Lösungsvorschlag zur Klausur „Compilerbau“ vom 29. September 2014

Punkte ____ *von* **12**

Aufgabe 1 (3+3+3+3)

a) Was versteht man im Compilerbau unter einem „Parser“?

Eine Compilerkomponente, die für die Syntaxanalyse zuständig ist.

Welche Funktion hat er?

Syntaxfehler melden, eine Ableitung zum Quelltext gemäß einer zugrunde liegenden Grammatik berechnen. Ggf. weitere syntaxorientierte Aktionen ausführen, z.B. Aufbau eines abstrakten Systaxbaums.

Wie arbeitet er mit anderen Compilerkomponenten zusammen?

Der Scanner wird vom Parser bei Bedarf aufgerufen, um das nächste Token zu liefern. Die vom Parser aufgebaute Zwischendarstellung ist Ausgangspunkt für die Code-Erzeugung. Die semantische Analyse kann syntaxorientiert beim Parsen oder nach dem Parsen auf der Basis der vom Parser erzeugten Zwischendarstellung erfolgen.

b) Was bedeutet im Hinblick auf typisierte Programmiersprachen der Begriff „Überladung“?

Es gibt mehrere an der selben Stelle im Quelltext verwendbare Definitionen eines Bezeichners oder Operators

Geben Sie ein Beispiel an.

- Der Operator ”+” kann eine Gleitpunkt- oder Ganzzahladdition repräsentieren.
- Mehrere Konstruktoren für die Klasse String in Java, z.B. $String()$ und $String(char[]\ value)$.

Wie geht ein Compiler damit um?

Er muss anhand der an der Verwendungsstelle des überladenen Bezeichners oder Operators benutzten Argumenttypen die richtige Definition eindeutig zuordnen.

c) Was versteht man unter einer regulären Sprache?

Eine formale Sprache, die vom einem endlichen Automaten (DEA oder NEA) akzeptiert wird. Alternativ: Eine formale Sprache, die durch einen regulären Ausdruck spezifiziert ist.

Welche Bedeutung haben reguläre Sprachen im Compilerbau?

Die Spezifikation der Token-Syntax (Schlüsselwörter, Bezeichner, Literale, Operatoren usw.) erfolgt durch reguläre Ausdrücke. Ein Scannergenerator kann daraus eine auf endlichen Automaten basierte Erkennung der Tokens generieren.

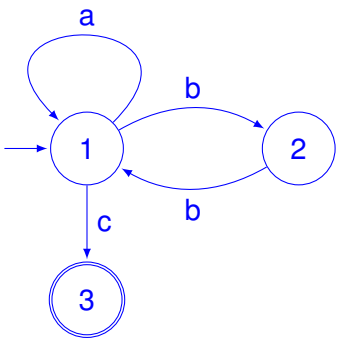
d) Ein Ausdruck ist ein syntaktisches Element, das im Quelltext einen Wert repräsentiert. Wie verarbeitet ein Compiler typischerweise einen Ausdruck (Frontend und Backend)?

Das Frontend muss die Syntax prüfen und dabei den Operatoren im Ausdruck die Operanden korrekt zuordnen. Der Typ muss bestimmt werden. Falls der Wert nicht direkt vom Compiler bestimmt werden kann, muss Maschinencode erzeugt werden, der zur Laufzeit den Wert berechnet.

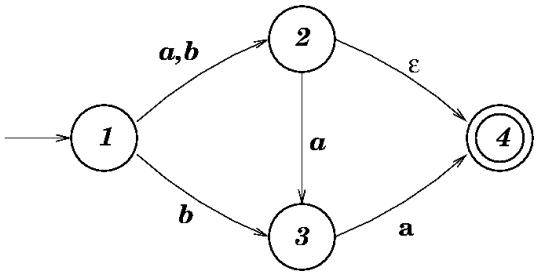
Aufgabe 2 (3+3 Punkte)

