

Klausur zu „Grundlagen des Compilerbaus“, WS 2003/04

19. Februar 2004

Hinweise:

- **Bearbeitungszeit:** 2 Stunden
Gesamtpunktzahl: 90 Punkte
 Zum Bestehen der Klausur sind **36 Punkte** erforderlich.
- Hilfsmittel sind nicht erlaubt.
- Jede Aufgabe ist auf einem eigenen Blatt zu bearbeiten.
- Bis auf das Konzeptpapier sind alle Blätter zurückzugeben.

Viel Erfolg!

**Wiederholungsblatt mit
mündlichen Aufgaben**

Mat.-Nr.: Studienfach:

Aufgabe	max. Punktzahl	erreichte Punktzahl
1	10	
2	10	
3	10	
4	10	
5	10	
6	12	
7	11	
8	8	
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		

**Besprechung: ab
11.2.2006 in der Übung**

Aufgaben

Aufgabe 1: T-Diagramme

10 Punkte

Die Sprache C_p sei eine Erweiterung von C um Packages (Modulgruppen mit privaten und öffentlichen Methoden). Zur Implementierung steht Ihnen ein C-Compiler der Gnu-Compilersuite (zweistufig mit Zwischensprache RTL) in ausführbarer Form und als C-Programm zur Verfügung.

Beschreiben Sie mit Hilfe von T-Diagrammen *zwei verschiedene* Vorgehensweisen zur Erstellung eines ausführbaren Compilers, der C_p -Programme in Maschinencode M übersetzt. Nennen Sie jeweils Vor- und Nachteile jeder Vorgehensweise.

Aufgabe 2: DFAs und reguläre Ausdrücke

10 Punkte

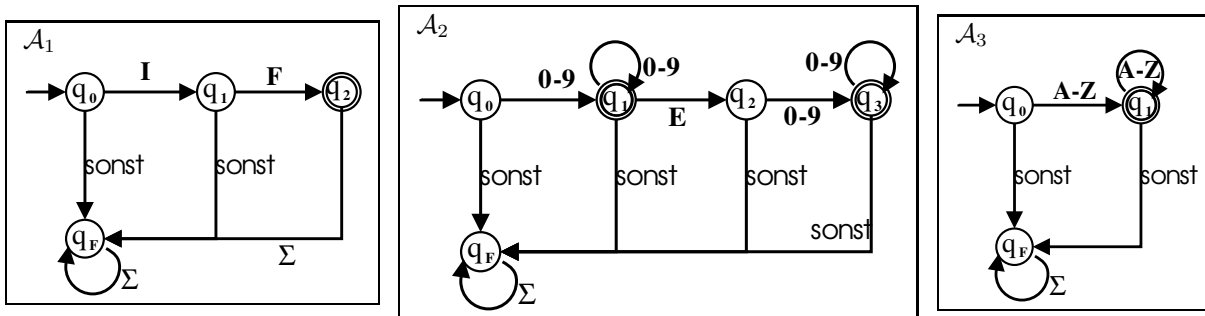
Eine Gleitpunktkonstante bestehe aus einem nicht-leeren ganzzahligen Teil, einem Dezimalpunkt, einem Dezimalbruch, dem Zeichen **e** oder **E**, einem ganzzahligen Exponenten mit optionalem Vorzeichen und einem optionalen Typ-Suffix, nämlich **f**, **F**, **l** oder **L**. Ganzzahliger Teil und Dezimalbruch seien Ziffernfolgen. Entweder der Dezimalbruch oder der Exponent beginnend mit **e**/**E** kann fehlen (aber nicht beide).

- Erstellen Sie (mit beliebiger Methode) einen DFA, der die beschriebenen Gleitpunktkonstanten erkennt. / 4
- Beschreiben Sie Gleitpunktkonstanten mit Hilfe von regulären Ausdrücken. Benutzen Sie dazu bei Bedarf übliche Abkürzungen für reguläre Ausdrücke und geben Sie Definitionen für diese an. / 6

Aufgabe 3: Produktautomat und lexikalische Analyse

10 Punkte

Gegeben seien die Automaten $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ zur Erkennung des Schlüsselworts **IF**, eines Zahlworts und eines Bezeichners. Die Automaten seien wie folgt bezeichnet: $\mathcal{A}_i = (Q^i, \Sigma, \delta^i, q_0^i, F^i)$ mit $\Sigma = \{A, \dots, Z, 0, \dots, 9\}$ und ihre Zustände gemäß der grafischen Darstellung nummeriert.



- Definieren Sie einen Produkt-DFA $\mathcal{A}$ mit $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2) \cup L(\mathcal{A}_3)$. / 2
- Geben Sie eine sinnvolle “first-Match”-Zerlegung der Endzustände von $\mathcal{A}$ an. / 3
- Charakterisieren Sie die produktiven Zustände von $\mathcal{A}$. / 2
- Welche Symbole werden bei Eingabe von **IFF1EIF** erkannt? / 3

Aufgabe 4: Grammatik für Zuweisungen

10 Punkte

Gegeben sei die folgende Grammatik $G = (\{L, S, D, E\}, \{\text{id}, \text{num}, :=, ;\}, L, P)$ mit den Produktionen:

$$\begin{aligned}P : L &\rightarrow S \\L &\rightarrow S ; L \\S &\rightarrow D := E \\D &\rightarrow \text{id} \\E &\rightarrow \text{id} \\E &\rightarrow \text{num}\end{aligned}$$

Die Grammatik beschreibt eine durch Semikolon getrennte Liste von Zuweisungen an Bezeichner (**id**).

