Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen
Sprachtechnologie und Compiler WS 2009/2010 http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de
Übungsleiter: Sebastian Buchwald sebastian.buchwald@kit.edu

Übungsblatt 7 | Ausgabe: 03.12.2009 | Besprechung: 09.12.2009

Aufgabe 1: *Auswertungsreihenfolgen*

1.1 *Eigenschaften*

Geben Sie attributierte Grammatiken die die folgenden Bedingungen erfüllen. Geben Sie zusätzlich zur Verdeutlichung einen beliebigen Parsebaum.

Geben Sie eine AG an, die:

- Nur synthetisierte Attribute enthält.
- Nur ererbte Attribute enthält.
- Ein Attribut enthält das sowohl synthetisiert als auch ererebt wird.
- Stets mit einem Baumdurchlauf berechenbar ist.
- Mehr als einen Baumdurchlauf für ihre Berechnung benötigen.
- Nur für manche Bäume berechenbar ist. Es sollte sowohl berechenbare als auch nicht berechenbar Bäume geben, in denen alle Grammatikproduktionen vorkommen.

Aufgabe 2: *Attributierte Grammatiken und* LL

2.1 *Bedingungen*

Welche Bedingungen müssen gelten damit eine attributierte Grammatik während des LL parsens berechnet werden kann?

2.2 *Auswertung der Attribute beim rekursiven Abstieg*

Gegeben sei folgende attributierte Grammatik:

	Produktion	**Attributierungsregeln**
1)	$E \rightarrow T\ E'$	$E.val = T.val + E'.val$
		$E'.constants = E.constants$
		$T.constants = E.constants$
2)	$E' \rightarrow +\ T\ E'_1$	$E'.val = T.val + E'_1.val$
		$T.constants = E.constants$
		$E'_1.constants = E.constants$
3)	$E' \rightarrow \varepsilon$	$E'.val = 0$
4)	$T \rightarrow F\ T'$	$T.val = F.val * T'.val$
		$T'.constants = T.constants$
		$F.constants = T.constants$
5)	$T' \rightarrow *\ F\ T'_1$	$T'.val = F.val * T'_1.val$
		$F.constants = T'.constants$
		$T'_1.constants = T'.constants$
6)	$T' \rightarrow \varepsilon$	$T'.val = 1$
7)	$F \rightarrow (E)$	$F.val = E.val$
		$E.constants = F.constants$
8)	$F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$
9)	$F \rightarrow$ **constant**	$F.val = F.constants.getConstantValue($**constant**$.symbol)$

Ergänzen Sie den folgenden rekursiven Abstieg um die Auswertung der Attributierungsregeln.

```
void E() {
    T();
    E'();
}

void E'() {
    switch(t) {
    case '+':
        t = nextSymbol();
        T();
        E'();
        break;
    case ')':
    case EOF:
        break;
    default: Fehler();
    }
}

void T() {
    F();
    T'();
}

void T'() {
    switch(t) {
    case '*':
        t = nextSymbol();
        F();
        T'();
        break;
    case '+':
    case EOF:
        break;
    default: Fehler();
    }
}

void F() {
    switch(t) {
    case '(':
        t = nextSymbol();
        E();
        if (t == ')')
            t = nextSymbol();
        else
            Fehler();
        break;
    case digit:
        break;
    case constant:
        break;
    default: Fehler();
    }
}
```