Prof. Dr. Uwe Schmidt

fhwedel
UNIVERSITY OF APPLIED SCIENCES

25. August 2009

Aufgaben zur Klausur **Compilerbau** im SS 2009 (BInf 251, BMInf 252, II h769)

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.

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 6 Seiten.

Aufgabe 1:

Transformieren sie den regulären Ausdruck $(a?(b|c+))*$ **gemäß dem Transformationsschema aus der Vorlesung** in einen nichtdeterministischen endlichen Automaten. Das zu Grunde liegende Alphabet sei dabei $I = \{a, b, c\}$. Hinweis: $*$ und $+$ binden stärker als der $|$.

Der Automat als Zustandsübergangsdiagramm:

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

Aufgabe 2:

Entwickeln Sie den Assembler–Code für eine Kellermaschine, wie sie in der Vorlesung vorgestellt worden ist, der von einem nicht optimierenden Übersetzer für das folgende Programmfragment erzeugt wird. b sei dabei eine Boolesche Variable, x und y ganzzahlige Variablen.

if (**not** (x = y) **and** y > 0) **or not** b
then
  x, b := x * y, **true**
else
  x, y := y, 0 - x
endif

Verwenden Sie in den Instruktionen die Namen der Variablen als symbolische Adressen für die zugehörigen Speicheradressen. Die Bedingungen sollen nicht strikt ausgewertet werden. Nutzen Sie den vorgegebenen Platz, wenn Sie mehr als 12 Instruktionen benötigen, indem Sie den Code zweispaltig anordnen.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 3:

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

T = { id, :=, +, -, eq, ne, (,), [,] }

N = { S, R, A, P, L }

S = S

und den Produktionen **P**:

1. S ::= R
2. S ::= L := R
3. R ::= A
4. R ::= A eq A
5. R ::= A ne A
6. A ::= P
7. A ::= P + P
8. A ::= P - P
9. P ::= id
10. P ::= (R)
11. P ::= P [A]
12. L ::= id
13. L ::= (L)
14. L ::= L [A]

Konstruieren Sie die FIRST–Mengen für die Nichtterminalsymbole.

FIRST(S) = ...

FIRST(R) = ...

FIRST(A) = ...

FIRST(P) = ...

FIRST(L) = ...

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 P:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

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

Faktorisieren Sie die Regeln für R:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Die Faktorisierung für A 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:

. . . . . .

. . . . . .
