Einführung in die Informatik I

Winter 2005/2006

7 Einführung in Programmiersprachen

Versionsdatum: 10. November 2005

7 Einführung in Programmiersprachen ...

Motivation:

Zeitliche Entwicklung: seit Mitte 50er Jahre (s. nächste Folie)

Unterschiedliche Typen:

- imperativ-prozedurale Programmiersprachen
- objekt-orientierte Programmiersprachen
- funktionale Programmiersprachen
- logische Programmiersprachen

Herausforderungen: verschiedene Typen von Programmiersprachen und deren Vertreter unterscheiden lernen

Literatur:

[Ernst] Teil des Kap. 7

[Watt] Watt, D.A.: Programmiersprachen - Konzepte und Paradigmen, Hanser, 1996

7.1 Programm

Programm: Realisierung eines Algorithmus

- wird automatisch von einem Prozessor (CPU) ausgeführt
- konkreter und detaillierter als ein in Worten formulierter Algorithmus
- genügt den Regeln einer (formalen) Programmiersprache

Programmieren: Erstellung eines Programms

Maschinensprache: Formulierung eines Maschinenprogramms als Folge von elementaren Befehlen (Binär-Code)

- Programmierung mit Hilfe von Assembler-Sprachen

Lowlevel-Programmiersprachen: Maschinennahe Befehle, hardware-orientiert

Highlevel-Programmiersprachen: Komplexe Befehle, problem-orientiert, Abstraktion/Unabhängigkeit von Hardware-Merkmalen

- Highlevel-Programme werden in Lowlevel-Programme übersetzt
  $\rightsquigarrow$ Compiler, Interpreter

7.2 Anwendungsgebiete

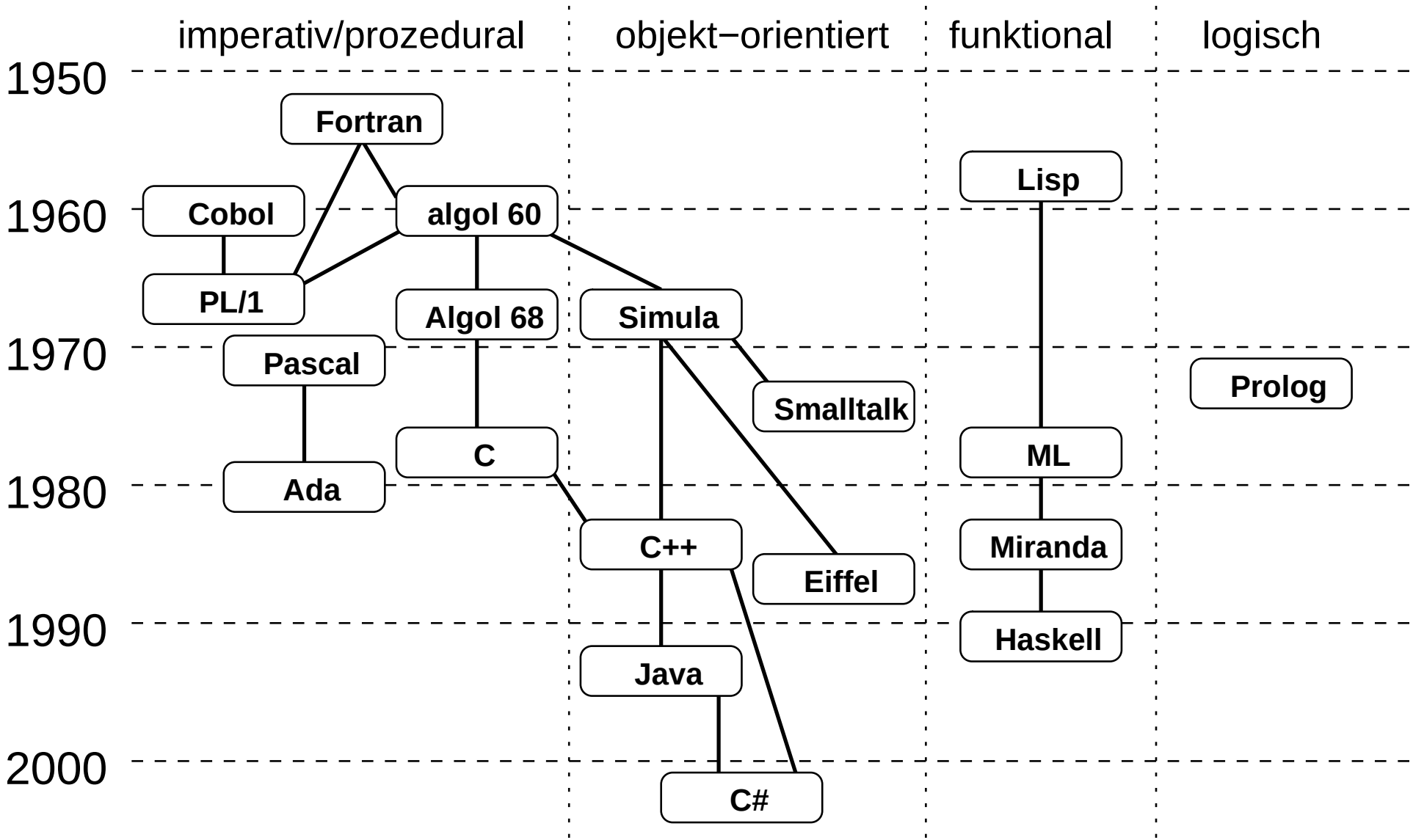
Unterschiedliche Anwendungsgebiete führen zu unterschiedlichen Sprachen, z.B.:

