

# Compiler I: Grundlagen

## Einleitung

Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen  
Informatik, TU Darmstadt

Wintersemester 2011/12

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Schnittstelle zwischen
  - Programmiersprache
  - Maschine

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Schnittstelle zwischen
  - Programmiersprache
  - Maschine

**Programmiersprache** Gut für Menschen handhabbar

- Smalltalk
- Java
- C++

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Schnittstelle zwischen
  - Programmiersprache
  - Maschine

**Programmiersprache** Gut für Menschen handhabbar

- Smalltalk
- Java
- C++

**Maschine** Getrimmt auf

- Ausführungsgeschwindigkeit
- Preis/Chip-Fläche
- Energieverbrauch
- Nur selten: Leichte Programmierbarkeit

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Entscheidet über dem Benutzer **zugängliche**  
Rechenleistung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Entscheidet über dem Benutzer **zugängliche**  
Rechenleistung

## Beispiel: Bildkompression auf Dothan CPU, 2GHz

Compiler	Ausführungszeit	Programmgröße
GCC 3.3.6	7,5 ms	13 KB

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Entscheidet über dem Benutzer **zugängliche**  
Rechenleistung

## Beispiel: Bildkompression auf Dothan CPU, 2GHz

Compiler	Ausführungszeit	Programmgröße
GCC 3.3.6	7,5 ms	13 KB
ICC 9.0	6,5 ms	511 KB

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Hohe Ebene:  
Smalltalk, Java, C++

```
let
  var i : Integer;
in
  i := i + 1;
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu



Hohe Ebene:  
Smalltalk, Java, C++

```
let
  var i : Integer;
in
  i := i + 1;
```

Mittlere Ebene:  
Assembler

```
LOAD R1, (i)
LOADI R2, 1
ADD R1, R1, R2
STORE R1, (i)
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Hohe Ebene:  
Smalltalk, Java, C++

```
let
  var i : Integer;
in
  i := i + 1;
```

Mittlere Ebene:  
Assembler

```
LOAD R1, (i)
LOADI R2, 1
ADD R1, R1, R2
STORE R1, (i)
```

