Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Informatik und Prakt. Mathematik
Prof. Dr. M. Hanus, Frank Huch

24118 Kiel
Christan-Albrechts-Platz 4
Tel.: 0431 / 880-7277

6. Übung „Übersetzerkonstruktion“

Abgabe am 26. November in der Vorlesung

Aufgabe 19 4 Punkte

Die kontextfreie Grammatik G sei gegeben durch:

$$\begin{array}{rcl} S' & \to & S\$ \\ S & \to & AS \mid b \\ A & \to & SA \mid a \end{array}$$

a) Konstruieren Sie (analog zur Vorlesung) einen erfolgreichen Lauf des shift-reduce-Parsers für die Eingabe *abab*.

b) Berechnen Sie den LR(0)-Automaten zu G. Ist $G \in LR(0)$?

Aufgabe 20 7 Punkte

Die Grammatik G_{bool} sei gegeben durch

$$\begin{array}{rcl} S & \to & B\$ \\ B & \to & (B \wedge B) \mid \neg B \mid t \mid f \end{array}$$

a) Berechnen Sie die LR(0)-Automaten von G_{bool}.

b) Geben Sie die LR(0)–Analysetabelle von G_{bool} an.

c) Bestimmen Sie die Konfigurationsfolge, die der LR(0)–Analyseautomat bei Eingabe des Wortes $((\neg t \wedge f) \wedge (f \wedge t))$ durchläuft.

Aufgabe 21 9 Punkte

In dieser Aufgabe sollen Sie einen shift-reduce-Parser implementieren. Gehen Sie dazu in folgenden Schritten vor:

a) Formalisieren Sie hierzu zunächst, wie und wann der shift-reduce-Parser einen shift- bzw. einen reduce-Schritt durchführt.
Repräsentieren Sie den Keller als ein Wort über Zuständen ($\mathbb{N}^*$). In der Vorlesung wurden auch noch (Nicht-)Terminalsymbole auf den Keller geschrieben. Diese sind für das Arbeiten des Parsers aber überflüssig und können wegfallen.

Um völlig unabhängig von der Grammatik zu sein erweitern Sie die reduce-Einträge um die Länge der rechten Seite der anzuwendenden Regel und das Nichtterminalsymbol, zu welchem reduziert wird.
Formalisieren Sie auch, in welcher Konfiguration der Parser erfolgreich terminiert.

b) Erweitern Sie den Parser um ein Ausgabeband, auf welchem die Nummer der Verwendeten Regel beim reduce-Schritt ausgegeben wird. Liefert Ihr Parser die Rechtsableitung als Ausgabe?

c) Definieren Sie eine polymorphe Datenstruktur `ParsingTabelle a b` zur Darstellung der Parsingtabelle. Die Parsingtabelle soll polymorph bezüglich der gewählten Alphabete sein. `a` repräsentiert die Terminal- und `b` die Nichtterminalsymbole. Da die Parsingtabelle in der Regel nur eine partielle Funktion repräsentiert, also Lücken aufweist, sollten Sie den Haskell-Datentyp `Maybe` verwenden.

d) Definieren Sie eine Funktion `parse :: ParsingTabelle a b -> [a] -> [Int]`. Als Eingabe erhält dieser Parser eine Parsingtabelle und eine Tokenfolge. Als Ausgabe soll der Parser im Erfolgsfall die Ausgabe des Automaten liefern.