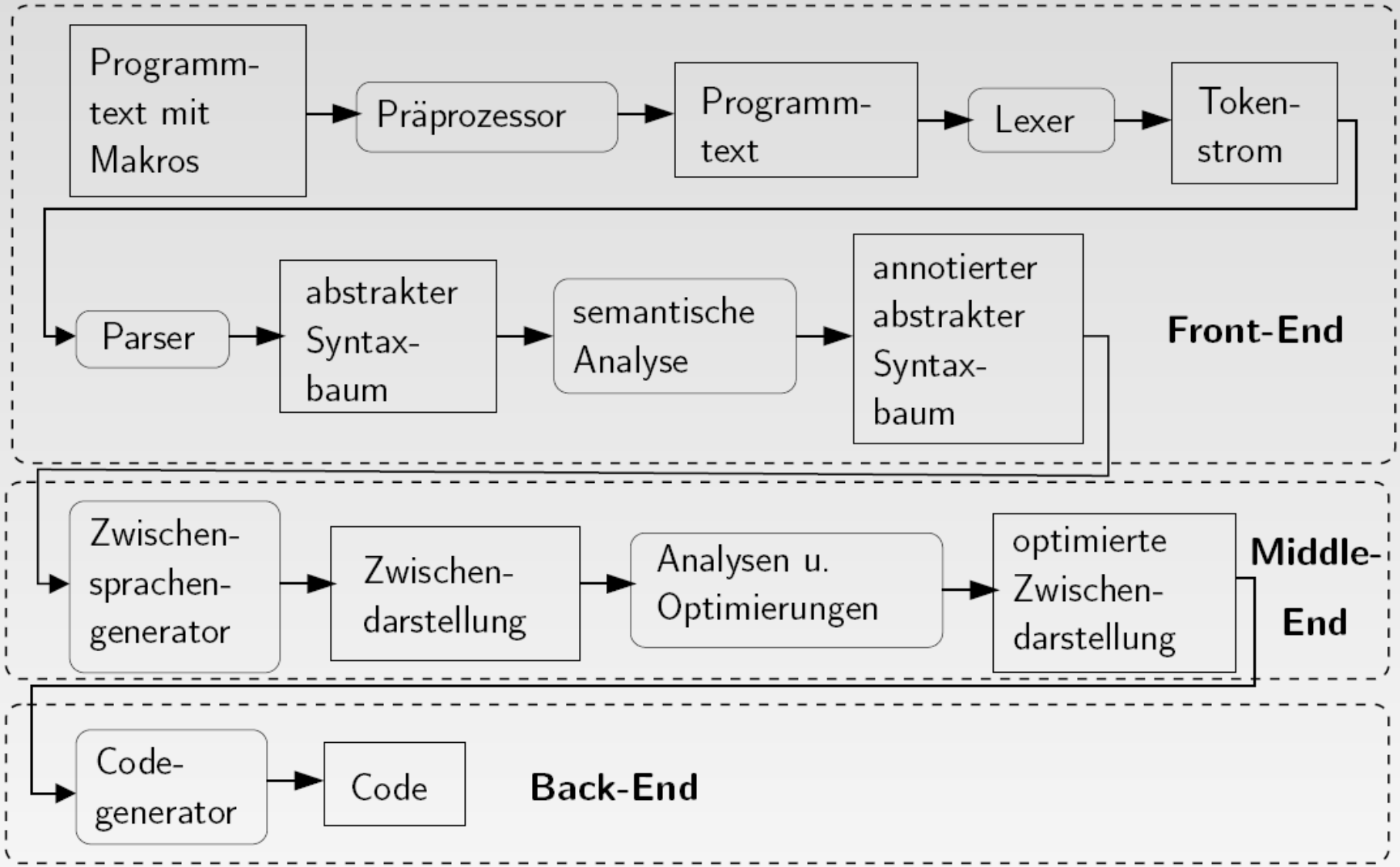
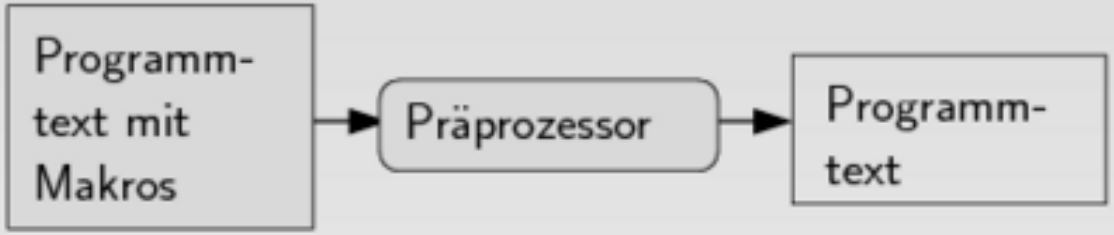
Compiler-Struktur



