P. Thiemann, S. Wehr Wintersemester 2006/2007

Compilerbau

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

Übungsblatt 7

13.12.2006

Aufgabe 1 (Typüberprüfung mit Hilfe von Attributgrammatiken; 4 Punkte)

Die folgende Attributgrammatik dient zur Typberechnung einfacher arithmetischer Ausdrücke mit Operatoren `+`, `-`, `*`, `/` und `div` und Konstanten vom Typ *int* und *float*. Zur Vereinfachung sind alle Integer- bzw. Float-Konstanten durch das Terminal `i` bzw. `f` repräsentiert. Die Werte für das Attribut *typ* sind $V_{typ} = \{int, float\}$, die Werte für *op* sind $V_{op} = \{+, -, *, /, \texttt{div}\}$. Die Funktion id bezeichnet die Identitätsfunktion.

$$\langle \circ E \to E\ Aop\ T\rangle.\,\mathrm{typ} = f_{\langle \circ E \to E\ Aop\ T\rangle}(\langle E \to \circ E\ Aop\ T\rangle.\,\mathrm{typ}, \langle E \to E\ Aop \circ T\rangle.\,\mathrm{typ}) \quad \text{s.u.}$$
$$\langle \circ T \to T\ Mop\ F\rangle.\,\mathrm{typ} = f_{\langle \circ T \to T\ Mop\ F\rangle}(\langle T \to T \circ Mop\ F\rangle.\,\mathrm{op}, \langle T \to \circ T\ Mop\ F\rangle.\,\mathrm{typ}, \langle T \to T\ Mop \circ F\rangle.\,\mathrm{typ}) \quad \text{s.u.}$$

$\langle \circ E \to T\rangle.\,\mathrm{typ} = f_{\langle \circ E \to T\rangle}(\langle E \to \circ T\rangle.\,\mathrm{typ})$	$f_{\langle \circ E \to T\rangle} = \mathrm{id}$
$\langle \circ T \to F\rangle.\,\mathrm{typ} = f_{\langle \circ T \to F\rangle}(\langle T \to \circ F\rangle.\,\mathrm{typ})$	$f_{\langle \circ T \to F\rangle} = \mathrm{id}$
$\langle \circ P \to \texttt{i}\rangle.\,\mathrm{typ} = f_{\langle \circ P \to \texttt{i}\rangle}$	$f_{\langle \circ P \to \texttt{i}\rangle} = int$
$\langle \circ P \to \texttt{f}\rangle.\,\mathrm{typ} = f_{\langle \circ P \to \texttt{f}\rangle}$	$f_{\langle \circ P \to \texttt{f}\rangle} = float$
$\langle \circ P \to \texttt{(}\ E\ \texttt{)}\rangle.\,\mathrm{typ} = f_{\langle \circ P \to \texttt{(}\ E\ \texttt{)}\rangle}(\langle P \to \texttt{(} \circ E\ \texttt{)}\rangle.\,\mathrm{typ})$	$f_{\langle \circ P \to \texttt{(}E\texttt{)}\rangle} = \mathrm{id}$
$\langle \circ F \to P\rangle.\,\mathrm{typ} = f_{\langle \circ F \to P\rangle}(\langle F \to \circ P\rangle.\,\mathrm{typ})$	$f_{\langle \circ F \to P\rangle} = \mathrm{id}$
$\langle \circ Aop \to \texttt{+}\rangle.\,\mathrm{op} = f_{\langle \circ Aop \to \texttt{+}\rangle}$	$f_{\langle \circ Aop \to \texttt{+}\rangle} = \texttt{+}$
$\langle \circ Aop \to \texttt{-}\rangle.\,\mathrm{op} = f_{\langle \circ Aop \to \texttt{-}\rangle}$	$f_{\langle \circ Aop \to \texttt{-}\rangle} = \texttt{-}$
$\langle \circ Mop \to \texttt{*}\rangle.\,\mathrm{op} = f_{\langle \circ Mop \to \texttt{*}\rangle}$	$f_{\langle \circ Mop \to \texttt{*}\rangle} = \texttt{*}$
$\langle \circ Mop \to \texttt{/}\rangle.\,\mathrm{op} = f_{\langle \circ Mop \to \texttt{/}\rangle}$	$f_{\langle \circ Mop \to \texttt{/}\rangle} = \texttt{/}$
$\langle \circ Mop \to \texttt{div}\rangle.\,\mathrm{op} = f_{\langle \circ Mop \to \texttt{div}\rangle}$	$f_{\langle \circ Mop \to \texttt{div}\rangle} = \texttt{div}$