Niedrige Ebene:  
Maschinensprache

```
0110000100000110
0111001001000001
1011000100010010
1001000100000110
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Auf unteren Ebenen immer feinere Beschreibung
- Immer näher an Zielmaschine (Hardware)
- Details werden von Compiler hinzugefügt
  - Durch verschiedenste Algorithmen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Auf unteren Ebenen immer feinere Beschreibung
- Immer näher an Zielmaschine (Hardware)
- Details werden von Compiler hinzugefügt
  - Durch verschiedenste Algorithmen
    - Analyse von Programmeigenschaften
    - Verfeinerung der Beschreibung durch Synthese von Details

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

# Auswirkungen der Zielmaschine

# Einfach: Hennessy & Patterson DLX

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

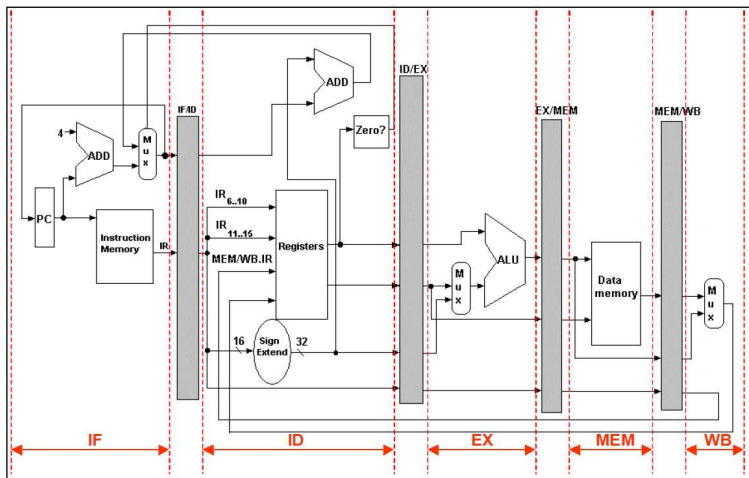
Syntax

Triangle

Kontext

Semantik

Zusammenfassung



# Komplizierter: Analog Devices TigerSHARC

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

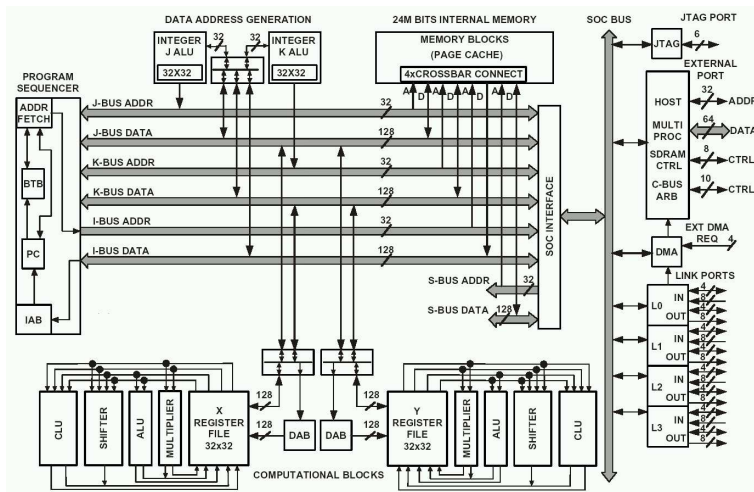
Syntax

Triangle

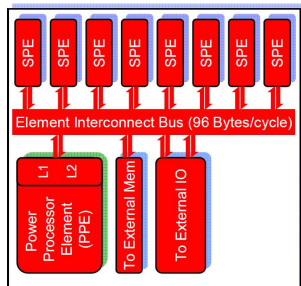
Kontext

Semantik

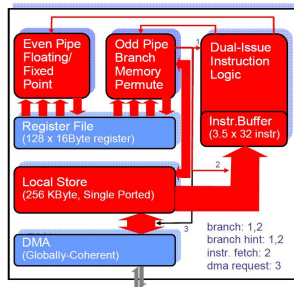
Zusammenfass...



## Übersicht



## SPE





- Rechenleistung (hoch/niedrig)
- Datentypen (Gleitkomma, ganzzahlig, Vektoren)
- Operationen (Multiplikationen, MACs)
- Speicherbandbreite (parallele Speicherzugriffe)
- Energieeffizienz
- Platzbedarf

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Rechenleistung (hoch/niedrig)
- Datentypen (Gleitkomma, ganzzahlig, Vektoren)
- Operationen (Multiplikationen, MACs)
- Speicherbandbreite (parallele Speicherzugriffe)
- Energieeffizienz
- Platzbedarf

... können häufig nur durch spezialisierte Prozessoren erfüllt werden

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Rechenleistung (hoch/niedrig)
- Datentypen (Gleitkomma, ganzzahlig, Vektoren)
- Operationen (Multiplikationen, MACs)
- Speicherbandbreite (parallele Speicherzugriffe)
- Energieeffizienz
- Platzbedarf

... können häufig nur durch spezialisierte Prozessoren erfüllt werden

➔ **Benötigen passende Compiler**

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

DOI:10.1145/1461928.1461946

**Research and education in compiler technology is more important than ever.**

BY MARY HALL, DAVID PADUA, AND KESHAV PINGALI

## Compiler Research: The Next 50 Years

*Communications of the Association for Computing Machinery (CACM)*, Februar 2009

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

- Taktfrequenz von Prozessoren nur mit Mühe steigerbar
- Trend weg von hochgetakteten Einzelprozessoren
- ... hin zu vielen (aber langsameren) Prozessoren
- Wie parallele Strukturen programmieren?
- Erste praktische Ansätze
  - OpenMP: Mehr-Kern-CPU's
  - NVidia CUDA: GPU's (→ PMPP Michael Gösele)
  - OpenCL: Heterogene Systeme (GPU's+CPU's)
- Aber noch wenig abstrakt
- *Keine* automatische Parallelisierung!

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

## Apple Computer

**Job Title** Sr Compiler Engineer

**Posting Date** Siemens

**Job Location**

**Description**

**Job Title** Principal Compiler Engineer

**Posting Date** Sony Computer Entertainment

**Job Location**

**Description**

**Job Title** Compiler Engineer

**Posting Date**

**Job Location**

**Description**

## RWTH University

**Job Title** Research assistant/PhD candidate

**Posting Date** 11/15/2005

**Job Location** Aachen, Germany

**Description** The SSS division performs research and development on electronic design automation tools for embedded systems, e.g. future telecommunication and multimedia systems. Major research areas include compilers for

Offene Stellen auf [www.compilerjobs.com](http://www.compilerjobs.com) bis 01-04/2011: 40

C1

A. Koch

Einleitung

Zielmaschine

**Aufbau**

Optimierung

Orga

Syntax

Triangle

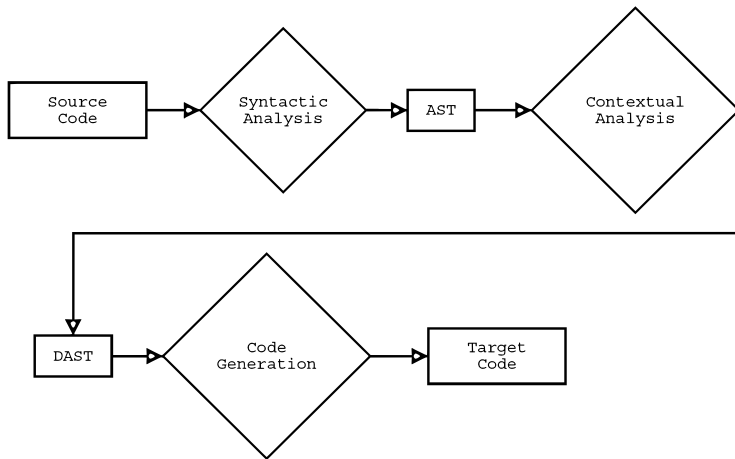
Kontext

Semantik

Zusammenfassu

# Aufbau von Compilern

# Vorgehen: Bearbeitung in mehreren Phasen



Zwischendarstellung(en) für den Informationsaustausch

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

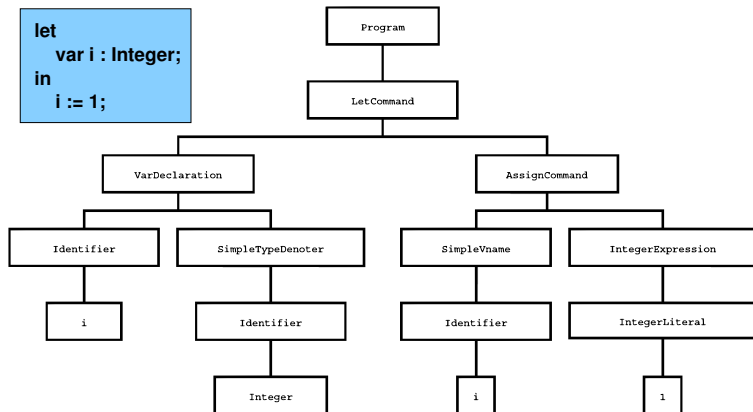
Kontext

Semantik

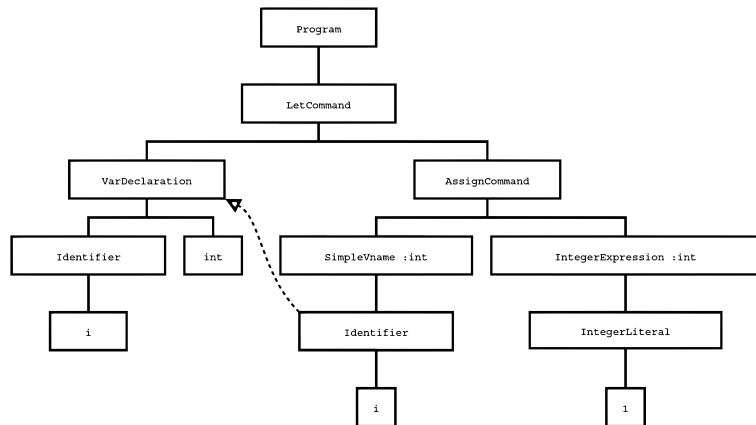
Zusammenfassung



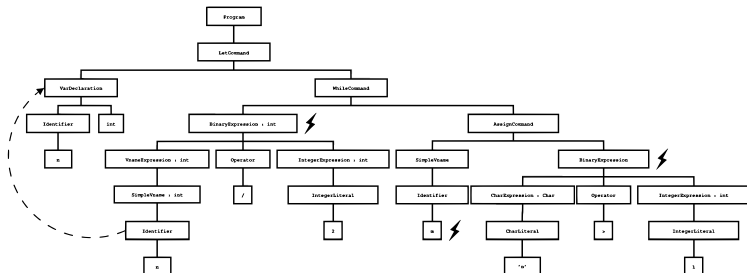
- Überprüfung ob Programm Syntaxregeln gehorcht
- Speichern des Programmes in geeigneter Darstellung



- Ordne Variablen ihren Deklarationen zu
- Berechne Typen von Ausdrücken



## Erkenne Fehler in Variablen- und Typzuordnung



Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Programm ist syntaktisch und kontextuell korrekt

C1

A. Koch

Einleitung

Zielmaschine

**Aufbau**

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Programm ist syntaktisch und kontextuell korrekt
- Übersetzung in Zielsprache
  - Maschinensprache
  - Assembler
  - C
  - Andere Hochsprache

C1

A. Koch

Einleitung

Zielmaschine

**Aufbau**

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Programm ist syntaktisch und kontextuell korrekt
- Übersetzung in Zielsprache
  - Maschinensprache
  - Assembler
  - C
  - Andere Hochsprache
- Ordne DAST-Teilen Instruktionen der Zielsprache zu

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Programm ist syntaktisch und kontextuell korrekt
- Übersetzung in Zielsprache
  - Maschinensprache
  - Assembler
  - C
  - Andere Hochsprache
- Ordne DAST-Teilen Instruktionen der Zielsprache zu
- Handhabung von Variablen
  - 1 Deklaration: Reserviere Speicherplatz für eine Variable
  - 2 Verwendung: Referenziere immer den zugeordneten Speicherplatz

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

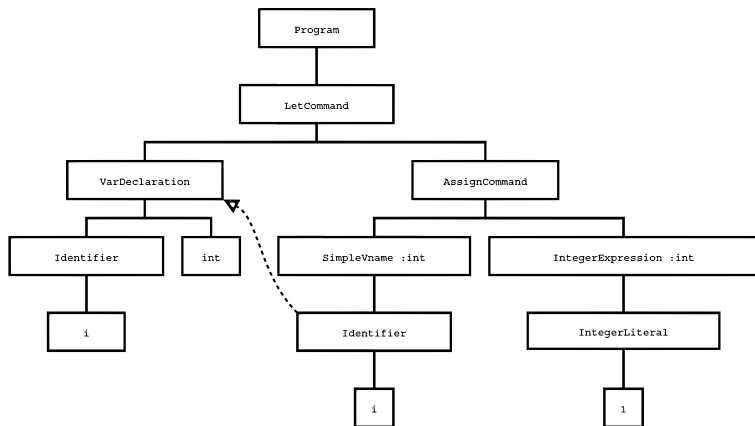
Syntax

Triangle

Kontext

Semantik

Zusammenfassung



