

Kapitel 7: Semantische Analyse

Aufgabe

Überprüfen und Auswerten der Typinformationen (Deklarationen)

Themen

- Typ – Ausdrücke
- Symboltabelle
- Typ – Überprüfung
- Overloading

Typ-Ausdrücke

Strukturierte Typen können durch Ausdrücke bzw. Bäume beschrieben werden

- Ein Basistyp ist ein Typ-Ausdruck, z.B. `bool`, `int`, `real`, `char`
- Ein Typ-Konstruktor, angewandt auf Typ-Ausdrücke T , ist ein Typ-Ausdruck

`array(n, T)`

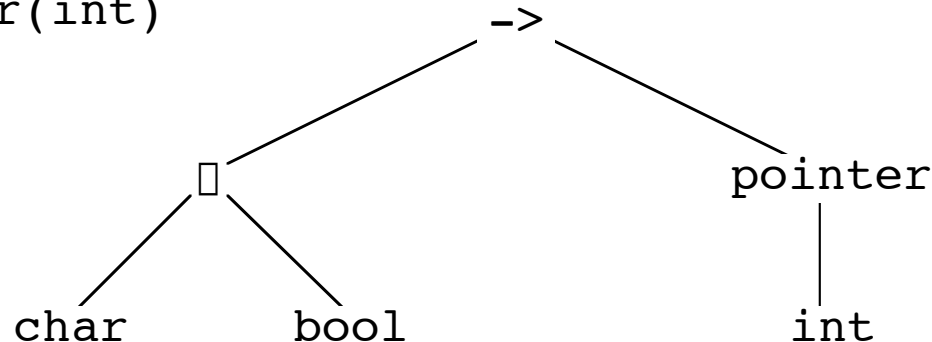
`T1 [] ... [] Tk`

`pointer(T)`

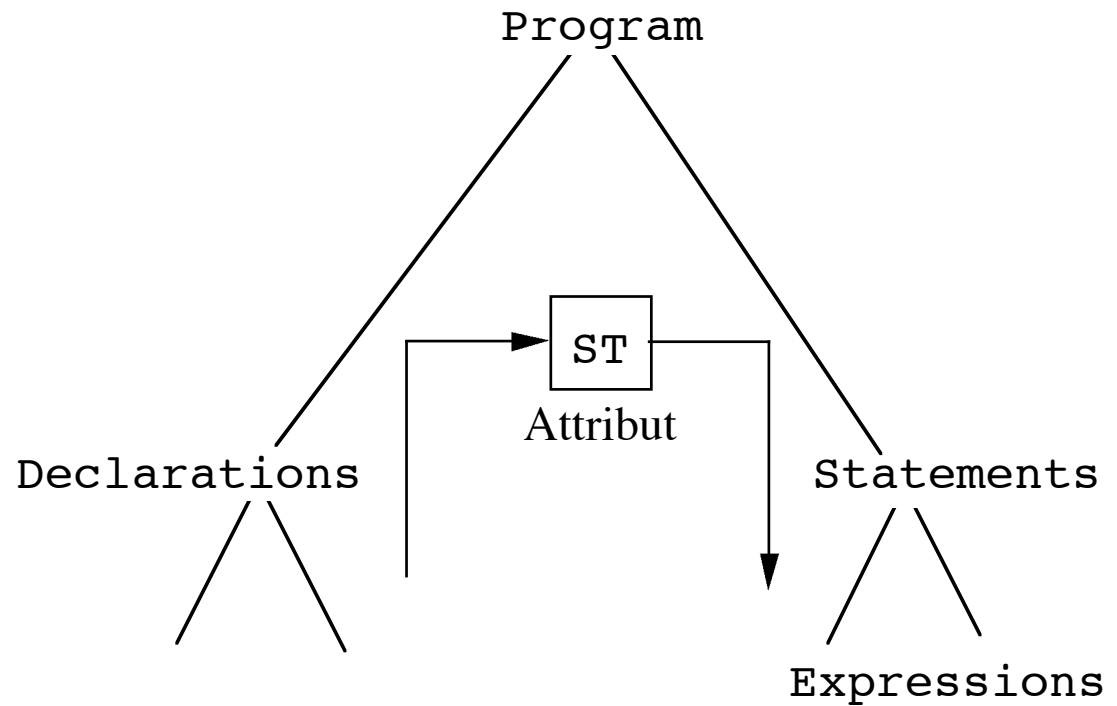
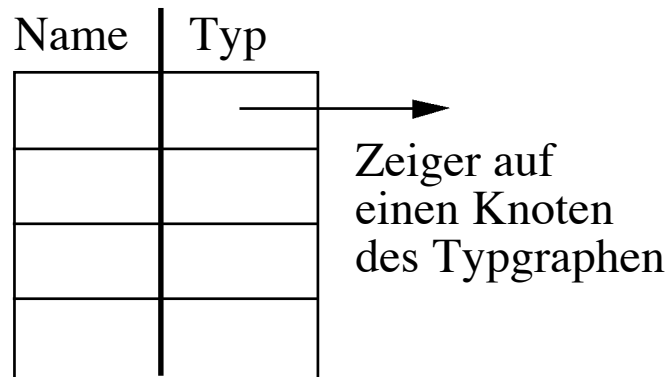
`T1 -> T2`

