Karlsruher Institut für Technologie

Praktikum Compilerbau
Sitzung 4 – Abstrakter Syntaxbaum

Prof. Dr.-Ing. Gregor Snelting
Matthias Braun und Sebastian Buchwald

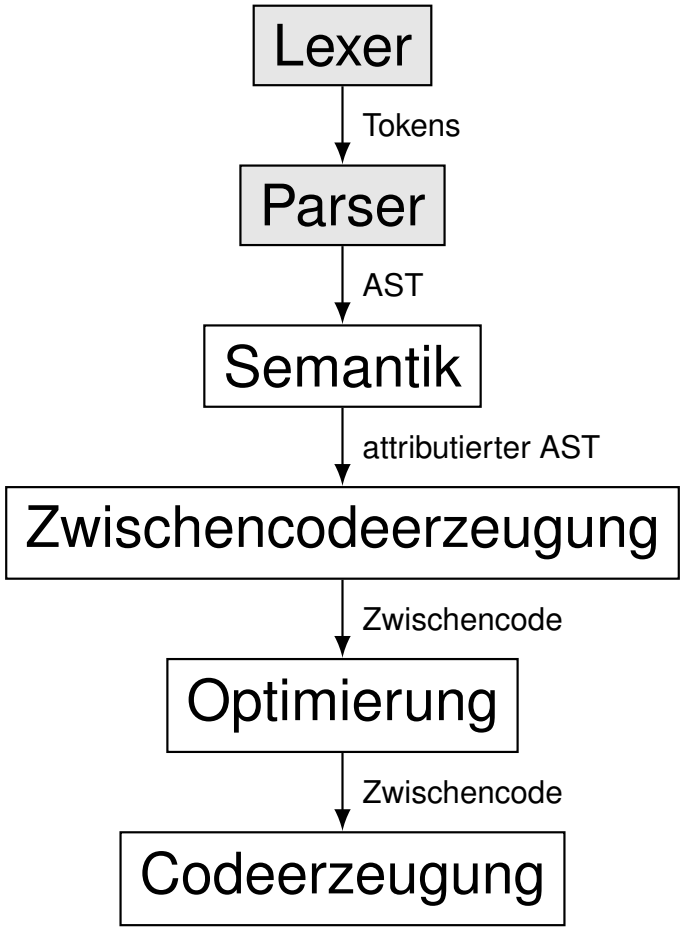
IPD Snelting, Lehrstuhl für Programmierparadigmen

1. Letzte Woche
2. Planung
3. Tipps & Tricks
4. Sonstiges

Letzte Woche

- Was waren eure Erfahrungen?

Compilerphasen



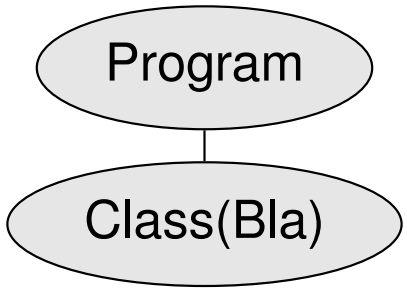
Aufgaben für diese Woche

- Entwurf des AST als abstrakte Algebra.
- Implementierung als reale Datenstrukturen.
- Anknüpfen der AST-Erzeugung an den Parser.

Was gehört in den AST?

- Semantisch unwichtige Dinge können weggelassen werden

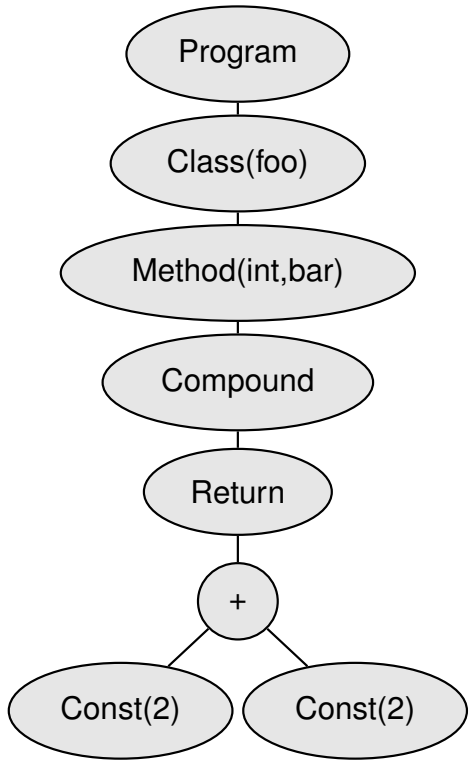
public class Bla { }



Was gehört in den AST?