```
0: PUSH      1      ; Platz fuer 'i' schaffen, Adresse 0[SB]
1: LOADL    1      ; Wert 1 auf Stack legen
2: STORE (1) 0[SB] ; ein Datenwort vom Stack nach Adresse 0[SB] schreiben
3: POP (0)  1      ; Platz von 'i' wieder aufgeben
4: HALT
```



C1

A. Koch

Einleitung

Zielmaschine

Aufbau

**Optimierung**

Orga

Syntax

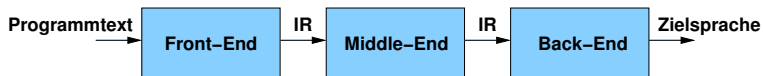
Triangle

Kontext

Semantik

Zusammenfassu

# Optimierung



- Front-End: Syntaktische/kontextuelle Analyse

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

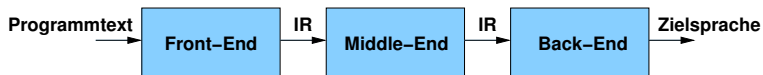
Syntax

Triangle

Kontext

Semantik

Zusammenfassung



- Front-End: Syntaktische/kontextuelle Analyse
- Back-End: Code-Erzeugung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

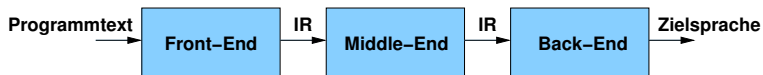
Syntax

Triangle

Kontext

Semantik

Zusammenfassung



- Front-End: Syntaktische/kontextuelle Analyse
- Back-End: Code-Erzeugung
- **Middle-End**: Transformation von Zwischendarstellungen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

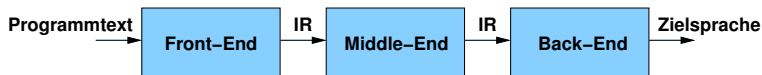
Kontext

Semantik

Zusammenfassung



- Front-End: Syntaktische/kontextuelle Analyse
- Back-End: Code-Erzeugung
- **Middle-End**: Transformation von Zwischendarstellungen
  - Intermediate Representation (IR)
  - Keine direkte Code-Erzeugung aus Front-End IR
  - Verwendet in der Regel zusätzliche interne Darstellungen



- Front-End: Syntaktische/kontextuelle Analyse
- Back-End: Code-Erzeugung
- **Middle-End**: Transformation von Zwischendarstellungen
  - Intermediate Representation (IR)
  - Keine direkte Code-Erzeugung aus Front-End IR
  - Verwendet in der Regel zusätzliche interne Darstellungen

**Ziel:** Verbesserung des erzeugten Codes in Bezug auf bestimmte Gütemaße

## Constant-Folding

$$x = (2+3) * y$$

$$x = 5*y$$

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

## Constant-Folding

$$x = (2+3) * y$$

$$x = 5*y$$

## Common-Subexpression Elimination

$$x = 5 * a + b;$$

$$y = 5 * a + c;$$

$$t = 5 * a;$$

$$x = t + b;$$

$$y = t + c;$$

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung



## Constant-Folding

$x = (2+3) * y$

$x = 5*y$

## Common-Subexpression Elimination

```
x = 5 * a + b;  
y = 5 * a + c;
```

```
t = 5 * a;  
x = t + b;  
y = t + c;
```

## Strength Reduction

```
for (i=0; i <= j; ++i) {  
    a[i*3] = 42;  
}
```

```
int t = 0;  
for (i=0; i <= j; ++i) {  
    a[t] = 42;  
    t = t + 3;  
}
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

## Constant-Folding

$x = (2+3) * y$

$x = 5*y$

## Common-Subexpression Elimination

```
x = 5 * a + b;  
y = 5 * a + c;
```

```
t = 5 * a;  
x = t + b;  
y = t + c;
```

## Strength Reduction

```
for (i=0; i <= j; ++i) {  
    a[i*3] = 42;  
}
```

```
int t = 0;  
for (i=0; i <= j; ++i) {  
    a[t] = 42;  
    t = t + 3;  
}
```