- Büro-Anwendungen (C++, C#, . . .)
- Web-Anwendungen (Java, Perl, PHP, Python, . . .)
- Wissenschaftliche Anwendungen (C, Fortran, . . .)
- Datenbank-Anwendungen (PHP, MySQL, . . .)
- Graphik-Anwendungen (C++, Assembler, Nvidia Cg, . . .)
- Server-Anwendungen/Unternehmenssoftware (C/C++, . . .)
- Eingebettete Systeme (VHDL, . . .)
- Kaufmännische Systeme (Cobol, . . .)

Visuelle Sprachen (Diagrammsprachen) als Hilfsmittel für den Design-Prozess

- UML (Unified Modeling Language)
- UML-Tools zur Code-Generierung

7.3 Zeitliche Entwicklung



imperativ/prozedural (z.B. Pascal, C, Basic):

Pascal-Beispiel:

```
procedure do(x, y: integer);
begin
  x := y; y := 0;
  writeln(x, y);
end;
```

- Hauptgesichtspunkt: Umsetzung von Algorithmen in Form von Prozeduren/Funktionen
- Kontrollfluss wird durch das Programm gesteuert
- Sprachelemente: Variablen, arithmetische Operatoren, Funktionen, Kontrollstrukturen

objekt-orientiert (z.B. C++, Java):

C++-Beispiel:

```
vector<Entity*> iterator entity;
for (entity = v.begin(); entity != v.end(); entity++)
  (*entity).do();
```

- Hauptgesichtspunkt: Datenorganisation anhand Abstrakter Datentypen
- Fokussierung auf grundlegende Konzepte der Software-Entwicklung: Modularisierung, Erweiterbarkeit, Wartbarkeit, Portabilität
- Sprachelemente: Klassen, Objekte, prozedurale Strukturen

funktional (z.B. LISP, SML, Haskell):

LISP- bzw. Scheme-Beispiel:

```
(define (fakultaet n)
  (if (= n 0)
    1
    (* n (fakultaet(- n 1)))))
```

- Hauptgesichtspunkt: Definition von rekursiven Funktionen
- Ausführungsreihenfolge ist durch Verschachtelung von Funktionen festgelegt
- Sprachelemente: Funktionen, arithmetische Operationen, keine sequentiellen Kontrollstrukturen

logisch/prädikativ (z.B. Prolog):

Prolog-Beispiel:

```
umkreist(erde, sonne).                          (Faktum)
umkreist(mars, sonne).
planet(B) :- umkreist(B, sonne).                (Regel)
satellit(B) :- umkreist(B, P), planet(P).
?- umkreist(erde, sonne).                       (Anfrage)
```

- Hauptgesichtspunkte: Prädikate, Regeln, Termersetzung
- Umsetzung der prädikatenlogischen Denkweise
- Sprachelemente: Fakten, Regeln, Anfragen