- Code ist Kompositional
- AST stellt Hierarchie dar

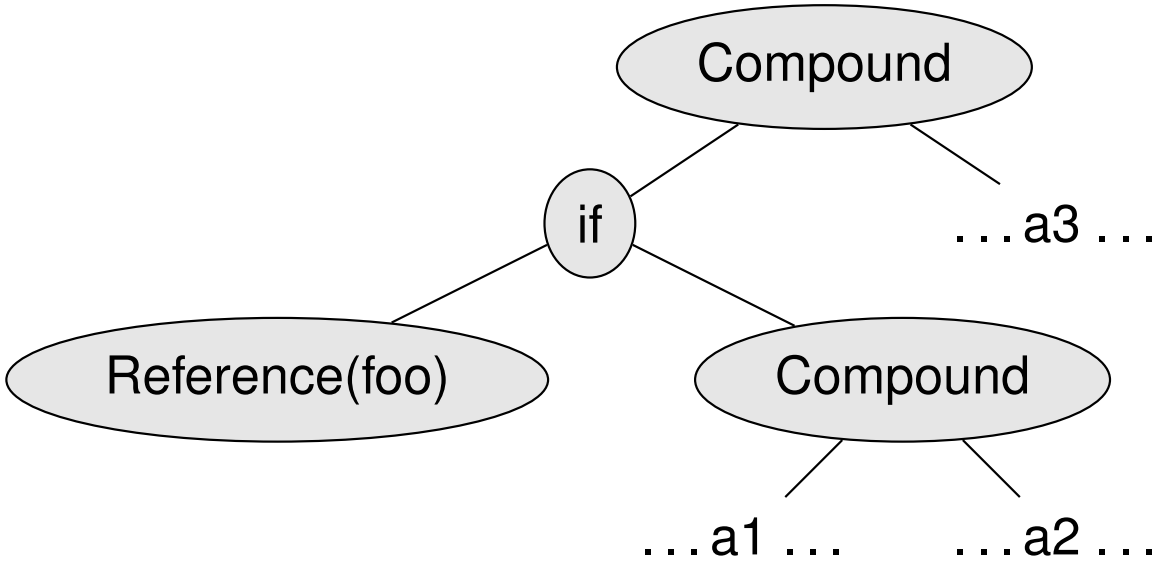
```
public class Foo {
   public int bar() { return 2 + 2; }
}
```



Was gehört nicht in den AST?

- Wörter die Konstrukte voneinander trennen

if (foo) { a1(); a2(); } a3();

