Compilerbau (BSc Informatik; SS12)

Dipl. Inf. M. Förster, Dipl.-Ing. J. Lotz und Prof. Dr. U. Naumann

LuFG Informatik 12: Software and Tools for Computational Engineering
RWTH Aachen

Übungsblatt zum Algorithmischen Differenzieren

Abgabe: 09.07.2012 bis 10:00 Uhr

1 Tangenten-Linearer Ableitungscode per Hand (10 Punkte)

1. Schreiben Sie für folgende SL^2 Programme tangenten-linearen Ableitungscode per Hand:

   a)

   ```
   y=sin(sin(cos(x)/2)/x)*x;
   ```

   b)

   ```
   y=x+a+2*sin(y);
   z=cos(y*z);
   ```

   c)

   ```
   x=exp(x)-2;
   x=5;
   ```

2. Schreiben Sie entsprechende Treiberprogramme zur Berechnung der partiellen Ableitungen aller Ausgaben bezüglich aller Eingaben (Gradient bzw. Jacobimatrix).

3. Schreiben Sie entsprechende Treiberprogramme zur Verifikation der berechneten Gradienten bzw. Jacobimatrizen mittels finiter Differenzenquotienten.

2 Parser (5 Punkte)

Betrachten Sie numerische Programme, welche syntaktisch durch eine Grammatik mit folgenden Produktionsregeln beschrieben sind:

```
p : s
  ;
s : a
  | a s
  ;
a : V '=' e ';'
  ;
e : ADD '(' e ',' e ')'
  | MUL '(' e ',' e ')'
  | V
  | C
  ;
```

Startsymbol ist p, Die Terminalsymbole stehen für Variablen (V), Konstanten (C), binäre Gleitkommaadditionen (ADD) und -multiplikationen (MUL).

1. Zeichnen Sie den LR(0)-Automaten dieser Grammatik.
2. (optional) Implementieren Sie einen entsprechenden SLR-Parser mit `flex` und `bison`.

3 Single-Pass Ableitungscodecompiler (5 Punkte)

1. Erweitern Sie die Grammatik aus Aufgabe 2 so zu einer Attributgrammatik, dass
   (a) auf Zuweisungsebene *single assignment code* generiert wird;
   (b) tangenten-linearer Code generiert wird.
2. (optional) Stellen Sie sicher, dass der Compiler rekursiv auf seine eigenen Ausgaben angewandt werden kann. Ist somit die Generierung von korrektem Ableitungscode höherer Ordnung möglich?