Programmiersprache

Syntax

formale Beschreibung der Kompo-nenten eines Programms

mittels kontextfreier Grammatiken bzw. BNF

Pragmatik

Handhabung der Sprache/ Sprachkon-zepte

Implementie-rung

Werkzeuge

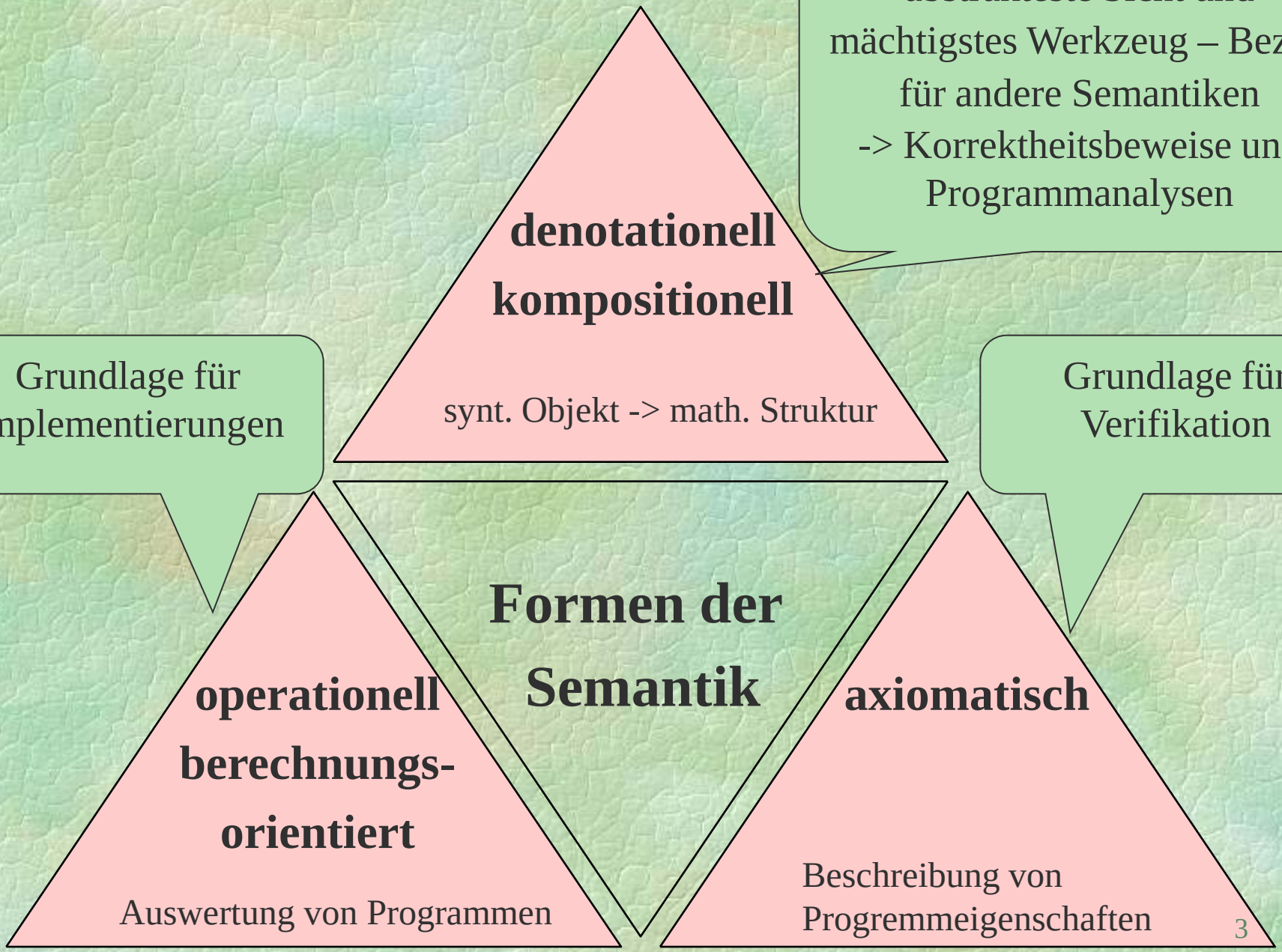
Semantik

Bedeutung der Sprache /Progr.

