Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik

Studiengruppe IF 4 B
Sommersemester 2014

Prüfung Compiler

Datum	:	22.07.2014, 10:30 Uhr
Bearbeitungszeit	:	90 Minuten
Prüfer	:	Prof. Dr. Oliver Braun
Hilfmittel	:	Keine
Erreichbare Punkte	:	90

Name: _____

Vorname: _____

Matrikelnummer: _____ Studiengruppe: _____

Hörsaal: _____ Platz Nr.: _____

Unterschrift: _____

Bitte kontrollieren Sie, ob Sie eine vollständige Angabe mit 5 Aufgaben auf 9 Seiten erhalten haben.

Aufgabe	1	2	3	4	5	Summe
max. Punkte	16	24	18	20	12	90

Anmerkungen:

- Schreiben Sie die Lösungen in die dafür vorgesehenen Kästchen. Sollte Ihnen der Platz dabei nicht reichen, benutzen Sie die Rückseite **und vermerken Sie das im dazugehörigen Kästchen!**

Aufgabe 1 (16 Punkte)

Gegeben sei folgender Code in einer Java-ähnlichen Programmiersprache:

```
do {
    if (x<12)
        break;
    else
        x = x + 1;
} while (true);
```

(a) Geben Sie den Token-Strom an, den ein Scanner hier erkennen müsste. Sie können die Token entweder als Haskell-Token oder als Tupel bestehend aus dem Token und der syntaktischen Kategorie angeben, also z.B. entweder If oder (if, Schlüsselwort). (8)

(b) Geben Sie den den Code repräsentierenden Syntax-**DAG** an. (8)

Aufgabe 2 (24 Punkte)

Geben Sie für die folgenden regulären Ausdrücke jeweils einen **deterministischen endlichen Automat**, der die gleichen Zeichenfolgen akzeptiert, als Zustandsübergangsdiagramm an. Sie brauchen die Zustände nicht benennen, es reicht wenn Sie kleine Kreise zeichnen. Sie brauchen den DFA nicht formal herleiten bzw. konstruieren.

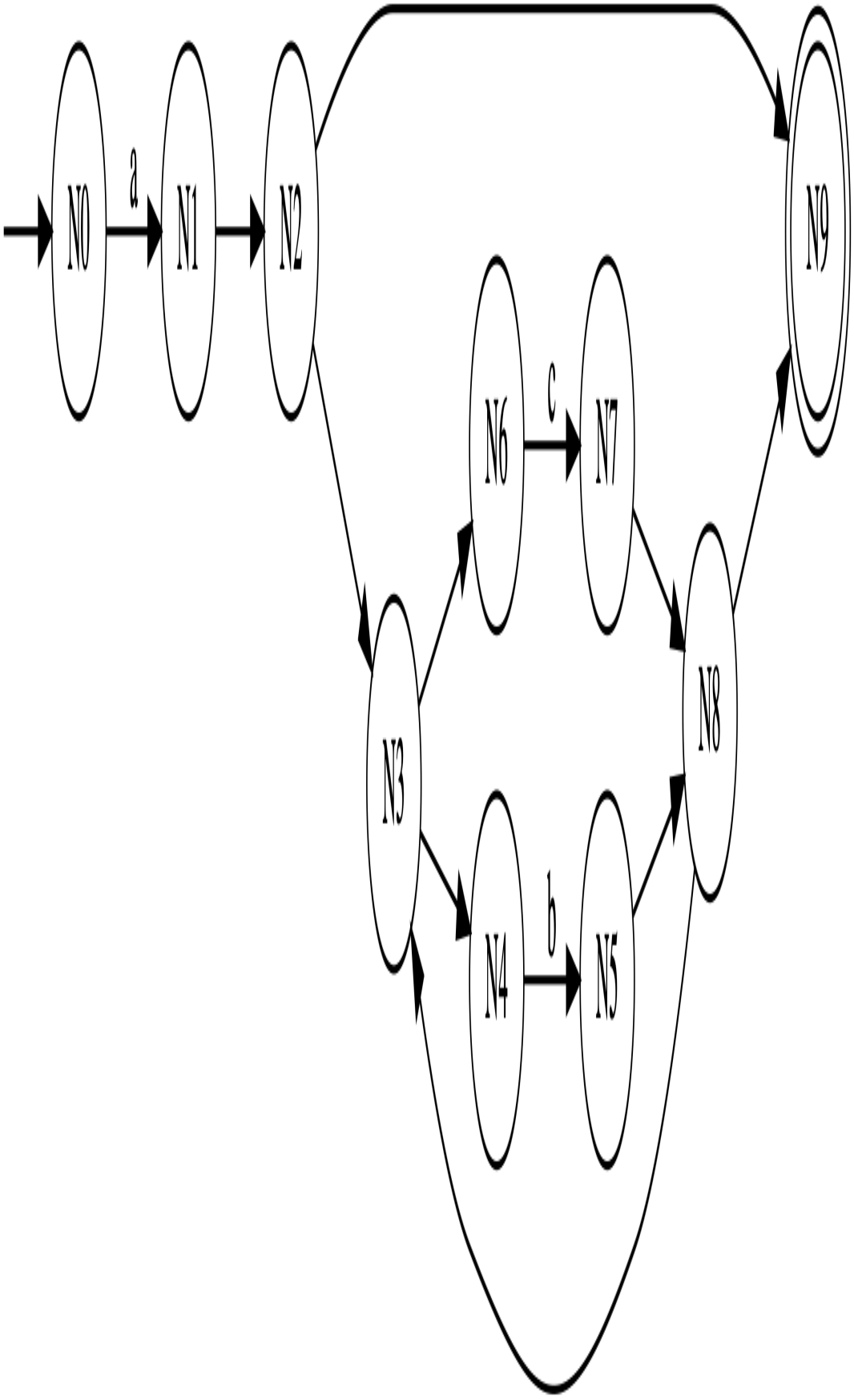
(a) $abb^*(ca|d|ea)c^*$ (6)