7.7 Ebenen der Sprachverarbeitung

Lexikalische Analyse: Zerlegung in Tokens

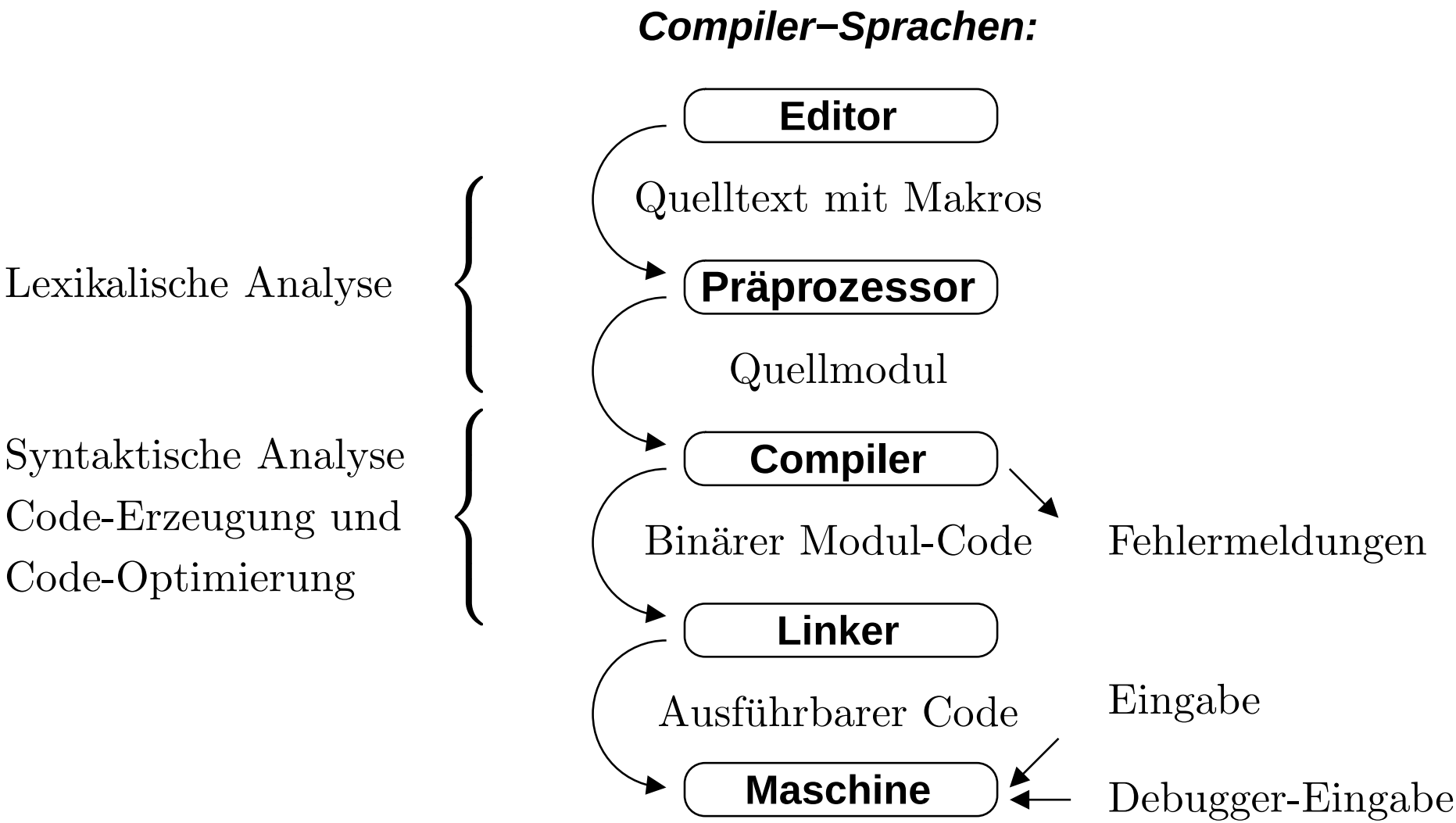
- z.B. Bezeichner, Literale (Zahlen, Zeichenketten), Schlüsselwörter
- formal definiert durch reguläre Ausdrücke/Grammatiken