## Loop-invariant Code Motion

```
int t;  
for (i=0; i <= j; ++i) {  
    t = x * y;  
    a[i] = t * i;  
}
```

```
int t = x * y;  
for (i=0; i <= j; ++i) {  
    a[i] = t * i;  
}
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

# Organisatorisches

## Dozent

Andreas Koch  
Sprechstunde

koch@esa.informatik.tu-darmstadt.de  
i.d.R. Mi 14:00-15:00, E103

## Web-Seite

<http://www.esa.cs.tu-darmstadt.de>

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Grundlagen von Compilern: **Fast vollständig**

**Programming Language Processors in Java**  
von David Watt und Deryck Brown, Prentice-Hall 2000

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Grundlagen von Compilern: **Fast vollständig**

## **Programming Language Processors in Java**

von David Watt und Deryck Brown, Prentice-Hall 2000

Weiterführende Themen: **Auszugsweise**

## **Programming Language Pragmatics**

von Michael Scott, Morgan Kaufmann 2009

Guter allgemeiner Überblick, aber im Detail mit anderen Schwerpunkten als bei uns

## **Compilers, 2. Auflage (!)**

von Aho, Sethi, Ullmann, Lam, Addison-Wesley 2006

Auch auf Deutsch verfügbar

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

- Vorlesung
  - Theoretische und algorithmische Grundlagen
- Semesterbegleitende Prüfung gegen Ende der Vorlesungszeit
  - Vermutlich Klausur, gegen Ende der Vorlesungszeit
- Optionale Hausaufgaben
  - Zur Verbesserung der Note um bis zu +0,7
  - **Nicht** aber zum Anheben über die Bestehensgrenze

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung



## Vorläufige Planung

- Gruppengröße: Vermutlich Zweiergruppen
- Geplant: Nach jedem Abschnitt der Vorlesung ein Aufgabenblatt (6-7 Blätter)
- Anmeldung zu den Aufgaben in der Vorlesung
- Verständnisfragen und kleine (ca. 20-100 Zeilen) Programmieraufgaben
- 1. Aufgabenblatt zum Thema Lexing/Parsing erwartet ab 3. VL Woche

Festlegung der endgültigen Modalitäten erst in der 2. VL Woche

## Überblick über Front-End<sup>1</sup> (ca. 3 Wochen)

- Lexing/Parsing
- Zwischendarstellungen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

## Überblick über Front-End<sup>1</sup> (ca. 3 Wochen)

- Lexing/Parsing
- Zwischendarstellungem

## Überblick über Middle-End<sup>1</sup> (ca. 2 Wochen)

- Semantische / Kontextanalyse

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

## Überblick über Front-End<sup>1</sup> (ca. 3 Wochen)

- Lexing/Parsing
- Zwischendarstellungem

## Überblick über Middle-End<sup>1</sup> (ca. 2 Wochen)

- Semantische / Kontextanalyse

## Überblick über Back-End<sup>1</sup> (ca. 4 Wochen)

- Laufzeitorganisation
- Code-Erzeugung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

## Überblick über Front-End<sup>1</sup> (ca. 3 Wochen)

- Lexing/Parsing
- Zwischendarstellungem

## Überblick über Middle-End<sup>1</sup> (ca. 2 Wochen)

- Semantische / Kontextanalyse

## Überblick über Back-End<sup>1</sup> (ca. 4 Wochen)

- Laufzeitorganisation
- Code-Erzeugung

<sup>1</sup>: Diese Teile lehnen sich an an die Veranstaltungen

- IMT3052 von Ivar Farup, Universität Gjøvik, Norwegen
- Vertalerbouw von Theo Ruys, Universität Twente, Niederlande

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

## Weiterführende Themen

- Aufbau von Interpretern (ca. 1-2 Wochen)
- Verwendung von Front-End-Generatoren (ca. 1-2 Wochen)
- Entwurfskonzepte von Programmiersprachen (ca. 2-3 Wochen)

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

**Syntax**

Triangle

Kontext

Semantik

Zusammenfassu

# Syntax

Beschreibt die Satzstruktur von korrekten Programmen

- $n := n + 1;$

Syntaktisch korrektes Statement in Triangle

- “Ein Kreis hat zwei Ecken.”

**Syntaktisch** korrekte Aussage in Deutsch



Dazu gehören Regeln für den Geltungsbereich (*scope*) und den Typ von Aussagen.

- $n$  muß bei Auftreten des Statements passend deklariert sein.
- Kreise haben im allgemeinen keine Ecken. Hier passen die Typen offenbar nicht.

Die Bedeutung einer Anweisung/Aussage in einer Sprache.  
Wird bei Programmiersprachen häufig beschrieben ...

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

**Syntax**

Triangle

Kontext

Semantik

Zusammenfassu

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Die Bedeutung einer Anweisung/Aussage in einer Sprache.  
Wird bei Programmiersprachen häufig beschrieben ...

**Operationell** Welche Schritte laufen ab, wenn das  
Programm gestartet wird?

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Die Bedeutung einer Anweisung/Aussage in einer Sprache.  
Wird bei Programmiersprachen häufig beschrieben ...

**Operationell** Welche Schritte laufen ab, wenn das  
Programm gestartet wird?

**Denotational** Abbildung von Eingaben auf Ausgaben

- Für alle drei Teile
  - 1 Syntax
  - 2 Kontextuelle Einschränkungen
  - 3 Semantik

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

**Syntax**

Triangle

Kontext

Semantik

Zusammenfassu

- Für alle drei Teile
  - 1 Syntax
  - 2 Kontextuelle Einschränkungen
  - 3 Semantik
- ... gibt es jeweils zwei Spezifikationsarten
  - Formal
  - Informal

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

- Für alle drei Teile
  - 1 Syntax
  - 2 Kontextuelle Einschränkungen
  - 3 Semantik
- ... gibt es jeweils zwei Spezifikationsarten
  - Formal
  - Informal

## Triangle-Spezifikation

- Formale Syntax (reguläre Ausdrücke, EBNF)
- Informale kontextuelle Einschränkungen
- Informale Semantik

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Eine **Sprache** ist eine Menge von **Zeichenketten** aus einem **Alphabet**

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

**Syntax**

Triangle

Kontext

Semantik

Zusammenfassu



Eine **Sprache** ist eine Menge von **Zeichenketten** aus einem **Alphabet**

- Wie diese Menge angeben?
- Bei endlichen Sprachen: Einfach Elemente aufzählen
- Geht nicht bei unendlichen Sprachen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Eine **Sprache** ist eine Menge von **Zeichenketten** aus einem **Alphabet**

- Wie diese Menge angeben?
- Bei endlichen Sprachen: Einfach Elemente aufzählen
- Geht nicht bei unendlichen Sprachen
- Mögliche Vorgehensweisen
  - 1 Mathematische Mengennotation
  - 2 Reguläre Ausdrücke
  - 3 Kontextfreie Grammatik

## Beispiele für die beschriebenen Zeichenketten

- $L = \{\mathbf{a, b, c}\}$  beschreibt **a, b, c**
- $L = \{\mathbf{x}^n | n > 0\}$  beschreibt **x, xx, xxx, ...**
- $L = \{\mathbf{x}^n \mathbf{y}^m | n > 0, m > 0\}$  beschreibt **xy, xyy, xxxyy, ...**
- $L = \{\mathbf{x}^n \mathbf{y}^n | n > 0\}$  beschreibt **xy, xxyy, ...**, aber z.B. nicht **xy**

## Beispiele für die beschriebenen Zeichenketten

- $L = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$  beschreibt **a**, **b**, **c**
- $L = \{\mathbf{x}^n \mid n > 0\}$  beschreibt **x**, **xx**, **xxx**, ...
- $L = \{\mathbf{x}^n \mathbf{y}^m \mid n > 0, m > 0\}$  beschreibt **xy**, **xyy**, **xxxxyy**, ...
- $L = \{\mathbf{x}^n \mathbf{y}^n \mid n > 0\}$  beschreibt **xy**, **xxxxyy**, ..., aber z.B. nicht **xyy**

Offensichtlich keine sonderlich nützliche und gut zu handhabende Spezifikationsform für komplexere Sprachen.

