Prof. Dr. R. Loogen, M. Dieterle
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
D-35032 Marburg

25. November 2009

Philipps Universität Marburg

Übungen zu „Grundlagen des Compilerbau“, Winter 2009/10

Nr. 6, Abgabe der Aufgaben: 2. Dezember 2009 vor der Vorlesung

---

Aufgaben

6.1 Top-Down-Parser für eine While-Sprache II

6 Punkte

Die Grammatik in Abb. 1 erweitert die While-Sprache von Blatt 5 um Ausdrücke und eignet sich für Top-Down-Parsing.

Aufgabe: Programmieren Sie mit Parsec einen *Recursive-Descent*-Parser für die erweiterte While-Sprache. Benutzen Sie nicht die vordefinierte Parsergenerierungsfunktion `buildExpressionParser`. Auf der Webseite der Vorlesung finden Sie eine Umsetzung von Aufgabe 5.3 für Parsec, diese können Sie anpassen und erweitern. Des weiteren sind die Haskell-Datei `ProgType.hs` mit Datentypdefinitionen, zwei Testeingaben sowie ein passendes Scannermodul auf der VL-Seite bereitgestellt. Ihr Parser soll die Ausgabe des Scanners[1] (`[Token]`) einlesen und einen abstrakten Syntaxbaum für die Eingabe aufbauen.

program	→	**program** Id '{' *dec stmts* '}'
dec	→	**var** *decls*
decls	→	*varlist* ':' *type A*
type	→	**int**
	|	**bool**
A	→	';' *decls*
	|	ε
varlist	→	Id *B*
B	→	',' Id *B*
	|	ε
stmt	→	Id ':=' *expr*
	|	**print** *expr*
	|	**if** *expr* **then** '{'*stmts*'}' *maybeElse*
	|	**while** *expr* **do** *stmt*
	|	'{' *stmts* '}'
maybeElse	→	**else** '{' *stmts* '}'
	|	ε
stmts	→	*stmt C*
C	→	';' *stmts*
	|	ε

expr	→	*T expr'*
expr'	→	'+' *T expr'*
	|	'-' *T expr'*
	|	'|' *T expr'*
	|	ε
T	→	*F T'*
T'	→	'*' *F T'*
	|	'/' *F T'*
	|	'&' *F T'*
	|	ε
F	→	'{' *expr* RelOp *expr* '}'
	|	'(' *expr* ')'
	|	Id
	|	BoolVal
	|	NumVal

Bezeichner (Id), Werte (*Val) und Operatoren (*Op) seien wie auf Blatt 3 beschrieben. Ausgehend von den Symbolklassen von Blatt 3 ist die Sprache um die booleschen Operatoren '&' und '|' erweitert.

Abbildung 1: Grammatik für eine While-Sprache

[1] eine modifizierte Version des Scanners aus Aufgabe 3.2

Bitte wenden!

6.2 **Grammatik für Boolesche Ausdrücke** 6 Punkte

Die Grammatik G_{bool} sei gegeben durch:

$$\begin{array}{lcl} \mathrm{S} & \rightarrow & \mathrm{B} \\ \mathrm{B} & \rightarrow & (\mathrm{B} \wedge \mathrm{B}) \mid \neg \mathrm{B} \mid \mathrm{t} \mid \mathrm{f} \end{array}$$

Wobei $\Sigma = \{), (, \wedge, \neg, \mathrm{t}, \mathrm{f}\}$ und $N = \{\mathrm{S}, \mathrm{B}\}$

(a) Berechnen Sie die LR(0)-Informationen von G_{bool} mit Hilfe der Potenzmengenkonstruktion. / 2

(b) Geben Sie die LR(0)-Analysetabelle von G_{bool} an. / 2

(c) Bestimmen Sie die Konfigurationsfolge, die der LR(0)-Analyseautomat bei Eingabe des Wortes $((\neg \mathrm{t} \wedge \mathrm{f}) \wedge (\mathrm{f} \wedge \mathrm{t}))$ durchläuft. / 2