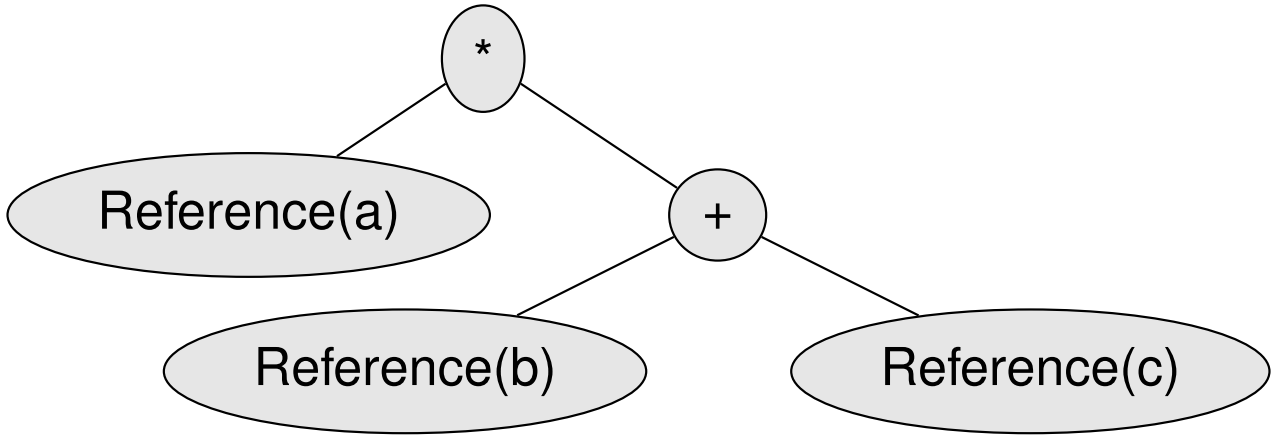


Was gehört nicht in den AST?

- Wörter die Mehrdeutigkeiten in der Hierarchie verhindern

$$a*(b+c)$$

Keine Klammern:



Syntax als abstrakte Algebra

Karlsruher Institut für Technologie

Beispiel: abstrakte Syntax für Expressions und Statements

$$
\begin{array}{rcl}
\mathit{Stmt} & = & \mathit{IfStmt} \mid \mathit{IfElseStmt} \mid \mathit{WhileStmt} \mid \mathit{Assignment} \mid \mathit{Block} \ldots \\
\mathit{IfStmt} & :: & \mathit{Expr}\ \mathit{Stmt} \\
\mathit{IfElseStmt} & :: & \mathit{Expr}\ \mathit{Stmt}\ \mathit{Stmt} \\
\mathit{WhileStmt} & :: & \mathit{Expr}\ \mathit{Stmt} \\
\mathit{Block} & :: & \mathit{Decls}\ \mathit{StmtList} \\
\mathit{StmtList} & :: & \mathit{Stmt}\ + \\
\mathit{Assignment} & :: & \mathit{Var}\,\mathit{Expr} \\
\mathit{Var} & :: & \texttt{Symbol}
\end{array}
\qquad
\begin{array}{rcl}
\mathit{Expr} & = & \mathit{Addop} \mid \mathit{MultOp} \mid \mathit{Var} \mid \ldots \\
\mathit{Addop} & :: & \mathit{Expr}\ \mathit{Expr} \\
\mathit{Multop} & :: & \mathit{Expr}\ \mathit{Expr}
\end{array}
$$

siehe Vorlesung *Sprachtechnologie und Compiler*

Tipps

- Nicht jede Produktion der Grammatik muss ein eigener AST-Knoten werden!
- Gemeinsame Basisklassen sinnvoll wo Alternativen in der Grammatik vorhanden sind.
  - Statements
  - Expressions
  - Types?
  - ClassMember?