## Erweitere Zeichenketten aus dem Alphabet um Operatoren

- | zeigt Alternativen an
- \* zeigt Null oder mehr Vorkommen des vorangehenden Zeichens an
- $\varepsilon$  ist die leere Zeichenkette
- ( ... ) erlauben die Gruppierung von Teilausdrücken durch Klammerung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

## Erweitere Zeichenketten aus dem Alphabet um Operatoren

- | zeigt Alternativen an
- \* zeigt Null oder mehr Vorkommen des vorangehenden Zeichens an
- $\varepsilon$  ist die leere Zeichenkette
- ( ... ) erlauben die Gruppierung von Teilausdrücken durch Klammerung

## Beispiele

- $L = \mathbf{a|b|c}$  ergibt **a, b, c**
- $L = \mathbf{ab^*}$  ergibt **a, ab, abb, ...**
- $L = (\mathbf{ab})^*$  ergibt die leere Zeichenkette  $\varepsilon$ , **ab, abab, ababab, ...**
- $L = \mathbf{a(b|\varepsilon)}$  ergibt **a oder ab**

Kann man die Menge aller regulären Ausdrücke selber durch einen regulären Ausdruck beschreiben?

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

**Syntax**

Triangle

Kontext

Semantik

Zusammenfassung

Kann man die Menge aller regulären Ausdrücke selber durch einen regulären Ausdruck beschreiben?

Nein! REs sind daher ungeeignet zur Beschreibung der Syntax komplexer Programmiersprachen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung



Kann man die Menge aller regulären Ausdrücke selber durch einen regulären Ausdruck beschreiben?

Nein! REs sind daher ungeeignet zur Beschreibung der Syntax komplexer Programmiersprachen

... also Weitersuchen nach geeigneter Beschreibungsform für Syntax

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Kann man die Menge aller regulären Ausdrücke selber durch einen regulären Ausdruck beschreiben?

Nein! REs sind daher ungeeignet zur Beschreibung der Syntax komplexer Programmiersprachen

... also Weitersuchen nach geeigneter Beschreibungsform für Syntax

Aber: REs sind trotzdem innerhalb eines Compilers nützlich (siehe PLPJ Kapitel 4, Scanner).

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Eine kontextfreie Grammatik besteht aus

- Einer Menge von Terminalsymbolen  $T$  aus dem Alphabet
- Einer Menge von Nicht-Terminalsymbolen  $N$
- Einem Startsymbol  $S \in N$
- Einer Menge von Produktionen  $P$ 
  - Beschreiben, wie Nicht-Terminalsymbole aus Terminalsymbolen zusammengesetzt sind.

## Produktionen in Backus-Naur-Form (BNF)

Nicht-Terminal ::= **Zeichenkette** aus Terminal und Nicht-Terminalsymbolen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

**Syntax**

Triangle

Kontext

Semantik

Zusammenfassung

## Produktionen in Backus-Naur-Form (BNF)

Nicht-Terminal ::= **Zeichenkette** aus Terminal und Nicht-Terminalsymbolen

## Produktionen in Extended BNF (EBNF)

Nicht-Terminal ::= **RE** aus Terminal und Nicht-Terminalsymbolen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

$$T = \{\mathbf{x}, \mathbf{y}\}, N = \{\mathbf{S}, \mathbf{B}\}, S = \mathbf{S}, P = \{$$
$$\mathbf{S} ::= \mathbf{xS} \quad (1)$$
$$\mathbf{S} ::= \mathbf{yB} \quad (2)$$
$$\mathbf{S} ::= \mathbf{x} \quad (3)$$
$$\mathbf{B} ::= \mathbf{yB} \quad (4)$$
$$\mathbf{B} ::= \mathbf{y} \quad \} \quad (5)$$

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

$T = \{\mathbf{x}, \mathbf{y}\}$ ,  $N = \{\mathbf{S}, \mathbf{B}\}$ ,  $S = \mathbf{S}$ ,  $P = \{$

$\mathbf{S} ::= \mathbf{xS}$  (1)

$\mathbf{S} ::= \mathbf{yB}$  (2)

$\mathbf{S} ::= \mathbf{x}$  (3)

$\mathbf{B} ::= \mathbf{yB}$  (4)

$\mathbf{B} ::= \mathbf{y}$  } (5)

Ist die Zeichenkette **xyyy** Element der durch  $T, N, S, P$  beschriebenen Sprache?

$T = \{\mathbf{x}, \mathbf{y}\}, N = \{\mathbf{S}, \mathbf{B}\}, S = \mathbf{S}, P = \{$

$\mathbf{S} ::= \mathbf{xS} \quad (1)$

$\mathbf{S} ::= \mathbf{yB} \quad (2)$

$\mathbf{S} ::= \mathbf{x} \quad (3)$

$\mathbf{B} ::= \mathbf{yB} \quad (4)$

$\mathbf{B} ::= \mathbf{y} \quad \}$   $(5)$

Ist die Zeichenkette **xyyy** Element der durch  $T, N, S, P$  beschriebenen Sprache?

$\mathbf{S} \rightarrow \mathbf{xS} \rightarrow \mathbf{xxS} \rightarrow \mathbf{xyB} \rightarrow \mathbf{xyyB} \rightarrow \mathbf{xyyy}$

➡ Ja, da sie sich aus  $S$  herleiten läßt.



$T = \{\mathbf{x}, \mathbf{y}\}, N = \{\mathbf{S}, \mathbf{B}\}, S = \mathbf{S}, P = \{$

$\mathbf{S} ::= \mathbf{xS} \quad (6)$

$\mathbf{S} ::= \mathbf{yB} \quad (7)$

$\mathbf{S} ::= \mathbf{x} \quad (8)$

$\mathbf{B} ::= \mathbf{yB} \quad (9)$

$\mathbf{B} ::= \mathbf{y} \quad \} \quad (10)$

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

$T = \{\mathbf{x}, \mathbf{y}\}, N = \{\mathbf{S}, \mathbf{B}\}, S = \mathbf{S}, P = \{$

$\mathbf{S} ::= \mathbf{xS} \quad (6)$

$\mathbf{S} ::= \mathbf{yB} \quad (7)$

$\mathbf{S} ::= \mathbf{x} \quad (8)$

$\mathbf{B} ::= \mathbf{yB} \quad (9)$

$\mathbf{B} ::= \mathbf{y} \quad \} \quad (10)$

Ist die Zeichenkette **xy** Element der durch  $T, N, S, P$  beschriebenen Sprache?

$T = \{\mathbf{x}, \mathbf{y}\}, N = \{\mathbf{S}, \mathbf{B}\}, S = \mathbf{S}, P = \{$

$\mathbf{S} ::= \mathbf{xS} \quad (6)$

$\mathbf{S} ::= \mathbf{yB} \quad (7)$

$\mathbf{S} ::= \mathbf{x} \quad (8)$

$\mathbf{B} ::= \mathbf{yB} \quad (9)$

$\mathbf{B} ::= \mathbf{y} \quad \} \quad (10)$

Ist die Zeichenkette **xy** Element der durch  $T, N, S, P$  beschriebenen Sprache?

$\mathbf{S} \rightarrow \mathbf{xS} \rightarrow \mathbf{xyB} \rightarrow ?$

➡ Nein, da sie sich nicht aus  $S$  herleiten läßt.

Gegeben seien die Produktionen:

$$\mathbf{S} ::= \mathbf{S+S} \quad (11)$$

$$\mathbf{S} ::= \mathbf{x} \quad (12)$$

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Gegeben seien die Produktionen:

$$\mathbf{S} ::= \mathbf{S+S} \quad (11)$$

$$\mathbf{S} ::= \mathbf{x} \quad (12)$$

Wie läßt sich die Zeichenkette  $\mathbf{x+x+x}$  herleiten?

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Gegeben seien die Produktionen:

$$\mathbf{S} ::= \mathbf{S+S} \quad (11)$$

$$\mathbf{S} ::= \mathbf{x} \quad (12)$$

Wie läßt sich die Zeichenkette  $\mathbf{x+x+x}$  herleiten?

$$\mathbf{S} \rightarrow \mathbf{S+S} \rightarrow \mathbf{x+S} \rightarrow \mathbf{x+S+S} \rightarrow \mathbf{x+x+S} \rightarrow \mathbf{x+x+x}$$

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Gegeben seien die Produktionen:

$$\mathbf{S} ::= \mathbf{S+S} \quad (11)$$

$$\mathbf{S} ::= \mathbf{x} \quad (12)$$

Wie läßt sich die Zeichenkette  $\mathbf{x+x+x}$  herleiten?

$$\mathbf{S} \rightarrow \mathbf{S+S} \rightarrow \mathbf{x+S} \rightarrow \mathbf{x+S+S} \rightarrow \mathbf{x+x+S} \rightarrow \mathbf{x+x+x}$$

Aber auch anders:

$$\mathbf{S} \rightarrow \mathbf{S+S} \rightarrow \mathbf{S+x} \rightarrow \mathbf{S+S+x} \rightarrow \mathbf{S+x+x} \rightarrow \mathbf{x+x+x}$$

Für sinnvolle praktische Anwendungen müssen CFGs **eindeutig** sein.

Eindeutige Produktionen für die gleiche CFG:

$$\mathbf{S} ::= \mathbf{x+S} \quad (13)$$

$$\mathbf{S} ::= \mathbf{x} \quad (14)$$



# (Mini-) Triangle

- Pascal-artige Sprache als Anschauungsobjekt
- Compiler-Quellcode auf Web-Page
- In der Vorlesung: Mini-Triangle
  - Weiter vereinfachte Version
  - Z.B. keine Definition von Unterfunktionen