Syntaktische Analyse: Zerlegung der Struktur von Sprachkonstrukten

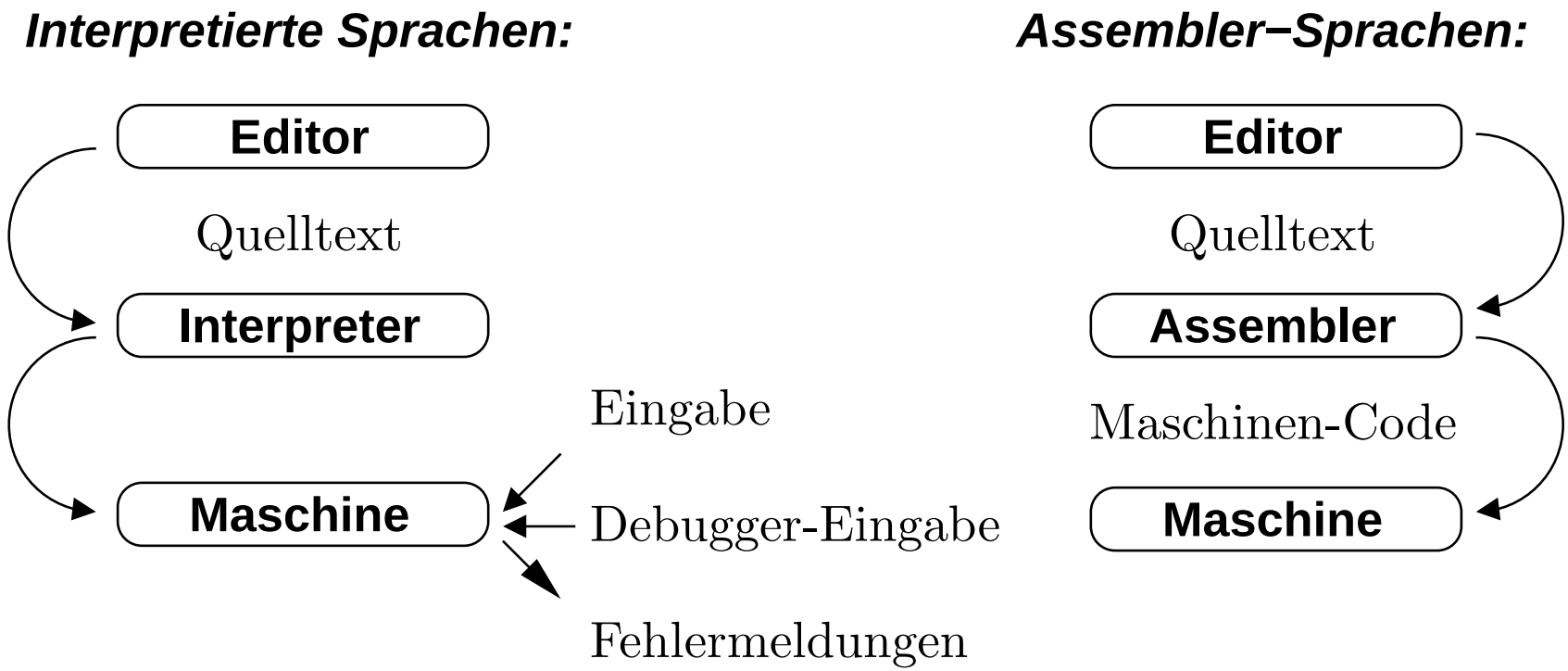
- formal definiert z.B. durch eine BNF-Syntax
- Ergebnis ist ein Ableitungsbaum

Semantische Analyse:

- Statische Zusammenhänge: Zum Beispiel Gültigkeit von Definitionen, Typregeln
- Dynamische Zusammenhänge: Wirkung und Bedeutung (meist verbal definiert (formale Definition ist aber möglich – operationale, mathematische Semantik)


Compiler–Sprachen:
Editor
Quelltext mit Makros
Präprozessor
Quellmodul
Compiler
Binärer Modul-Code
Fehlermeldungen
Linker
Ausführbarer Code
Eingabe
Maschine
Debugger-Eingabe
Lexikalische Analyse
Syntaktische Analyse
Code-Erzeugung und
Code-Optimierung

7.9 Programmierumgebung



Interpreter: durchläuft den Quell-Code und führt jeden Befehl direkt aus

- Beispiele: BASIC, Skriptsprachen (z.B. Perl, PHP, Python etc.)

Assembler: übersetzt eine Folge von Assembler-Befehlen in Maschinen-Code