$$f_{\langle \circ E \to E\ Aop\ T\rangle}(a_1, a_2) = \begin{cases} int & \text{wenn } a_1 = a_2 = int \\ float & \text{sonst} \end{cases}$$

$$f_{\langle \circ T \to T\ Mop\ F\rangle}(a_1, a_2, a_3) = \begin{cases} int & \text{wenn } a_1 = \texttt{*} \wedge a_2 = a_3 = int \\ int & \text{wenn } a_1 = \texttt{div} \wedge a_2 = a_3 = int \\ float & \text{sonst} \end{cases}$$

Gib Typen für alle Teilausdrücke des arithemtischen Ausdrücke

$$42 \texttt{*} (3 \texttt{/} (12 \texttt{div} 5)) \texttt{+} 3$$

an, indem Du die Attributdekoration des Ableitungsbaumes ermittelst.

Aufgabe 2 (Typüberprüfung für Listen; 4 Punkte)
Die folgende Grammatik erzeugt Programme, P, die aus einer Folge von Deklarationen, D, und einem Ausdruck, E, bestehen, wobei Ausdrücke Listen über Literallisten repräsentieren und Deklarationen Variablen bestimmte Listentypen zuweisen.

$$\begin{array}{lll} P & \rightarrow & D \texttt{ ; } E \\ D & \rightarrow & D \texttt{ ; } D \mid \texttt{id : } T \\ T & \rightarrow & \texttt{list(}T\texttt{)} \mid \texttt{char} \mid \texttt{integer} \\ E & \rightarrow & \texttt{(}\, L \,\texttt{)} \mid \texttt{literal} \mid \texttt{num} \mid \texttt{id} \\ L & \rightarrow & E \texttt{ , } L \mid E \end{array}$$

Erweitere die Grammatik zu einer Attributgrammatik, so dass die Attributierung gerade die Typen aller Ausrücke (E) und Listen (L) bestimmt.

Aufgabe 3 (Semantische Analyse einfacher sequentieller Programme; 4 Punkte)
Die folgende Grammatik erzeugt Anweisungen, S, einer einfachen sequentiellen Programmiersprache.

$$\begin{array}{lll} S & \rightarrow & \texttt{var id} \\ S & \rightarrow & \texttt{if}\, E\, \texttt{then}\, S\, \texttt{else}\, S \\ S & \rightarrow & \texttt{while}\, E\, \texttt{do}\, S \\ S & \rightarrow & \texttt{id :=}\, E \\ S & \rightarrow & \texttt{return}\, E \\ S & \rightarrow & S \texttt{ ; } S \\ E & \rightarrow & \texttt{const} \\ E & \rightarrow & \texttt{id} \\ E & \rightarrow & E \texttt{ + } E \end{array}$$

Formuliere eine semantische Analyse mit Hilfe einer Attributgrammatik, die überprüft, dass Variablen erst dann referenziert werden (durch `id` bzw. `id :=` E), nachdem sie deklariert wurden (mit Hilfe einer Variablendeklaration `var id`), und

---

Abgabe: 20.12.2006
Die Abgabe erfolgt bis zu Beginn der Übungsstunde. Für Plagiate werden keine Punkte vergeben.