



Optimierende Compiler

1. Einleitung

Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen
Informatik, TU Darmstadt

Sommersemester 2006

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



- Schnittstelle zwischen
 - Programmiersprache
 - Maschine

Programmiersprache Gut für Menschen handhabbar

- Smalltalk
- Java
- C++

Maschine Getrimmt auf

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



Entscheidet über dem Benutzer zugängliche Rechenleistung

Beispiel: Bildkompression auf Dothan CPU, 2GHz

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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)
```

Niedrige Ebene:
Maschinensprache

```
0110000100000110
0111001001000001
1011000100010010
1001000100000110
```

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Unterschiedliche Abstraktionsebenen



- 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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

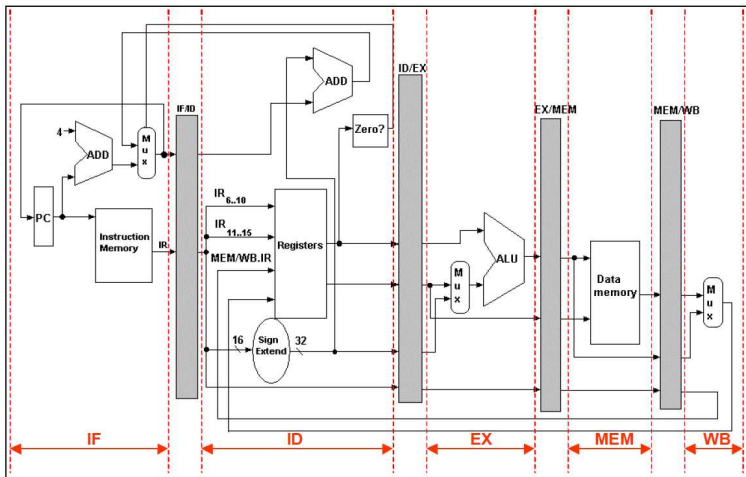
(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Einfach: Hennessy & Patterson DLX



Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

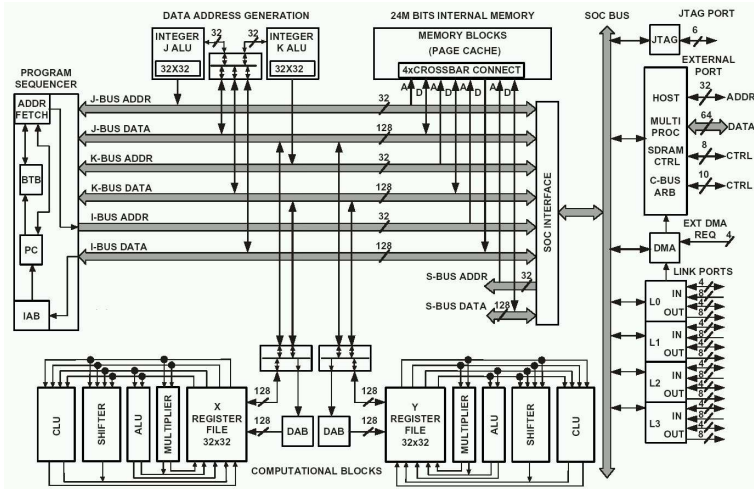
(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Komplizierter: Analog Devices TigerSHARC



Einleitung

Auswirkungen der Zielmaschine

Aufbau von Compilern

Optimierung

Organisatorisch

Syntax

(Mini-) Triangle

Kontextuelle Einschränkungen

Semantik

Zusammenfassung

Problematisch: IBM/Sony Cell Processor



Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

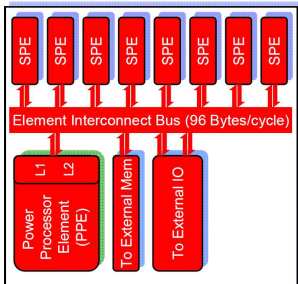