oft informell; **formale Methoden** vermeiden Unvollständig-keit und Mehr-deutigkeiten

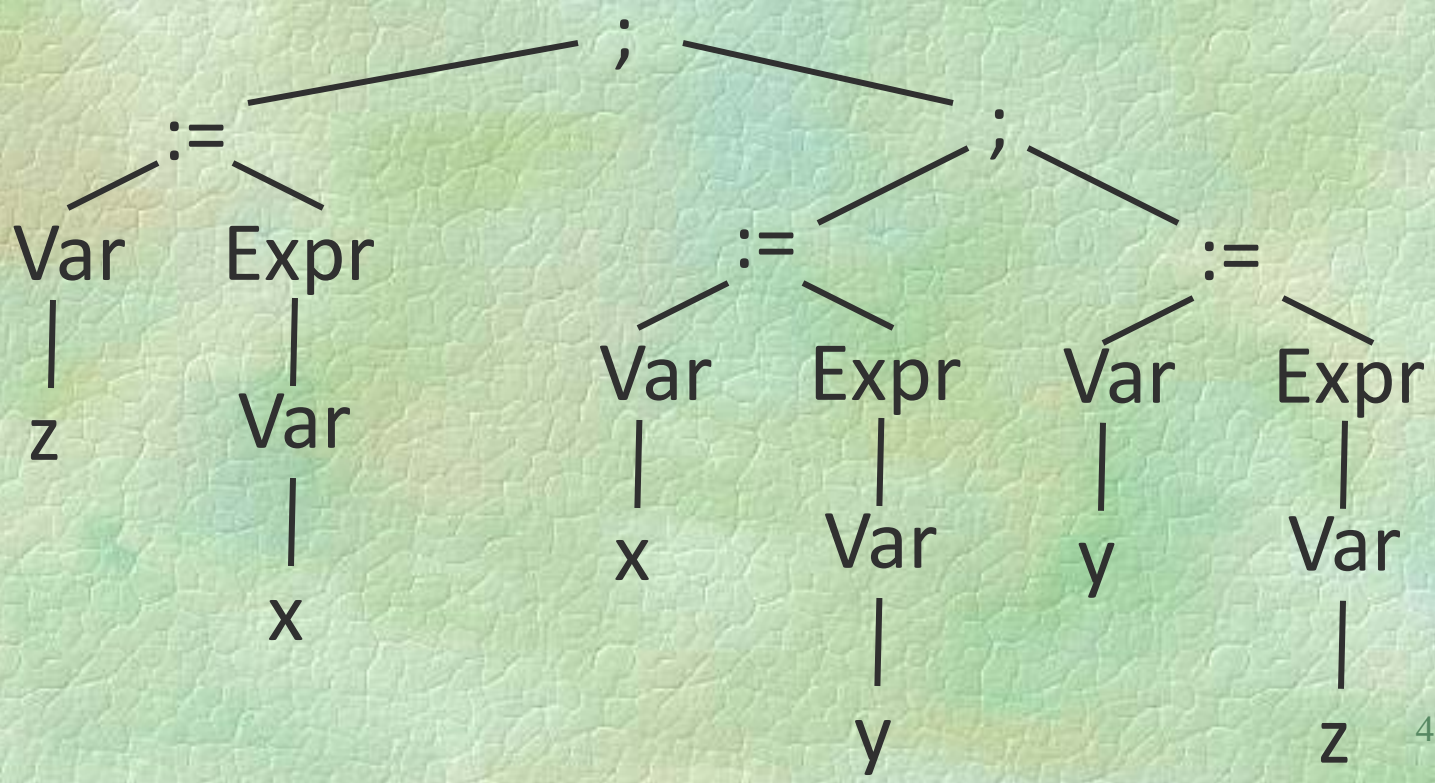
Semantik-Anwendungen

- implementierungsunabhängiger Referenzstandard für Sprachen
  -> Validierung von Implementierungen
- Programmverifikation und –analyse
- Programmoptimierung / -transformation
- automatische Werkzeugerzeugung (Compiler, Debugger, Optimierer…)
- Theorie der Programmierung – Verbesserung des Verständnisses von Programmiersprachen
- ….


