79	`EVAL`
24	`PUSH 1`	52	`PACK 3 2`	80	`PRINT`
25	`EVAL`	53	`SLIDE 2`	81	`OUTPUT "\n"`
26	`Eq`	54	`JUMP 0`	82	`NOP`
27	`EVAL`	55	`SLIDE 0`	83	`END`

Führen Sie den Code von Hand aus, bis Sie zu Befehl Nr. 23 kommen und zeichenen Sie den dann erreichten Maschinenzustand.

Lösung

(Werte die vom Stack gepopt wurden, sind durchgestrichen markiert.)

Stack:

10
~~1, 8, 1,~~ 6
~~2, 6, 7, 0, 0, 8, 7,~~ 5
~~3, 4, 5, 8, 9,~~ 8

Heap:

0	`(NGlobal 1 0)`
1	`(NGlobal 2 22)`
2	`(NGlobal 1 60)`
3	`(NGlobal 0 68)`
4	`42`
5	`NAp(2 4)`
6	`1000`
7	`NAp(1 6)`
8	`NAp(7 5)`
9	`NAp(0 8)`
10	`0`

Dump:

9 8	pc=2
pc=80	

pc = 23
`Eval`

Aufgabe 4 Gegeben Sei folgender Zustand der G-Maschine.

Stack:

18
15
20

Heap:

0	`(NGlobal 2 0)`
1	`(NGlobal 3 6)`
2	`(NGlobal 0 14)`
3	`(NGlobal 1 16)`
4	`(NGlobal 2 38)`
5	`(NGlobal 1 76)`
6	`(NGlobal 0 84)`
7	`42`
8	`NInd -> 16`
9	`1000`
10	`NAp(4 9)`
11	`NInd -> 21`
12	`NAp(3 11)`
13	`0`
14	`(Constr 1 [])`
15	`NAp(5 7)`
16	`(Constr 3 [ 7 15])`
17	`1`
18	`999`
19	`NAp(4 18)`
20	`NAp(19 15)`
21	`(Constr 3 [ 7 20])`

Dump:

12 11 20 7	pc=22
pc=96	

pc = 38
`PUSHINT 0`

Führen Sie eine *Two-Space-Copy-Garbage-Collection* durch und geben Sie den Maschinenzustand danach an. Eleminieren Sie während der *garbage collection* Indirektionsknoten.

Lösung

Stack:

9
8
7

Heap:

0	`(NGlobal 2 0)`
1	`(NGlobal 3 6)`
2	`(NGlobal 0 14)`
3	`(NGlobal 1 16)`
4	`(NGlobal 2 38)`
5	`(NGlobal 1 76)`
6	`(NGlobal 0 84)`
7	`NAp 13 8`
8	`NAp 5 10`
9	`999`
10	`42`
11	`(Constr 3 [10,7])`
12	`NAp 3 11`
13	`NAp 4 9`

Dump:

12 11 7 10	pc=22
	pc=96

pc = 38
`PUSHINT 0`