Analysator- und Transformator-Struktur



Phasen eines Compilers

- Wissen über das Programm nimmt zu
  - syntaktische Dekomposition
  - semantische Attributierung
  - Namensbindung
  - Kontrollflussinformation
  - Datenflussinformation
- Abstraktion nimmt ab
  - abstrakter Syntaxbaum
  - maschinennahe einheitliche Zwischensprache, z.B. Register Transfer Language (RTL) von Gnu GCC
  - Maschinensprache

Unterschiede Compiler / Analysator

- lokal versus global
- optimistisch versus pessimistisch
- quellennah versus sprachenunabhängig

Der Tokenstrom

Für das Programmstück

```
declare
  X: real;
begin
  X := A * 5;
end;
```

liefert der Lexer neben der Quellposition die Lexeme (Token) durch Integer-Kennung der erkannten Kategorie und ggf. die Zeichenkette:

```
 9 — "declare"
 4 — id,        "X"
 7 —  ":"
 4 — id,        "real"
 5 — ";"
 6 — "begin"
 4 — id,        "X"
 3 —  ":="
 4 — id,        "A"
 8 —  "*"
10 — int_lit,   "5"
 5 — ";"
11 — "end"
```

Der Lexer

Reguläre Grammatik

```
D                [0-9]
L                [a-zA-Z_]
"else"           { return ELSE; }
...
{L}({L}|{D})*    { yylval.id = register_name(yytext); return ID; }
```

Lexergenerator: (a)(f)lex, Cocktail(95)

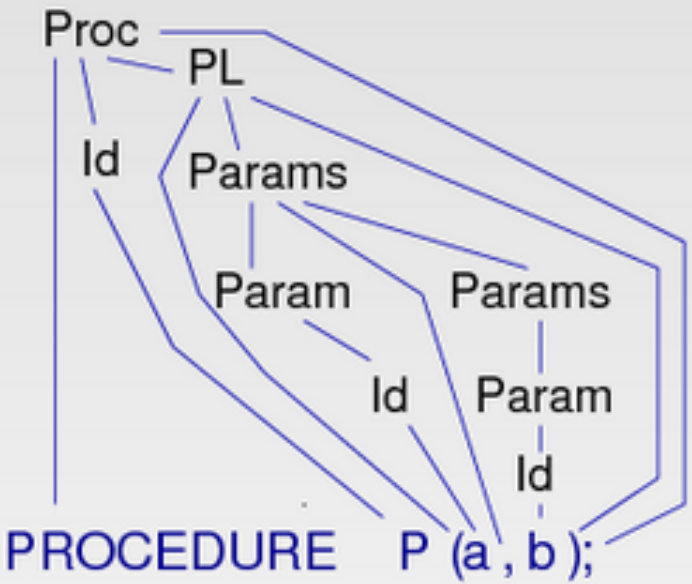
Lexer

Abstrakte Syntaxbäume

- Darstellung der syntaktischen Dekomposition des Eingabeprogramms
  - vereinfachter Ableitungsbaum: keine Kettenregeln, Sequenzen ersetzen Rekursionen etc.
  - syntaktische Kanten bilden einen Baum
- Eine formale Syntaxbeschreibung (BNF) legt die syntaktische Struktur fest:

```
Proc   ::= PROCEDURE Id PL ";" .
PL     ::= "(" Params ")" | ϵ .
Params ::= Param "," Params | Param .
Param  ::= Id .
```

Ableitungsbaum / abstrakter Syntaxbaum



Der Parser

Grammatik

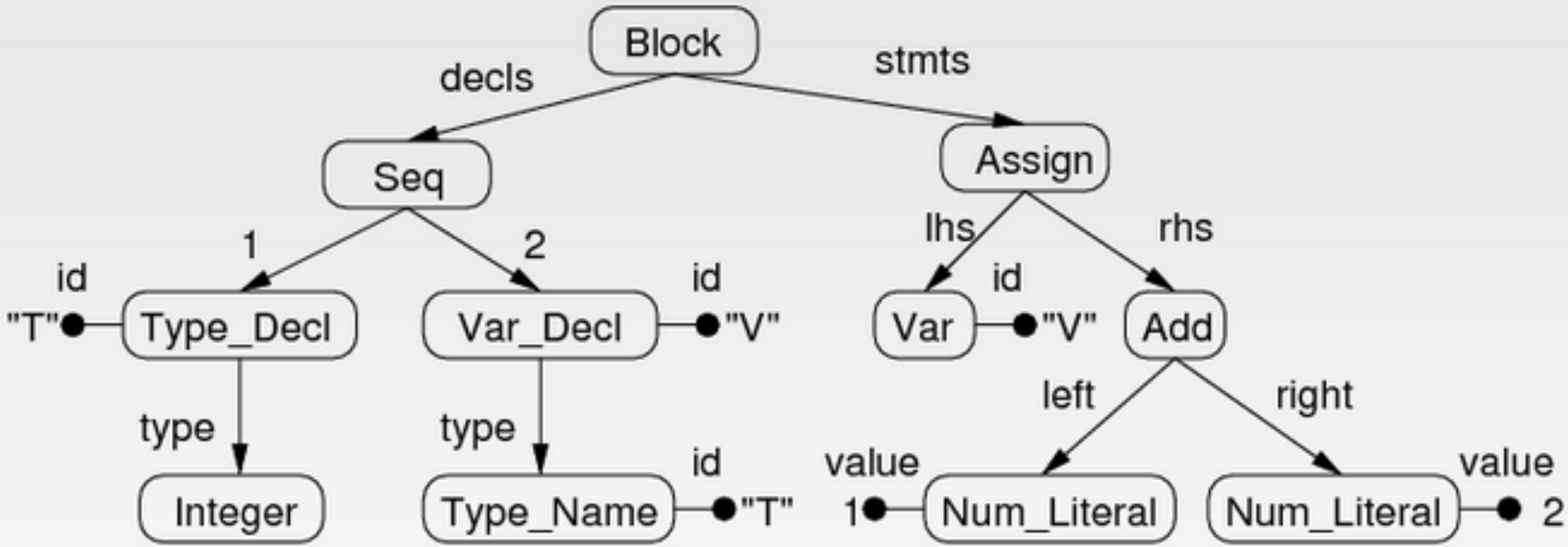
```
multiplicative_expression =-> [Tree:AST] <

= LOp:multiplicative_expression Op:"*" ROp:cast_expression
  {Tree := Make_Binary_Mult (LOp:Tree, ROp:Tree, Op:Position);
  } .
```