abstrakteste Sicht und mächtigstes Werkzeug – Bezug für andere Semantiken -> Korrektheitsbeweise und Programmanalysen
denotationell kompositionell
synt. Objekt -> math. Struktur
Grundlage für Implementierungen
Grundlage für Verifikation
Formen der Semantik
operationell berechnungs-orientiert
Auswertung von Programmen
axiomatisch
Beschreibung von Progremmeigenschaften

Beispiel

- Anweisung z := x; x := y; y:=z
- Syntaktische Analyse



Konkrete vs abstrakte Syntax

konkrete Syntax:

- Syntaxbaum gemäß der die Sprache definierenden kontextfreien Grammatik
- benutzerfreundliche Mixfixnotation unter Verwendung natürlicher Sprache für Terminalsymbole

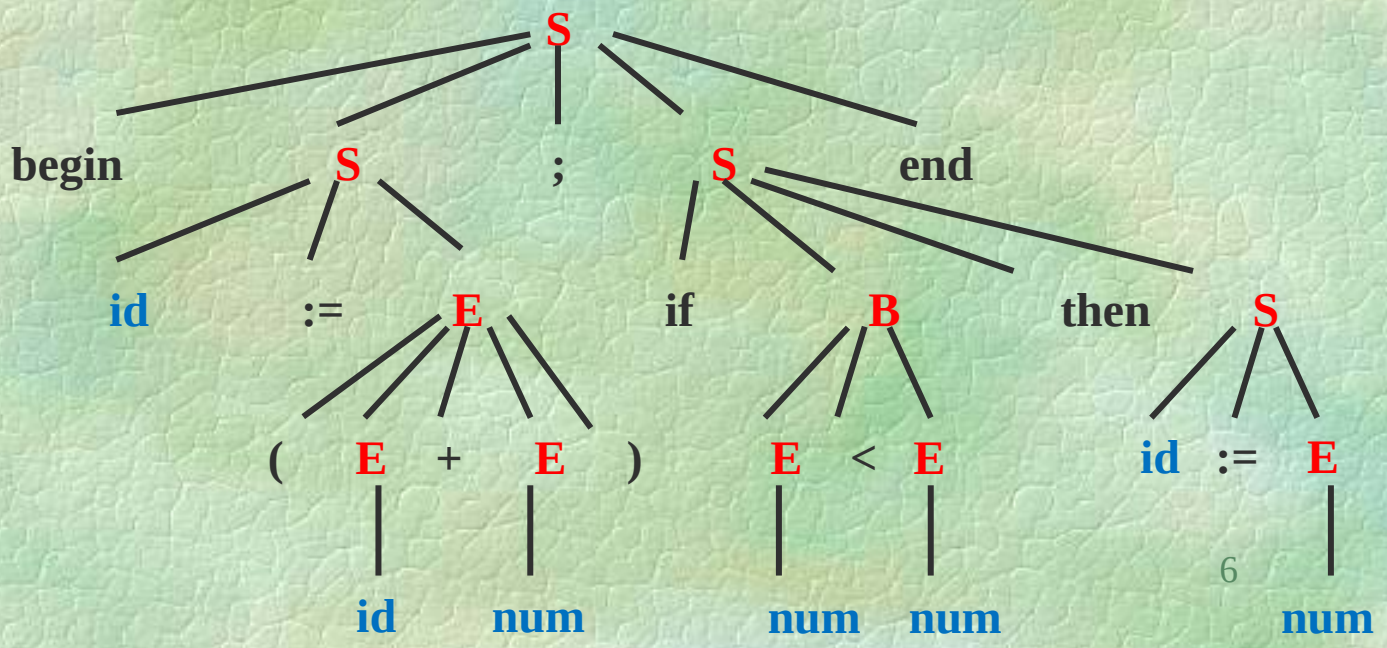
abstrakte Syntax:

- darstellungsunabhängige algebraische Struktur
- identifiziert im Programmwort auftretende Konstrukte und Schachtelungsbeziehung
- ausreichend für semantische Analyse

Syntax-definition

```
% Anweisungen (Statements)
S  ->   id := E                    % Wertzuweisung
    |   begin  S ; S end           % Hintereinanderausführung
    |   if  B then  S              % bedingte Anweisung
% Ausdrücke (Expressions)
E ->    ( E + E )                  % Summe
    |   id                         % Bezeichner
    |   num                        % Zahl
% Boolesche Ausdrücke (Boolean expressions)
B ->    E < E                      % Vergleichsausdruck
```

Syntax-baum

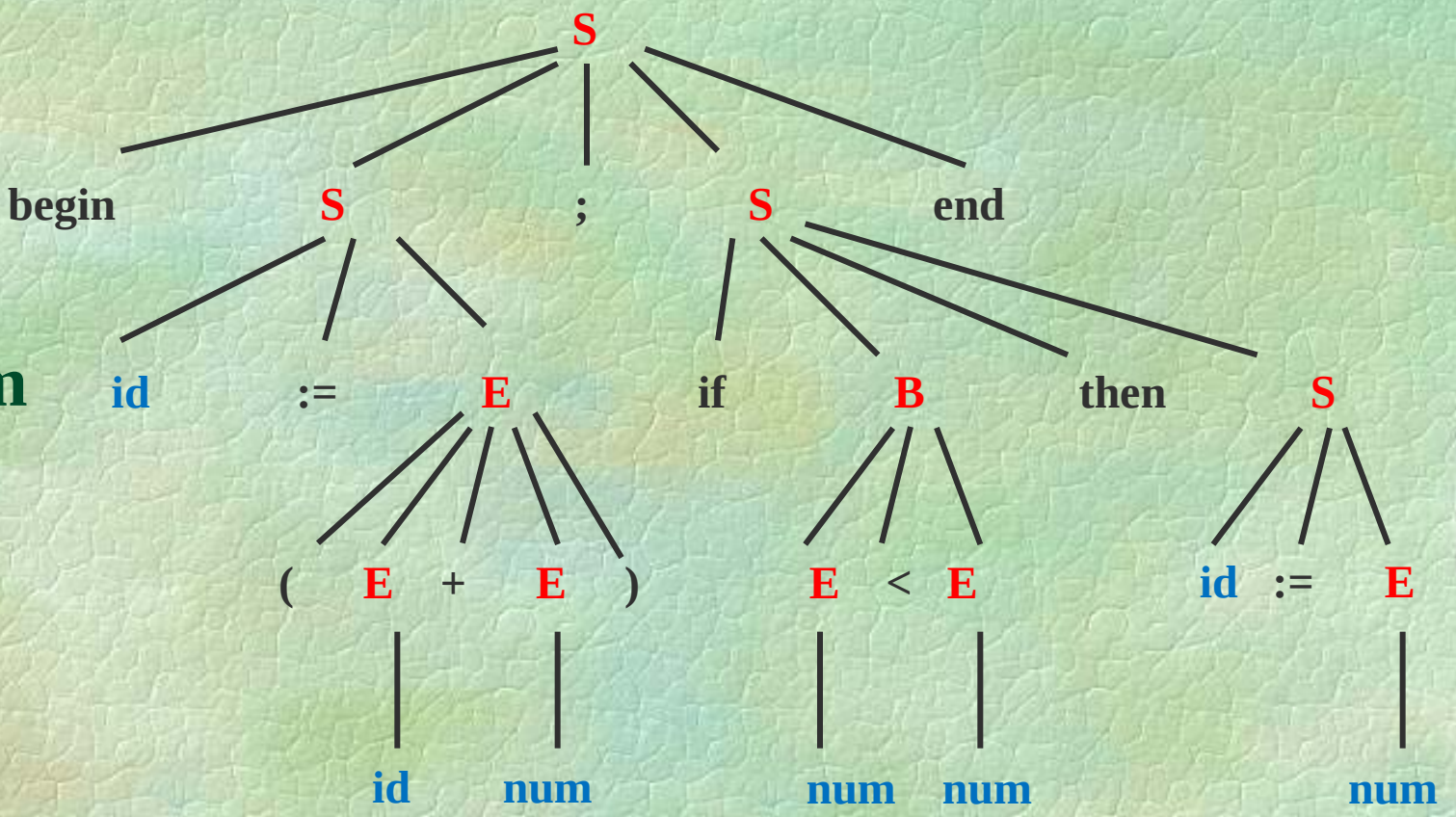


Konkrete Syntax => Abstrakte Syntax

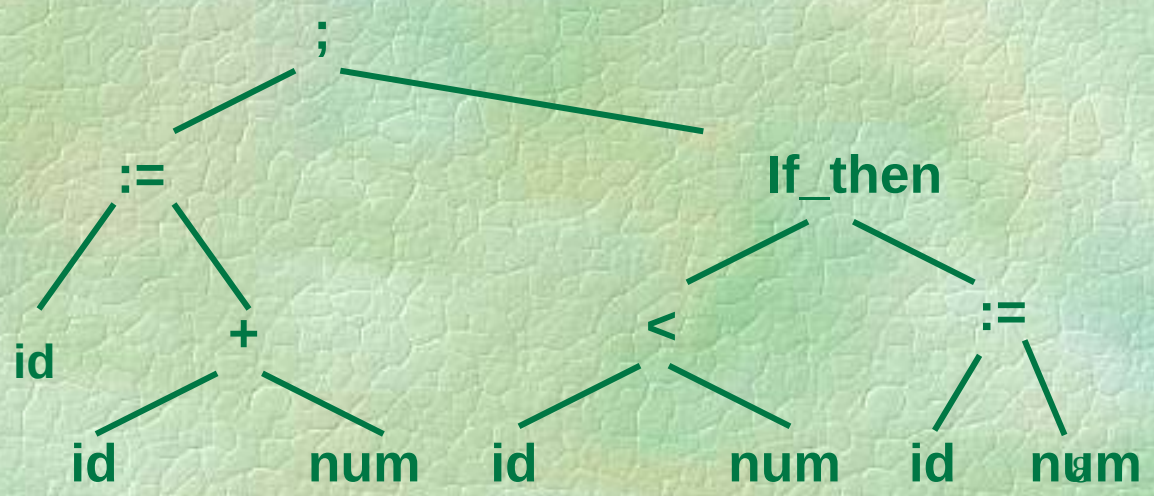
Regeln	=>	Operationssymbole
$\pi = A_0 \rightarrow w_0 A_1 w_1 \ldots A_n w_n$	=>	$F_\pi :: A_1 \times \ldots \times A_n \rightarrow A_0$