(b) $(a|ab|aab|b)c^*ca$ (8)

(c) $a(ba|bb^*c)^*$ (10)

Aufgabe 3 (18 Punkte)

Gegeben sei folgender NFA:


N0
a
N1
N2
N3
N4
b
N5
N6
c
N7
N8
N9

Erzeugen Sie mit Hilfe der Teilmengenkonstruktion (*subset construction*) daraus einen DFA.

Geben Sie alle ermittelten Teilmengen **und** den resultierenden DFA an.

Aufgabe 4 (20 Punkte)

Gegeben sei der Token-Strom

```
[ T_Star, T_Star, T_Star, T_Text "Hallo Welt!", T_Star, T_Star, T_Star ]
```

der aus dem Markdown `***Hallo Welt!***` vom Scanner erkannt worden ist. Dabei bedeutet Text der in einfachen Sternchen eingefasst wurde *kursiv* und Text in doppelten Sternchen *fett*. Der Benutzer Ihres Compilers würde also erwarten `Hallo Welt!` wird kursiv **und** fett dargestellt. Der Parser-Code der dafür zuständig ist, sieht so aus:

```
1  parse (T_Star:rest) =
2    let (iCont, (T_Star:rest')) = span (/=T_Star) rest
3    in case parse iCont of
4         Nothing -> Nothing
5         Just contAst -> case parse rest' of
6           Nothing -> Nothing
7           Just restAst -> Just $ Sequence [Italic contAst, restAst]
8  parse (T_Star:T_Star:rest) =
9    let (bCont, rest') = getContent rest
10        getContent (T_Star:T_Star:rest'') = ([],rest'')
11        getContent (t:ts) = let (c,r) = getContent ts in (t:c,r)
12   in case parse bCont of
13        Nothing -> Nothing
14        Just contAst -> case parse rest' of
15          Nothing -> Nothing
16          Just restAst -> Just $ Sequence [Bold contAst, restAst]
```

(a) Geben Sie den Teil des `AST`-Datentyps an, der in dem obigen Code benötigt wird. (`data AST = ...`) (5)

(b) Erklären Sie **zeilenweise** was in den Zeilen 2-7 berechnet wird. (5)

(c) Erklären Sie **zeilenweise** was in den Zeilen 9-16 berechnet wird. (5)

(d) Erklären Sie warum der Parser mit dem gezeigten Code nicht das gewünschte Ergebnis erzielt. Was wird der Parser stattdessen berechnen, wenn er zusätzlich noch folgende Funktionsgleichungen enthält: (5)

```
parse (T_Text text : rest) =
    maybe Nothing (\(Sequence ast) -> Sequence (Text text : ast))
        $ parse rest
parse "" = Just $ Sequence []
```

Geben Sie das Ergebnis als Haskell-Wert an.

Aufgabe 5 (12 Punkte)

Gegeben sei folgender Java-ähnlicher Quellcode:

```
1  int x = 12;
2  int y = 7;
3  while (y<12) {
4      int a = 10;
5      while (a>0) a--;
6      if (y!=x)
7          a = x-- + --y;
8      else
9          a = 5
10     y--;
11 }
```

(a) Geben Sie den Kontrollflußgraph (*control-flow graph*) für obigen Code an. (6)

(b) Geben Sie den Abhängigkeitsgraph (*dependence graph*) für obigen Code an. (6)