```
let
  const MAX ~ 10;
  var n: Integer
in begin
  getint(var n);
  if (n>0) /\ (n<=MAX) then
    while n > 0 do begin
      putint(n); puteol();
      n := n-1
    end
  else
  end
end
```

Lokale Deklarationen

Konstante (hässliches "~")

Variable kann in `getint` verändert werden

Folge von Anweisungen zwischen `begin/end`

else ist erforderlich (darf aber leer sein)

```
Program ::= single-Command
single-Command ::= empty
                | V-name := Expression
                | Identifier ( Expression )
                | if Expression then single-Command
                  else single-Command
                | while Expression do single-Command
                | let Declaration in single-Command
                | begin Command end
Command ::= single-Command
          | Command ; single-Command
...

```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

```
Expression
 ::= primary-Expression
    | Expression Operator primary-Expression
primary-Expression
 ::= Integer-Literal
    | V-name
    | Operator primary-Expression
    | ( Expression )
V-name ::= Identifier
Identifier ::= Letter
           | Identifier Letter
           | Identifier Digit
Integer-Literal ::= Digit
                | Integer-Literal Digit
Operator ::= + | - | * | / | < | > | =
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

```
Declaration
 ::= single-Declaration
    | Declaration ; single-Declaration
single-Declaration
 ::= const Identifier ~ Expression
    | var Identifier : Type-denoter
Type-denoter ::= Identifier
```

```
Comment ::= ! CommentLine eol
CommentLine ::= Graphic CommentLine
Graphic ::= any printable character or space
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

**Triangle**

Kontext

Semantik

Zusammenfassu

**Phrase** Von einem gegebenen Nicht-Terminalsymbol herleitbare Kette von Terminalsymbolen.  
Z.B. Expression-Phrase, Command-Phrase ...

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

**Triangle**

Kontext

Semantik

Zusammenfassung



- Phrase** Von einem gegebenen Nicht-Terminalsymbol herleitbare Kette von Terminalsymbolen.  
Z.B. Expression-Phrase, Command-Phrase ...
- Satz** S-Phrase, wobei S das Startsymbol der CFG ist

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

**Triangle**

Kontext

Semantik

Zusammenfassu

**Phrase** Von einem gegebenen Nicht-Terminalsymbol herleitbare Kette von Terminalsymbolen.  
Z.B. Expression-Phrase, Command-Phrase ...

**Satz** S-Phrase, wobei S das Startsymbol der CFG ist

Beispiel:

```
let           (1)
  var y : Integer (2)
in           (3)
  y := y + 1   (4)
```

- Das gesamte Program ist ein *Satz* der CFG

**Phrase** Von einem gegebenen Nicht-Terminalsymbol herleitbare Kette von Terminalsymbolen.  
Z.B. Expression-Phrase, Command-Phrase ...

**Satz** S-Phrase, wobei S das Startsymbol der CFG ist

Beispiel:

```
let           (1)
  var y : Integer (2)
in           (3)
  y := y + 1   (4)
```

- Das gesamte Program ist ein *Satz* der CFG
- Zeile 2 ist eine single-Declaration-Phrase

**Phrase** Von einem gegebenen Nicht-Terminalsymbol herleitbare Kette von Terminalsymbolen.  
Z.B. Expression-Phrase, Command-Phrase ...

**Satz** S-Phrase, wobei S das Startsymbol der CFG ist

Beispiel:

```
let                (1)
  var y : Integer (2)
in                 (3)
  y := y + 1      (4)
```

- Das gesamte Program ist ein *Satz* der CFG
- Zeile 2 ist eine single-Declaration-Phrase
- Zeile 4 ist eine single-Command-Phrase

Ein Syntaxbaum ist ein geordneter, markierter Baum bei dem

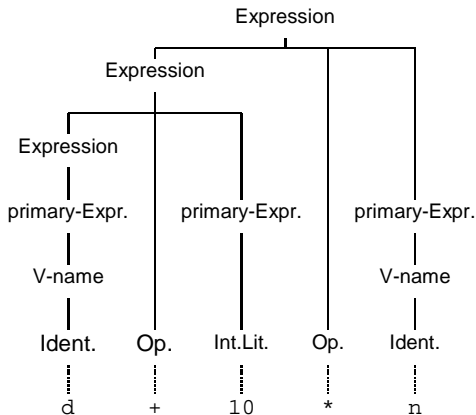
- ... die Blätter mit Terminalsymbolen markiert sind
- ... die inneren Knoten mit Nicht-Terminalsymbolen markiert sind
- ... jeder innere Knoten **N** (von links nach rechts) die Kinder  $\mathbf{X}_1, \dots, \mathbf{X}_n$  hat, entsprechend der Produktion  $\mathbf{N} := \mathbf{X}_1 \dots \mathbf{X}_n$

Ein Syntaxbaum ist ein geordneter, markierter Baum bei dem

- ... die Blätter mit Terminalsymbolen markiert sind
- ... die inneren Knoten mit Nicht-Terminalsymbolen markiert sind
- ... jeder innere Knoten **N** (von links nach rechts) die Kinder  $\mathbf{X}_1, \dots, \mathbf{X}_n$  hat, entsprechend der Produktion  $\mathbf{N} := \mathbf{X}_1 \dots \mathbf{X}_n$

Ein  $N$ -Baum ist ein Baum mit einem  $N$  Nicht-Terminalsymbol am Wurzelknoten.

# Expression-Baum für $d + 10 * n$



C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

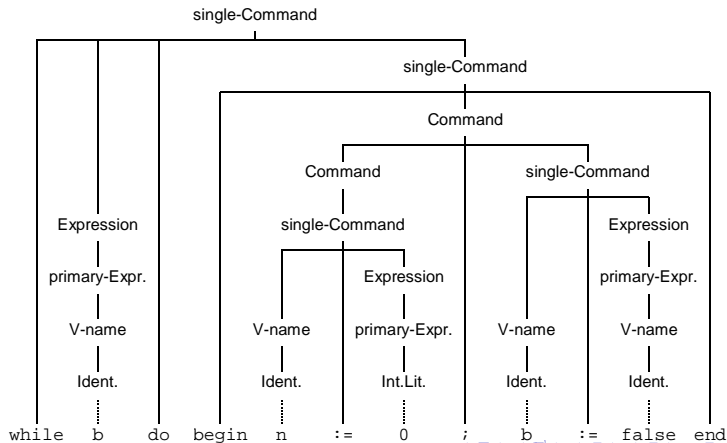
Triangle

Kontext

Semantik

Zusammenfassung