Weitere Fragen

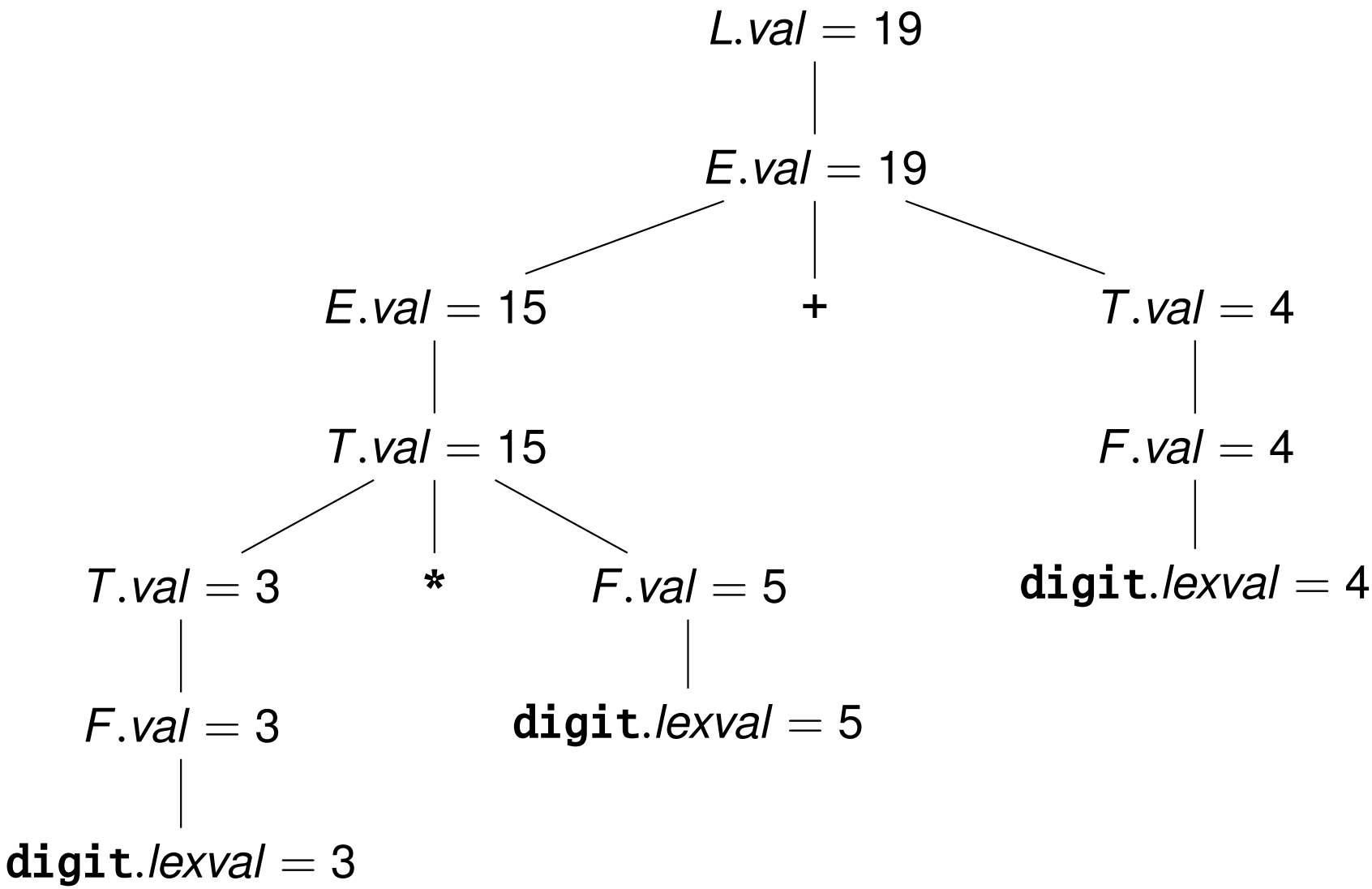
- Braucht man Verweise auf das Quellprogramm?
  - Warum (nicht)?
- Was sollte ein Attribut werden, was ein eigener Knoten im AST?
  - Wie ist das bei Typen oder Bezeichnern?

Beispiel: Taschenrechner mit Attributierter Grammatik

Karlsruher Institut für Technologie

	Produktion	Semantische Regeln
1)	$L \rightarrow E$	$L.val = E.val$
2)	$E \rightarrow E_1$ **+** T	$E.val = E_1.val + T.val$
3)	$E \rightarrow T$	$E.val = T.val$
4)	$T \rightarrow T_1$ ***** F	$T.val = T_1.val * F.val$
5)	$T \rightarrow F$	$T.val = F.val$
6)	$F \rightarrow$ **(**E**)**	$F.val = E.val$
7)	$F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$

Attributierter Parsebaum für `3 * 5 + 4`



Implementierung Attributierter Grammatiken

- ererbte Attribute werden zu Parameter
- synthetisierte Attribute werden zu Rückgabewerten

Feedback! Fragen? Probleme?

Karlsruher Institut für Technologie

- Wie läuft die Arbeitseinteilung?
- Anmerkungen?
- Probleme?
- Fragen?