P. Thiemann, S. Wehr Wintersemester 2006/2007

Compilerbau

`http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2006ws/`

Übungsblatt 4

22.11.2006

Aufgabe 1 (Grammatik für reguläre Ausdrücke; 4 Punkte)
Betrachten sie folgende Grammatik für reguläre Ausdrücke. (Der reguläre Ausdruck $R_1 \cup R_2$ stellt dabei die Alternative zwischen R_1 und R_2 dar.)

$$R \to R \cup R \mid RR \mid R^* \mid (R) \mid a \mid b \mid c$$

Die Terminalsymbolen seien $\cup$, $*$, (,), a, b sowie c. Das Start- und einzige Nichtterminalsymbol sei R.

(a) Geben sie jeweils zwei verschiedene Linksableitungen für die Ausdrücke $a \cup b^*$ und abc an. Was ist das Problem?

(b) Schreiben sie obige Grammatik so um, dass das Problem nicht mehr auftritt. Dabei soll $*$ am stärksten und $\cup$ am schwächsten binden; ferner sollen die Operatoren alle linksassoziativ sein.

(c) Entfernen sie, falls nötig, Linkrekursion aus der in (b) erstellten Grammatik.

Aufgabe 2 (LL(1) Lookahead; 4 Punkte)
Betrachten sie folgende Grammatik mit den Nichtterminalsymbolen S, B, D, E, F und den Terminalsymbolen u, v, w, x, y, z sowie dem Startsymbol S.

$$\begin{aligned} S &\to uBDz \\ B &\to Bv \mid w \\ D &\to EF \\ E &\to y \mid \varepsilon \\ F &\to x \mid \varepsilon \end{aligned}$$

Berechnen sie den LL(1)–Lookahead für alle Regeln der Grammatik. Ist die Grammatik stark–LL(1)? Falls nicht: Erstellen sie durch möglichst geringe Modifikation der Ausgangsgrammatik eine Grammatik, die stark–LL(1) ist und dieselbe Sprache erkennt.

Aufgabe 3 (Implementierung eines Topdown Parsers; 6 Punkte)

(a) Erweitern sie die aus der Vorlesung bekannte Grammatik für arithmetische Ausdrücke um Variablen und Variablenbindungen (siehe auch Übungsblätter 1 und 3).

(b) Auf der Vorlesungsseite steht eine Archivdatei (`arith.tar.gz`) zur Verfügung, welche den Lexer und die Evaluierungsfunktion für arithmetische Ausdrücke aus den vorhergehenden Übungsblättern enthält. Implementieren sie einen Topdown Parser gemäß der in (a) erstellten Grammatik. Der Code des Parsers soll in der Datei `parser.ml` gespeichert werden und der in `parser.mli` gegebenen Signatur entsprechen. Mittels eines Aufrufs von `make` können sie das komplette Programm kompilieren. Im Modul `driver.ml` sollten sie eigene Testfälle definieren. Abzugeben ist lediglich die Datei `parser.ml`.

Abgabe: 29.11.2006
Die Abgabe erfolgt bis zu Beginn der Übungsstunde. Einreichungen, die nicht den Abgabemodalitäten entsprechen, werden abgelehnt. Für Plagiate werden keine Punkte vergeben.

Abgabemodalitäten:

- Code muss per Email an die Adresse `wehr@informatik.uni-freiburg.de` geschickt werden, die Lösungen zu den restlichen Aufgaben sind auf Papier abzugeben.
- Der Code muss in einem Archiv mit dem Namen `vorname_nachname.tar.gz` oder `vorname_nachname.zip` an die Email angehängt werden.
- Entpacken des Archivs muss ein einzelnes Verzeichnis mit dem Namen `vorname_nachname` liefern.
- Innerhalb dieses Verzeichnisses müssen sich folgende Dateien befinden: `parser.ml` (für Aufgabe 3).
- Alle Dateien müssen compilierbar sein; Teile einer Datei, welche nicht vom Compiler akzeptiert werden, müssen auskommentiert sein. Abgaben, die der Compiler nicht akzeptiert, werden nicht bewertet.
- Wenn eine Aufgabe von der Lösung ein gewisses Format verlangt (wie etwa einen festen Namen oder Typ für eine Funktion, eine vorgegebene Signatur für ein Modul etc.), so ist dieses Format zwingend einzuhalten.