Typ-Ausdruck als Graph

`char [] bool -> pointer(int)`



Symboltabelle



AG zur Erzeugung der Symboltabelle

Produktion	Regeln
$P \rightarrow D S$	$D.i = ()$ $S.i = D.s$
$D \rightarrow D_1 ; D_2$	$D_1.i = D.i$ $D_2.i = D_1.s$ $D.s = D_2.s$
$D \rightarrow id : T$	$D.s = D.i \parallel (id.x, T.type)$
$T \rightarrow int$	$T.type = int$
$T \rightarrow real$	$T.type = real$
$D \rightarrow proc\ id\ P$	$h = D.i \parallel (id.x, proc)$ $D.s = h$ $P.i = h$
$P \rightarrow D S$	$D.i = P.i$ $S.i = D.s$

AG zur Typ-Überprüfung

Produktion	Regeln
$P \rightarrow D S$	$D.i = ()$ $S.i = D.s$
$S \rightarrow S_1 ; S_2$	$S_1.i = S.i$ $S_2.i = S.i$
$S \rightarrow id := E$	$E.i = S.i$ $if (looktype(id.x, S.i) \neq E.type) \text{ error}$
$E \rightarrow E_1 + E_2$	$E_1.i = E.i$ $E_2.i = E.i$ $E.type =$ $if (E_1.type == int \ \&\& \ E_2.type == int) \ int$ $else \ if (E_1.type == real \ \&\& \ E_2.type == real) \ real$ $else \ error$
$E \rightarrow id$	$E.type = looktype(id.x, E.i)$

Overloading in Ada

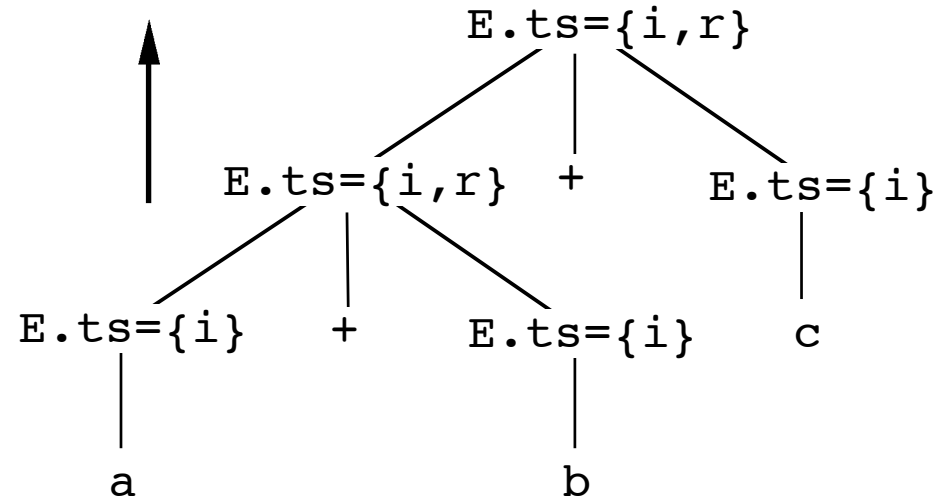
Operator wird durch Argument-Typen und Ergebnis-Typ identifiziert

`+: (1) int[]int->int, (2) real[]real->real, (3) int[]int->real`

`a,b,c:int; y:real`

`y = a+b+c`

1. Typmengen aufgrund des Ausdrucks



2. Eindeutigkeit durch Ergebnis-Typ

