Prof. Dr. Gerhard Reinelt
Dipl.-Math. Hanna Seitz
Institut für Informatik
Universität Heidelberg

Compilerbau

10. Übungsblatt SS 08
Abgabetermin: 24.06.08

Aufgabe 36

Zeigen Sie, dass sich in Aufgabe 33 die Konflikte durch $SLR(1)$-Analyse auflösen lassen. Berechnen Sie dazu den $SLR(1)$-Automaten und die Aktion-Tabelle.

Aufgabe 37 (6 Punkte)

Gegeben sei folgende mehrdeutige Grammatik $G_n = (N, T_n, S, P_n)$ für arithmetische Ausdrücke mit n binären Operatoren $\theta_1, \ldots, \theta_n$:

$$N = \{S, E\}, \quad T_n = \{\ \texttt{id},\ (,\ ),\ \theta_1,\ \ldots,\ \theta_n,\ \texttt{\$}\},$$

mit den Produktionen P_n

$$S \to E\ \texttt{\$}$$
$$E \to E\theta_1 E \ |\ E\theta_2 E\ | \ldots |\ E\theta_n E\ |\ \texttt{(}\ E\ \texttt{)}\ |\ \texttt{id}$$

Es gelte: θ_i hat höhere Priorität als $\theta_j \Leftrightarrow i > j$.

- **a)** Konstruieren Sie den LR(0)-Automaten $dchar_0(G_n)$. Geben Sie die Anzahl der Zustände als Funktion von n an.
- **b)** Stellen Sie die SLR(1)-Action-Tabelle auf. Die sich ergebenden Konflikte sollen wie folgt aufgelöst werden:

  Aktionen mit θ_i haben Vorrang vor Aktionen mit θ_j, wenn $i > j$. Bei gleichem Operator soll vorrangig reduziert werden, d.h. die Operatoren sollen linksassoziativ sein.

- **c)** Parsen Sie die drei Worte „$\texttt{id}\,\theta_i\,\texttt{id}\,\theta_i\,\texttt{id\$}$“ ($i$ beliebig) und „$\texttt{id}\,\theta_i\,\texttt{id}\,\theta_j\,\texttt{id\$}$“ für $i < j$ und für $i > j$.

Aufgabe 38

Gegeben sei die folgende Grammatik G:

$$S \to E\ \texttt{\$}$$
$$E \to \texttt{id}\ |\ \texttt{id}\ (\ E\ )\ |\ E + \texttt{id}.$$

Konstruieren Sie den LR(0) Automaten $dchar_0(G)$. Ist die Grammatik LR(0)? Ist die Grammatik SLR(1)? Begründen Sie Ihre Antworten.

Aufgabe 39

Gegeben sei die Grammatik G zur Darstellung reeller Zahlen im Exponentialformat durch

$$S \to A.B\ \texttt{e}\ A$$
$$A \to V B$$
$$V \to \texttt{+}\ |\ \texttt{-}$$
$$B \to B\ \texttt{ziffer}\ |\ \texttt{ziffer}$$

Es wird angenommen, dass **ziffer** als Ergebnis der lexikalischen Analyse entsteht und in dem Attribut **ziffer**.lexval der ermittelte Ziffernwert zu finden ist. Geben Sie eine syntaxgesteuerte Definition an, die den Zahlenwert von S im Fließkommaformat bestimmt.

Zusatzaufgabe (12 Punkte, Abgabe dieser Aufgabe bis 08.07.2008)

Implementieren Sie den Algorithmus von Cocke, Young und Kasami (s. im Skript Seite 51). Wenn das Eingabewort akzeptiert wird geben Sie zusätzlich die Ableitung aus. Grundsätzlich soll die Ausgabe in eine Datei erfolgen.

Der Quellcode muss in einer (!) Datei cyk.c vorliegen. Der Programmaufruf erfolgt durch

```
``./cyk <Eingabedatei> Eingabewort'',
```

wobei cyk die durch die Kompilierung erzeugte ausführbare Datei ist. <Eingabedatei> ist der Dateiname der einzulesenden Grammatik-Datei, das Eingabewort ein String.

Das Format der Eingabedatei ist folgendermaßen. In der ersten Zeile ist die Anzahl der Regeln angegeben. Dann folgt eine Regel pro Zeile. Der Übergang von der linken zur rechten Seite der Regel ist mit "–" gekenntzeichnet. Terminalsymbole werden mit Kleinbuchstaben, Nichtterminalsymbole mit Großbuchstaben bezeichnet. Das Startsymbol ist "S". Beachten (und überprüfen) Sie, dass es sich bei der Eingabe um eine Grammatik in Chomsky Normalform handeln muss. Eine Beispielgrammatik lautet folgendermaßen.

$$
\begin{array}{l}
6 \\
S - \varepsilon \\
S - DB \\
D - AT \\
T - DB \\
A - a \\
B - b
\end{array}
$$

Für den Fall, dass es die Regel $S \rightarrow \varepsilon$ gibt, testen Sie ob S auf der rechten Seite einer Regel vorkommt. Wenn ja geben Sie zurück, dass diese Implementierung des CYK Algorithmus das Wortproblem für die angegebene Grammatik nicht löst.

Rahmenbedingungen:

- Nur ANSI C ist erlaubt.
- Das Programm muss mit dem gcc-Kompiler unter Linux kompilierbar sein.
- Ein Drittel der Punkte werden für guten Programmierstil vergeben, d. h. für Programmaufbau, Namensgebung sowie ausreichend viele und gute Kommentare.
- Verwenden Sie eine sinnvolle Datenstruktur.