Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

3. Februar 2017

Aufgaben zur Klausur **Compilerbau** im WS 2016/17 (BInf, BMInf, BTInf)

Zeit: 75 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Nutzen Sie die Rückseiten der Klausur zur Entwicklung der Lösungen und übertragen die fertigen Lösungen in das Aufgabenblatt.

Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg!

Diese Klausur besteht einschließlich dieses Deckblattes aus 7 Seiten.

Aufgabe 1:

Gegeben sei die folgende kontextfreie Grammatik **G=(T,N,P,S)** mit

T = { id, =, *, /, ==, !=, (,), [,] }

N = { A, B, C, D, E }

S = A

und den Produktionen **P**:

1. A ::= E = B
2. A ::= B
3. B ::= C != C
4. B ::= C == C
5. B ::= C
6. C ::= D / D
7. C ::= D * D
8. C ::= D
9. D ::= D [C]
10. D ::= (B)
11. D ::= id
12. E ::= E [C]
13. E ::= (E)
14. E ::= id

Konstruieren Sie die FIRST–Mengen für die Nichtterminalsymbole.

FIRST(A) = ...

FIRST(B) = ...

FIRST(C) = ...

FIRST(D) = ...

FIRST(E) = ...

Warum ist die Berechnung der FOLLOW–Mengen für die Konstruktion eines LL-Parsers für diese Grammatik redundant?

. . . . . .

. . . . . .

Diese Grammatik ist wegen der Linksrekursionen keine LL(1)–Grammatik.

Eliminieren Sie die Linksrekursionen für das Nichtterminalsymbol D:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Die Elimination für E läuft analog. Die resultierende Grammatik ist immer noch nicht LL(1). Es ist eine Linksfaktorisierung notwendig.

Faktorisieren Sie die Regeln für B:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Die Faktorisierung für C läuft analog. Die resultierende Grammatik ist immer noch keine LL(1)–Grammatik. Geben Sie für mindestens eine Stelle in der LL-Parser–Tabelle die mehrfach anwendbaren Regeln an.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Ist die Grammatik mehrdeutig?

ja ☐ nein ☐

Begründung:

. . . . . .

. . . . . .

Aufgabe 2:

Transformieren sie den regulären Ausdruck $((yz|y*)x?)+$ **gemäß des Transformationsschemas aus der Vorlesung** in einen nichtdeterministischen endlichen Automaten. Das zu Grunde liegende Alphabet sei dabei $A = \{x, y, z\}$.

Die Transformaton erfolgt schrittweise. Nutzen Sie für die Zwischenschritte die Rückseiten der Klausur.

Der fertige Automat als Zustandsübergangsdiagramm:

Konstruieren Sie den minimalen deterministischen Automaten für den obigen Ausdruck als Zustandsübergangsdiagramm:

Aufgabe 3:

Skizzieren Sie, welcher (Assembler–)Code für einen von–Neumann–Rechner für das folgende Programmfragment sinnvollerweise erzeugt wird:

while (i1 = i2) **or** (b **and not** c)
do
  a, b := b, f(a, a+b)
endwhile

Nutzen Sie in den Instruktionen die Namen der Variablen als symbolische Adressen für die zugehörigen Speicheradressen. Verwenden sie den Instruktionssatz der in der Vorlesung behandelten virtuellen Maschine. Die Operatoren **and** und **or** sollen nicht strikt ausgewertet werden.

Aufgabe 4:

Gegeben sei eine beliebige kontextfreie Grammatik **G=(T,N,P,S)**

1. Definieren Sie die zugehörige Sprache **L(G)**

   . . . . . .

   . . . . . .

2. Definieren Sie den Begriff **mehrdeutige kontextfreie Grammatik**

   . . . . . .

   . . . . . .

3. Definieren Sie den Begriff **mehrdeutige kontextfreie Sprache**

   . . . . . .

   . . . . . .

4. Aus welchen Gründen werden Programmiersprachen möglichst nicht mit Hilfe mehrdeutiger Grammatiken definiert?

   . . . . . .

   . . . . . .

   . . . . . .

   . . . . . .