```
while b do begin
  n := 0;
  b := false
end
```



C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung



- Grammatik spezifiziert präzise syntaktische Details
    - **do**, **:=**, ...
- ➔ Konkrete Syntax, wichtig für das Verfassen korrekter Programme

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

**Triangle**

Kontext

Semantik

Zusammenfassu

- Grammatik spezifiziert präzise syntaktische Details
  - **do**, **:=**, ...
- ↳ Konkrete Syntax, wichtig für das Verfassen korrekter Programme
- Konkrete Syntax hat aber *keinen* Einfluß auf Semantik der Programme
  - **V = E**
  - **V := E**
  - **set V = E**
  - **assign E to V**
  - **V ← E**
- ... können alle das gleiche bedeuten: Eine Zuweisung von **E** nach **V**

- Grammatik spezifiziert präzise syntaktische Details
  - **do**, **:=**, ...

➔ Konkrete Syntax, wichtig für das Verfassen korrekter Programme

- Konkrete Syntax hat aber *keinen* Einfluß auf Semantik der Programme
  - **V = E**
  - **V := E**
  - **set V = E**
  - **assign E to V**
  - **V ← E**
- ... können alle das gleiche bedeuten: Eine Zuweisung von **E** nach **V**

➔ Für weitere Verarbeitung Darstellung vereinfachen!

- Modelliert nur **essentielle Information**
- Idee: Orientierung an der Subphrasen-Struktur der Produktionen
- Beispiel: **V-name := Expression** hat zwei Subphrasen
  - 1 **V-name**
  - 2 **Expression**

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

- Schlüsselworte, Begrenzer wie **do**, **:=** sind irrelevant
- Unterscheidungen zwischen
  - **Command** und **single-Command**
  - **Declaration** und **single-Declaration**
  - **Expression** und **primary-Expression**
- ..... sind nur für das Erkennen des Programmes relevant, nicht zur Darstellung seiner Semantik.

- Schlüsselworte, Begrenzer wie **do**, **:=** sind irrelevant
- Unterscheidungen zwischen
  - **Command** und **single-Command**
  - **Declaration** und **single-Declaration**
  - **Expression** und **primary-Expression**
- ..... sind nur für das Erkennen des Programmes relevant, nicht zur Darstellung seiner Semantik.

➡ Alle dafür unwichtigen Details weglassen!

## Command

<code>::= V-name := Expression</code>	<i>AssignCmd</i>
Identifier ( Expression )	<i>CallCmd</i>
<b>if</b> Expression <b>then</b> Command	
<b>else</b> Command	<i>IfCmd</i>
<b>while</b> Expression <b>do</b> Command	<i>WhileCmd</i>
<b>let</b> Declaration <b>in</b> Command	<i>LetCmd</i>
Command ; Command	<i>SequentialCmd</i>

Expression

::= Integer-Literal

| V-name

| Operator Expression

| Expression Op Expression

V-name ::= Identifier

*IntegerExp*

*VnameExp*

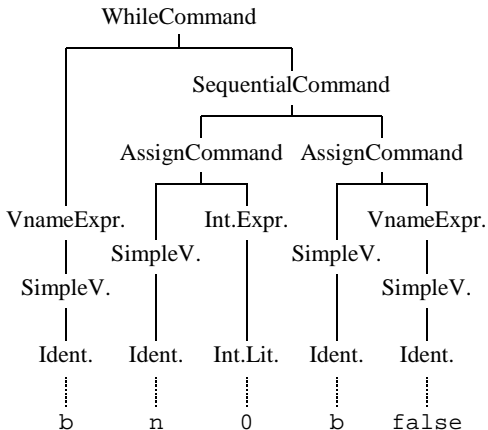
*UnaryExp*

*BinaryExp*

*SimpleVName*



```
while b do begin
  n := 0;
  b := false
end
```



- AST ist eine weit verbreitete Form der IR
- High-level IR
- Sehr nah an der Eingabesprache
- Gut geeignet für weitreichende Analysen und Transformationen
  - Unabhängig von Architektur der Zielmaschine
  - Verschieben von Anweisungen
  - Änderungen der Programmstruktur

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Schlechter geeignet für maschinennahe Analysen und Transformationen

- Ausnutzung von Maschinenregistern
- Ausnutzung von speziellen Maschinenbefehlsfolgen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

**Triangle**

Kontext

Semantik

Zusammenfassung

Schlechter geeignet für maschinennahe Analysen und Transformationen

- Ausnutzung von Maschinenregistern
- Ausnutzung von speziellen Maschinenbefehlsfolgen

➔ Hier: Konzentration auf **maschinunenabhängige** Ebene

- (D)AST ist Hauptrepräsentation
- Für einzelne Bearbeitungsschritte: Andere IRs

# Kontextuelle Einschränkungen

# Kontextuelle Einschränkungen: Geltungsbereiche

Syntaktische Korrektheit reicht nicht aus für sinnvolle  
Übersetzung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

**Kontext**

Semantik

Zusammenfassu

Syntaktische Korrektheit reicht nicht aus für sinnvolle  
Übersetzung

## Geltungsbereiche (Scope)

- Betreffen *Sichtbarkeit* von Bezeichnern
- Jeder verwendete Bezeichner muss vorher *deklariert* werden
  - ... nicht bei allen Programmiersprachen
- Deklaration ist sog. *bindendes Auftreten* des Bezeichners
- Benutzung ist sog. *verwendendes Auftreten* des Bezeichners
- Aufgabe: Bringe jede Verwendung mit genau der einen passenden Bindung in Zusammenhang

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

```
let
  const m ~ 2;
  var n: Integer
in begin
  ..
  n := m*2;
  ..
end
```

Deklaration von n:  
Bindung

Benutzung von von n:  
Verwendung

??

```
let
  var n: Integer
in begin
  ..
  n := m*2;
end
```

Falls im Geltungsbereich der  
Verwendung vom m keine Bindung  
von m existiert: Fehler!

Verwendung von m



## Typen

- Jeder Wert hat einen Typ
- Jede Operation
  - ... hat Anforderungen an die Typen der Operanden
  - ... hat Regeln für den Typ des Ergebnisses

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

**Kontext**

Semantik

Zusammenfassu

## Typen

- Jeder Wert hat einen Typ
- Jede Operation
  - ... hat Anforderungen an die Typen der Operanden
  - ... hat Regeln für den Typ des Ergebnisses

... auch nicht bei allen Programmiersprachen.

- Hier: statische Typisierung (zur Compile-Zeit)
- Alternativ: dynamische Typisierung (zur Laufzeit)

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