- a) Ist $G \in LL(1)$? Begründen Sie Ihre Antwort. / 5
Falls nein, ändern Sie G zu einer äquivalenten $LL(1)$ -Grammatik ab.
- b) Bestimmen Sie die $LL(1)$ -Analysetabelle zu G . / 5

Aufgabe 5: Parserkombinatoren

10 Punkte

- a) Definieren Sie einen Parser-Kombinator

`repUntil :: Parser a c -> Parser a b -> Parser a ([b], c),`

welcher so lange den zweiten Parser (Elemente vom Typ **b**) anwendet, bis mit dem ersten Parser ein Element vom Typ **c** erkannt wird. Die Parameter sind also ein Parser für das abschließende Element und ein zweiter für die wiederholten Elemente. Der Parser soll eine Liste erkannter **b**-Elemente und das abschließende **c**-Element als Paar zurückliefern.

- b) Definieren Sie unter Verwendung der Kombinatoren `letter`, `word` und `tok` aus der Vorlesung einen Parser, welcher eine Deklaration der Form / 4

`<durch Komma getrennte Bezeichnerliste> , <Typname>`

erkennt. Dabei seien Bezeichner nicht-leere Folgen von Buchstaben, gültige Typnamen seien `int`, `float`, `bool`.

- c) Welche Ausgabe hat Ihr Parser aus Teil b) bei Eingabe von `"an,int,is,not,a,bool"`? / 2

Aufgabe 6: Java-Namensgrammatik

12 Punkte

In der Sprachdefinition von Java wird eine Grammatik mit den nebenstehenden Produktionen verwendet (die Nonterminale sind dabei **M**: **Method Name**, **P**: **Package Name**, **A**: **Ambiguous Name**). `id` steht für ein Terminal (Identifier), der Punkt ist ebenfalls Terminal. Als Startsymbol wurde für diese Aufgabe **S** hinzugefügt. Wie auch in der Beschreibung von Java betont wird, eignet sich diese Grammatik nicht zur Implementierung.

S	$\rightarrow$	M
		P . id
P	$\rightarrow$	id
		P . id
M	$\rightarrow$	id
		A . id
A	$\rightarrow$	id
		A . id

- a) Begründen Sie, dass es sich hierbei nicht um eine $LALR(1)$ -Grammatik handeln kann. / 2
- b) Bestimmen Sie eine $LR(1)$ -Menge, welche einen Konflikt aufweist. Um welche Art Konflikt handelt es sich? / 3
- c) Geben Sie eine $SLR(1)$ -Grammatik an, welche die gleiche Sprache erzeugt und erstellen Sie die zugehörige Analysetabelle. / 7

Aufgabe 7: Attributgrammatiken

11 Punkte

- a) Gegeben sei die Grammatik

/ 3

$$G = (\{S, B\}, \text{num}, B, \{S \rightarrow B, B \rightarrow BB, B \rightarrow \text{num}\})$$

Das Terminal **num** stehe für eine Ziffernfolge und besitze ein synthetisches Attribut $n \in \mathbb{N}$, welches den dargestellten Zahlwert angibt.

Erweitern Sie die Grammatik um Attribute, so dass *jedes* Terminal ein Attribut m mit dem Maximum der Zahlwerte n aller benutzten Terminale besitzt.

- b) Betrachten Sie erneut die Grammatik G aus Aufgabe 4 mit einer zusätzlichen Startregel $S' \rightarrow L$. / 8

Das Terminalsymbol **num** (Zahl) besitze ein Attribut *val* mit seinem Zahlwert aus $\mathbb{Z}$, **id** (Bezeichner) ein Attribut *name* mit dem Bezeichnernamen aus einer Menge **NAMES**. Beide Attribute werden von der lexikalischen Analyse geliefert.

Erweitern Sie G zu einer *L-Attributgrammatik*, welche die Speicherbelegung vor und nach jeder Zuweisung angibt. Der Speicher sei dabei durch eine Menge von Paaren $(name, val) \in \mathbf{NAMES} \times \mathbb{Z}$ dargestellt, die zu Anfang leer ist und jeden Bezeichner nur einmal aufführt. Geben Sie auch die Wertebereiche aller verwendeten Attribute an.

Aufgabe 8: Codegenerierung für PSA

8 Punkte

Gegeben sei das folgende PSA-Programm:

```
const X=3;
var Y,Z;
Y := 100;
Z := X;
while Z < Y do
  Y = (Y*X)/Z;
  Z = Z+1
```

Übersetzen Sie das Programm in Maschinencode für die abstrakte PSA-Maschine der Vorlesung.

Aufgabe 9: Maschinenzustände PSP

9 Punkte

- a) Gegeben sei der folgende Maschinenzustand der abstrakten MP-Maschine aus der Vorlesung. / 3

(35, 4 : 3 : 2, 8 : 3 : 10 : 7 : 4 : 3 : 27 : 9 : 4 : 3 : 54 : 1 : 5 : 4 : 28 : 13 : 19 : 14...)

Geben Sie den Folgezustand nach Bearbeitung des Befehls **LOAD(2,2)** an.

- b) Könnten die folgenden Keller-Inhalte bei der Ausführung eines übersetzten PSP-Programms als Prozedurkeller der MP-Maschine entstehen? Begründen Sie Ihre Antworten. / 6

- (a) 15:4:9:-3:5:5:6:3:2:1:5:4:10:3:4:4:3:25:0:0:0:0:12
(b) 15:4:9:-3:5:10:4:3:2:1:5:4:12:3:4:4:3:25:0:0:0:0:12
(c) 15:4:9:-3:5:14:4:3:2:1:5:4:12:3:4:4:3:25:0:0:0:0:12

Viel Erfolg!