P. Thiemann, M. Neubauer Wintersemester 2004/2005

Compilerbau
`http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2004/`

Übungsblatt 5

Abgabe: 1.12.2004

Aufgabe 1 (Parser für reguläre Ausdrücke):

Schreibe einen Yacc-Parser[1] der reguläre Ausdrücke erkennt. Ausgabe des Parsers sollen die einzelnen Schritte der entsprechende Rechtsableitung sein.

Betrachte dazu reguläre Ausdrücke der Grammatik $G = (\{R\}, \{\mathtt{a}, \mathtt{b}. \mathtt{|}, \mathtt{*}, \mathtt{(}, \mathtt{)}\}, P, R)$ mit Produktionen P

$$R \quad \to R\,\mathtt{|}\,R \mid RR \mid R\mathtt{*} \mid \mathtt{(}R\mathtt{)} \mid \mathtt{a} \mid \mathtt{b}$$

Aufgabe 2 (Parser für JavaScript):

Auf der Vorlesungswebseite ist eine unvollständige Ocamlyacc-Spezifikation eines JavaScript-Parsers[2] zu finden. Die Grammatik in ihrem jetztigen Zustand hat 5 *shift/reduce*- und 7 *reduce/reduce*-Konflikte. Passe die Grammatik-Beschreibung so an, dass der erzeugt Parser korrekt funktioniert.

Der Parsergenerator `ocamlyacc` erzeugt mit Hilfe der Option `-v` eine Beschreibung der Parser-Tabelle inklusive aller auftretenden Konflikte. Folgende Teile der Grammatik müssen hierzu näher untersucht und gegebenenfalls bearbeitet werden:

- Innerhalb von Array-Literalen können zwei Mal hintereinander optional sogenannte Auslassungen (*elision*) auftreten, was zu zwei *shift/reduce*-Konflikten führt.
- In *expression-statements* dürfen im Lookahead korrekterweise weder {- noch `function`-Tokens auftreten. Die bisherige Parser-Spezifikation beachtet dies nicht und enthält deshalb 2 *shift/reduce*- und 7 *reduce/reduce*-Konflikte. Entferne diese durch passende Duplikation und anpassung der *expression*-Regeln.
- Ein weiter *shift/reduce*-Konflikt kann bestehen bleiben, da dieser von `Ocamlyacc` richtig behandelt wird. Begründe weshalb `Ocamlyacc` diesen Konflikt korrekt auflöst, indem Du die passenden Abschnitte im EcmaScript-Standard zitierst und dies mit dem Verhalten des erzeugten Parsers vergleichst.

[1] Anstelle von Yacc kann auch ein entsprechendes Tool für eine andere Programmierpsrache wie etwa Ocamlyacc, JavaCC oder CUP benutzt werden.

[2] `http://www.ecma-international.org/publications/standards/Ecma-262.htm`