Parsergenerator: Ayacc, Bison, Cocktail(95)

Parser

Beispiel eines ASTs

```
declare
   type T is new Integer;
   V : T;
begin
   V := 1 + 2;
end;
```



Attributierter Syntaxbaum

Definition

Syntaktische Kanten des ASTs repräsentieren syntaktische Dekomposition.

→ bilden einen Baum

Semantische Analyse attributiert AST mit semantischen Informationen

- z.B. die Namens- und Typbindung.

Definition

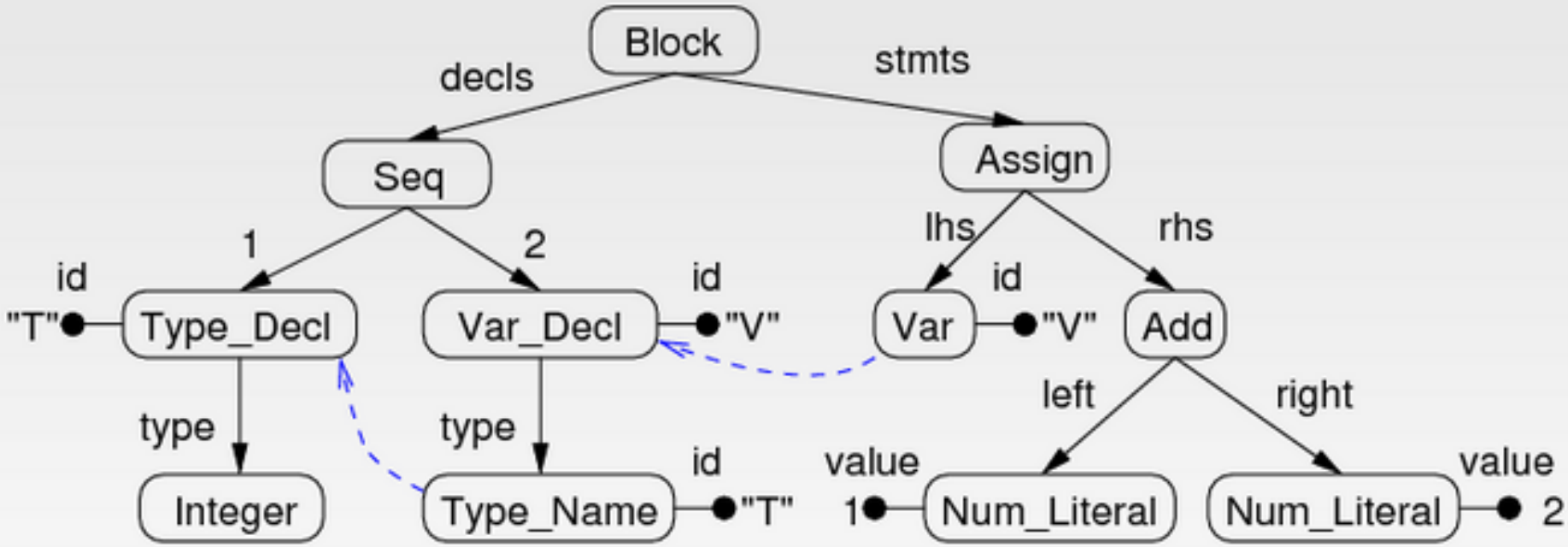
Semantische Kanten repräsentieren semantische Verweise auf andere Knoten.

→ bilden einen allgemeinen Graphen

→ Abstract Syntax Graph (ASG)

Namensbindung

```
declare
  type T is new Integer;
  V : T;
begin
  V := 1 + 2;
end;
```



Typinformation

```
declare
   type T is new Integer;
   V : T;
begin
   V := 1 + 2;
end;
```