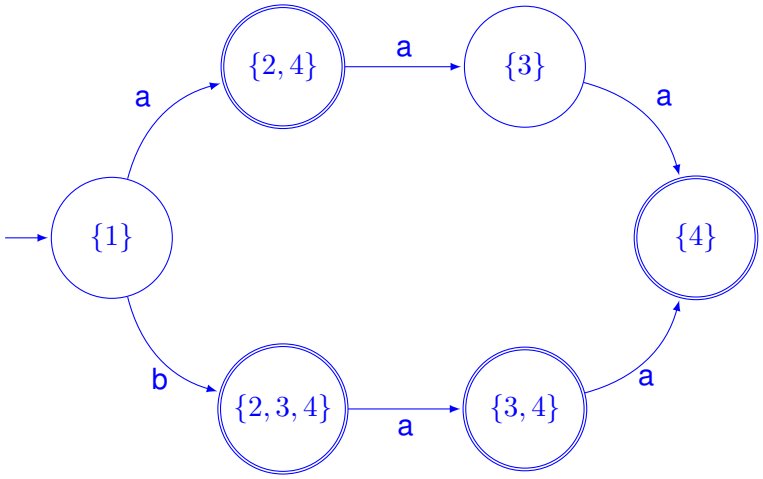
Punkte ____ von **6**

a) Geben Sie das Zustandsdiagramm eines endlichen Automaten für die Sprache an, die durch den regulären Ausdruck $(a|bb|\epsilon)^*c$ repräsentiert wird. Der Akzeptor kann nichtdeterministisch sein.



b) Geben Sie zu dem nachfolgenden nichtdeterministischen endlichen Automaten einen äquivalenten deterministischen endlichen Automaten (Zustandsdiagramm) an.





Aufgabe 3 (3+3+3 Punkte)

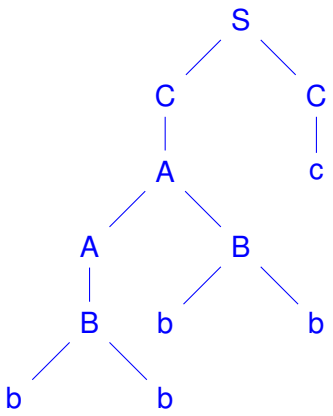
Punkte ____ von **9**

Gegeben sei folgende kontextfreie Grammatik G

$S \rightarrow AB \mid CC$
$A \rightarrow AB \mid B$
$B \rightarrow bb$
$C \rightarrow A \mid c$

a) Geben Sie eine Linksableitung und den Ableitungsbaum zu $bbbbc$ an.

$S \Rightarrow \underline{C}C \Rightarrow \underline{A}C \Rightarrow \underline{AB}C \Rightarrow \underline{B}Bc \Rightarrow bb\underline{B}C \Rightarrow bbbb\underline{C} \Rightarrow bbbbc$

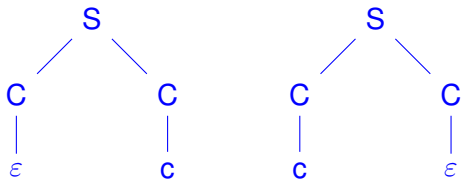


b) Geben Sie alle Wörter an, die sich zusätzlich ableiten lassen, wenn man die Regel $C \to \epsilon$ hinzufügt.

ε, bb, c

c) Beweisen Sie, dass die Grammatik durch die zusätzliche Regel $C \to \epsilon$ mehrdeutig wird.

Eine Grammatik G ist mehrdeutig, wenn es für ein Wort $w \in L(G)$ verschiedene Ableitungsbäume gibt. Für das Wort c ist dies der Fall.



Aufgabe 4 (2+2+2 Punkte)

Punkte ____ *von* **6**

Betrachten Sie folgende Grammatik

$$
\begin{aligned}
S &\to ABC \\
A &\to aabA \mid Cc \mid \epsilon \\
B &\to bcBd \mid \epsilon \\
C &\to dC \mid \epsilon
\end{aligned}
$$

a) Bestimmen Sie $FIRST(A)$: $\{a, d, c, \varepsilon\}$

b) Bestimmen Sie $FOLLOW(B)$: $\{d, \$\}$

c) Was steht in der LL(1)-Parsertabelle in dem Eintrag zu B und d ? $B \to \varepsilon$.

Aufgabe 5 (4+3+3 Punkte)

Punkte ____ *von* **10**

a) Bestimmen Sie zur nachfolgenden Grammatik die LR(0)-Elemente und die Übergänge im zugehörigen DEA.

b) Geben Sie die SLR(1)-Parsertabelle dazu an.