```
let
  var n: Integer
in begin
  ...
  while n > 0 do
    n := n-1;
  ...
end
```

Typregel für  $E_1 > E_2$  (GreaterOp):  
Wenn  $E_1$  und  $E_2$  beide vom Typ **int**,  
dann ist **Ergebnis** vom Typ **bool**.

Typregel für **while** E **do** C (WhileCmd):  
**E** muss vom Typ **boolean** sein.

Typregel für  $V := E$  (AssignCmd):  
Die Typen von **V** und **E** müssen  
**äquivalent** sein.

Typregel für  $E_1 - E_2$  (SubOp):  
Wenn  $E_1$  und  $E_2$  beide vom Typ **int**,  
dann ist **Ergebnis** vom Typ **int**.

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassu

# Semantik

*Semantik* beschreibt die Bedeutung eines Programmes zur Ausführungszeit. Allgemeine Terminologie:

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassu

*Semantik* beschreibt die Bedeutung eines Programmes zur Ausführungszeit. Allgemeine Terminologie:

## Anweisungen

... werden **ausgeführt**. Mögliche Seiteneffekte:

- Ändern der Werte von Variablen
- Ein-/Ausgabeoperationen

## Ausdrücke

... werden **ausgewertet** (evaluiert), um ein Ergebnis zu erhalten.

- Die Evaluation kann in einigen Sprachen auch Seiteneffekte haben.

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassung

## Ausdrücke

... werden **ausgewertet** (evaluiert), um ein Ergebnis zu erhalten.

- Die Evaluation kann in einigen Sprachen auch Seiteneffekte haben.

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassung



## Ausdrücke

... werden **ausgewertet** (evaluiert), um ein Ergebnis zu erhalten.

- Die Evaluation kann in einigen Sprachen auch Seiteneffekte haben.

## Deklarationen

... werden **elaboriert** um eine Bindung vorzunehmen.

Mögliche Seiteneffekte:

- Allokieren von Speicherplatz
- Initialisieren von Speicherplatz

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Die Beschreibung orientiert sich am AST.

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassu

Die Beschreibung orientiert sich am AST.

AssignCmd  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Die Beschreibung orientiert sich am AST.

**AssignCmd**  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten
- 2  $v$  wird an die Variable  $v$  zugewiesen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Die Beschreibung orientiert sich am AST.

**AssignCmd**  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten
- 2  $v$  wird an die Variable  $v$  zugewiesen

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassu

Die Beschreibung orientiert sich am AST.

**AssignCmd**  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten
- 2  $v$  wird an die Variable  $v$  zugewiesen

**BinaryExp**  $E_1 \text{ op } E_2$

- 1 Der Ausdruck  $E_1$  wird evaluiert um einen Wert  $v_1$  zu erhalten

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Die Beschreibung orientiert sich am AST.

**AssignCmd**  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten
- 2  $v$  wird an die Variable  $v$  zugewiesen

**BinaryExp**  $E_1 \text{ op } E_2$

- 1 Der Ausdruck  $E_1$  wird evaluiert um einen Wert  $v_1$  zu erhalten
- 2 Der Ausdruck  $E_2$  wird evaluiert um einen Wert  $v_2$  zu erhalten

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Die Beschreibung orientiert sich am AST.

AssignCmd  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten
- 2  $v$  wird an die Variable  $v$  zugewiesen

BinaryExp  $E_1 \text{ op } E_2$

- 1 Der Ausdruck  $E_1$  wird evaluiert um einen Wert  $v_1$  zu erhalten
- 2 Der Ausdruck  $E_2$  wird evaluiert um einen Wert  $v_2$  zu erhalten
- 3 Die Werte  $v_1$  und  $v_2$  werden mit dem Operator  $\text{op}$  zu einem Wert  $v_3$  verknüpft.



Die Beschreibung orientiert sich am AST.

AssignCmd  $v := E$

- 1 Der Ausdruck  $E$  wird evaluiert um einen Wert  $v$  zu erhalten
- 2  $v$  wird an die Variable  $v$  zugewiesen

BinaryExp  $E_1 \text{ op } E_2$

- 1 Der Ausdruck  $E_1$  wird evaluiert um einen Wert  $v_1$  zu erhalten
- 2 Der Ausdruck  $E_2$  wird evaluiert um einen Wert  $v_2$  zu erhalten
- 3 Die Werte  $v_1$  und  $v_2$  werden mit dem Operator  $\text{op}$  zu einem Wert  $v_3$  verknüpft.
- 4  $v_3$  ist das Ergebnis der BinaryExp

Declaration **var** **I** : **T**

- 1 Der Bezeichner **I** wird an eine Variable vom Typ **T** gebunden

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Declaration **var** **I** : **T**

- 1 Der Bezeichner **I** wird an eine Variable vom Typ **T** gebunden
- 2 Es wird ein für **T** passender Speicherbereich bereitgestellt

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Declaration **var**  $I$  :  $T$

- 1 Der Bezeichner  $I$  wird an eine Variable vom Typ  $T$  gebunden
- 2 Es wird ein für  $T$  passender Speicherbereich bereitgestellt
- 3 Der Speicherbereich ist *nicht* initialisiert

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Declaration **var**  $\mathbf{I}$  :  $\mathbf{T}$

- 1 Der Bezeichner  $\mathbf{I}$  wird an eine Variable vom Typ  $\mathbf{T}$  gebunden
- 2 Es wird ein für  $\mathbf{T}$  passender Speicherbereich bereitgestellt
- 3 Der Speicherbereich ist *nicht* initialisiert
- 4 Der Geltungsbereich für  $\mathbf{I}$  ist der eingeschlossene Block (LetCmd)

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

Declaration **var**  $\mathbf{I}$  :  $\mathbf{T}$

- 1 Der Bezeichner  $\mathbf{I}$  wird an eine Variable vom Typ  $\mathbf{T}$  gebunden
- 2 Es wird ein für  $\mathbf{T}$  passender Speicherbereich bereitgestellt
- 3 Der Speicherbereich ist *nicht* initialisiert
- 4 Der Geltungsbereich für  $\mathbf{I}$  ist der eingeschlossene Block (LetCmd)
- 5 Am Ende des Blockes wird die Bindung aufgehoben

## Declaration `var I : T`

- 1 Der Bezeichner  $\mathbf{I}$  wird an eine Variable vom Typ  $\mathbf{T}$  gebunden
- 2 Es wird ein für  $\mathbf{T}$  passender Speicherbereich bereitgestellt
- 3 Der Speicherbereich ist *nicht* initialisiert
- 4 Der Geltungsbereich für  $\mathbf{I}$  ist der eingeschlossene Block (LetCmd)
- 5 Am Ende des Blockes wird die Bindung aufgehoben
- 6 ... und der Speicherbereich wieder freigegeben

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassu



- In Vorlesung: Mini-Triangle
  - Stark vereinfacht
  - Z.B. Keine Unterprogramme (Prozeduren/Funktionen)

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

**Semantik**

Zusammenfassung

- In Vorlesung: Mini-Triangle
  - Stark vereinfacht
  - Z.B. Keine Unterprogramme (Prozeduren/Funktionen)
- Im praktischen Teil: Triangle
  - Pascal-artige Sprache
  - Arrays, Records, Prozeduren, Funktionen
  - Parameterübergabe durch Wert oder Referenz
  - Prozeduren/Funktionen als Parameter erlaubt
  - Ausdrücke haben *keine* Seiteneffekte

- In Vorlesung: Mini-Triangle
  - Stark vereinfacht
  - Z.B. Keine Unterprogramme (Prozeduren/Funktionen)
- Im praktischen Teil: Triangle
  - Pascal-artige Sprache
  - Arrays, Records, Prozeduren, Funktionen
  - Parameterübergabe durch Wert oder Referenz
  - Prozeduren/Funktionen als Parameter erlaubt
  - Ausdrücke haben *keine* Seiteneffekte
- Beschreibung in PLPJ, Anhang B

- Überblick
- Organisation
- Material in PLPJ, Kapitel 1
  - Syntax (konkrete und abstrakte)
  - Kontextuelle Einschränkungen
  - Semantik
  - AST als IR
  - (Mini-)Triangle

C1

A. Koch

Einleitung

Zielmaschine

Aufbau

Optimierung

Orga

Syntax

Triangle

Kontext

Semantik

Zusammenfassung