S ->	id := E	=>	:=	: I x E	–> S
|	begin S ; S end	=>	;	: S x S	–> S
|	if B then S	=>	if_then	: B x S	–> S
E ->	E + E	=>	+	: E x E	–> E
|	id	=>	id	:	–> E
|	num	=>	num	:	–> E
B ->	E < E	=>	<	: E x E	–> B

Konkreter Syntaxbaum



Abstrakter Syntaxbaum



Die Modellsprache While

- **syntaktische Bereiche**
  - **Zahlen n ∈ Num und Wahrheitswerte t ∈ T := {true, false}**
  - **Variablen (Speicherplatzbezeichner) X,Y ∈ Loc**
  - **arithmetische und Boolesche Ausdrücke**
    a ∈ AExpr und b ∈ BExpr
  - **Anweisungen c ∈ Cmd**
- **BNF-Definitionen**

AExp : a ::= n | X | $a_0 + a_1$ | $a_0 * a_1$ | $a_0 - a_1$

BExp : b ::= <u>true</u> | <u>false</u> | $a_0 = a_1$ | $a_0 \leq a_1$ | $\neg b$ | $b_0 \wedge b_1$ | $b_0 \vee b_1$

Cmd : c ::= <u>skip</u> | X:=a | c_0 ; c_1 | <u>if</u> b <u>then</u> c_0 <u>else</u> c_1 | <u>while</u> b <u>do</u> c