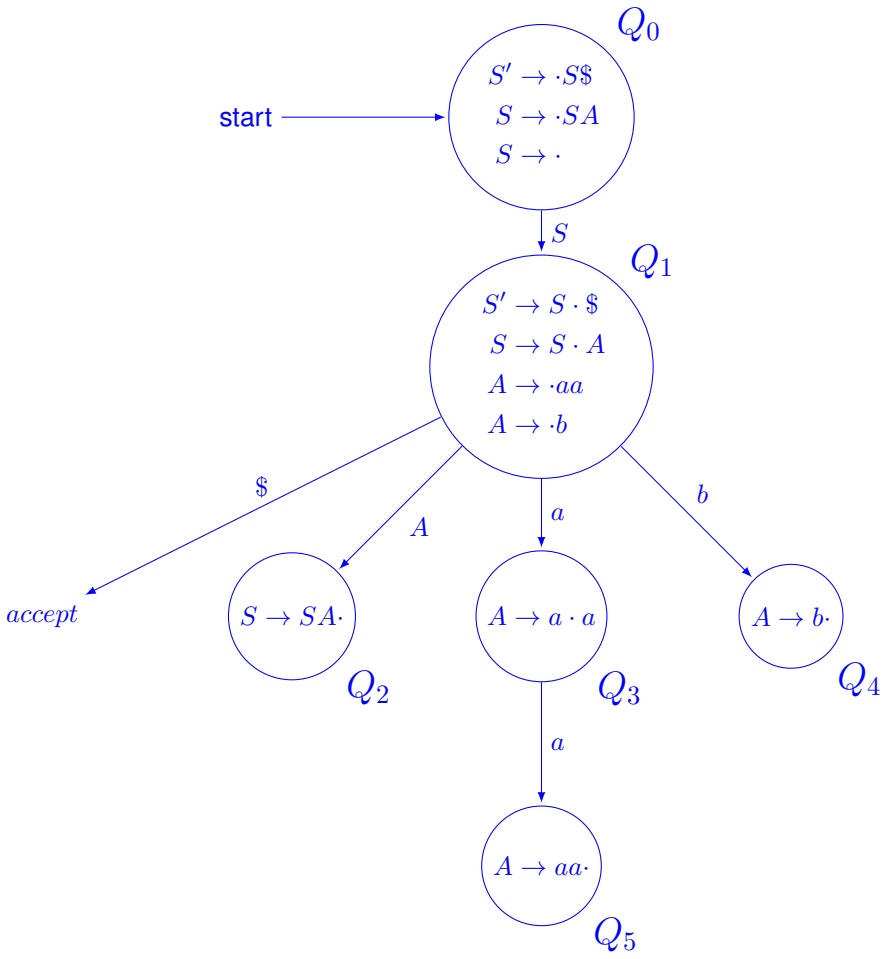
c) Geben Sie die Berechnung des SLR(1)-Parsers für die Eingabe b an.

(1) $S \to SA$
(2) $S \to \epsilon$
(3) $A \to aa$
(4) $A \to b$

Erweiterte Grammatik:

(0) $S' \to S\$$
(1) $S \to SA$
(2) $S \to \epsilon$
(3) $A \to aa$
(4) $A \to b$

DEA



Anmerkung:

Für die Bestimmung der Reduktionen in der Tabelle wird benötigt

$FOLLOW(A) = FOLLOW(S) = \{a, b, \$\}$

SLR(1)-Parsertabelle

State	Action			GOTO	
	a	b	$	S	A
Q_0	r2	r2	r2	Q_1	
Q_1	s3	s4	accept		Q_2
Q_2	r1	r1	r1		
Q_3	s5				
Q_4	r4	r4	r4		
Q_5	r3	r3	r3		

Aufgabe 6 (3 + 7 Punkte)

Punkte ____ *von* **10**

a) Bestimmen Sie zur SPL-Prozedur p das Frame-Layout für den Aktivierungsrahmen: Bestandteile in der richtigen Reihenfolge mit Offsets zum Framepointer und Größen in Bytes

b) Bestimmen Sie den ECO32-Assemblercode zu p. Die Prozedur *printi* erwartet einen Wertparameter vom Typ int. (SP=$29, FP=$25, RET=$31, verfügbare Register: $8-$15).

```
type paartyp = array [2] of int;

proc p (ref i:int, j:int) {
  var paar: paartyp;
  paar[j] := i;
  printi(i);
}
```

(a) Frame-Layout für p:

Adresse	Größe	Inhalt
FP-8	8 Byte	paar
FP-12	4 Byte	alter FP
FP-16	4 Byte	alte RETURN-Adresse
FP-20	4 Byte	Outgoing-Args: Platz für printi-Argument

(b) Assembler-Code für p

```
        .export p
p:
        sub     $29,$29,20              ; allocate frame
        stw     $25,$29,8               ; save old frame pointer
        add     $25,$29,20              ; setup new frame pointer
        stw     $31,$25,-16             ; save return register
        add     $8,$25,-8
        add     $9,$25,4
        ldw     $9,$9,0
        add     $10,$0,2
        bgeu    $9,$10,_indexError
        mul     $9,$9,4
        add     $8,$8,$9
        add     $9,$25,0
        ldw     $9,$9,0
        ldw     $9,$9,0
        stw     $9,$8,0
        add     $8,$25,0
        ldw     $8,$8,0
        ldw     $8,$8,0
        stw     $8,$29,0                ; store arg #0
        jal     printi
        ldw     $31,$25,-16             ; restore return register
        ldw     $25,$29,8               ; restore old frame pointer
        add     $29,$29,20              ; release frame
        jr      $31                     ; return
```