(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

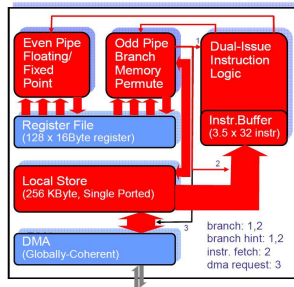
Semantik

Zusammenfassu

Übersicht



SPE



Spezialisierte Anforderungen



- 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**

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Spezialisten gesucht



Apple Computer

Job Title Sr Compiler Engineer

Posting Date 1/30/2006

Job Location

Description

Job Title Principal Compiler Engineer

Posting Date

Job Location

Description

Principal Compiler Engineer

Sony Computer Entertainment

Job Title Compiler Engineer

Posting Date

Job Location

Description

Compiler Engineer

Posting Date 11/15/2005

Job Location Aachen, Germany

Description

Research assistant/PhD candidate

11/15/2005

Aachen, Germany

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

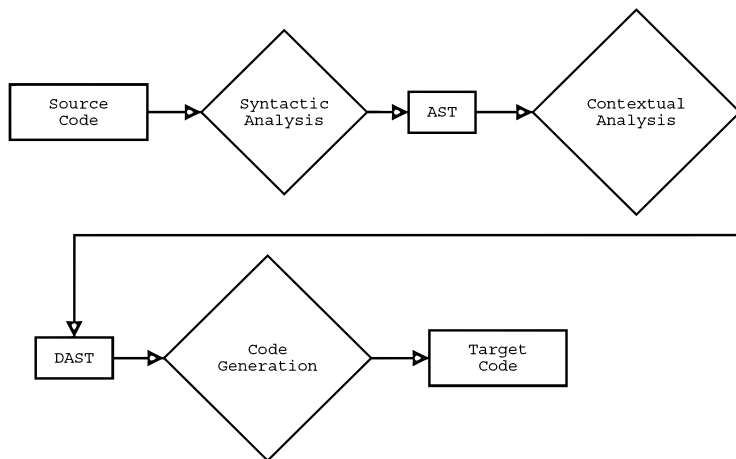
(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Vorgehen: Bearbeitung in mehreren Phasen



Zwischendarstellung(en) für den Informationsaustausch

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

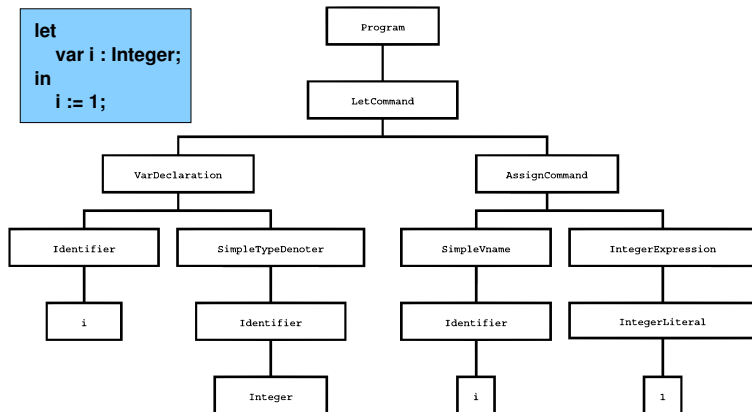
Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung



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



Abstrakter Syntaxbaum

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

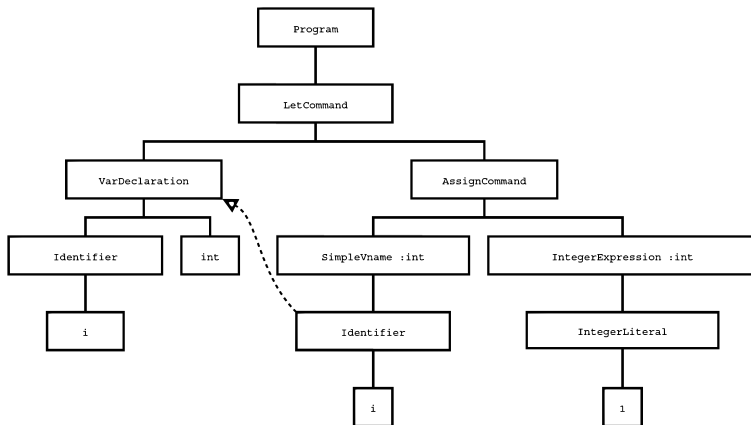
Semantik

Zusammenfassung

Kontextanalyse - Identifikation



- Ordne Variablen ihren Deklarationen zu
- Berechne Typen von Ausdrücken



DAST: Dekorierter bzw. annotierter AST

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

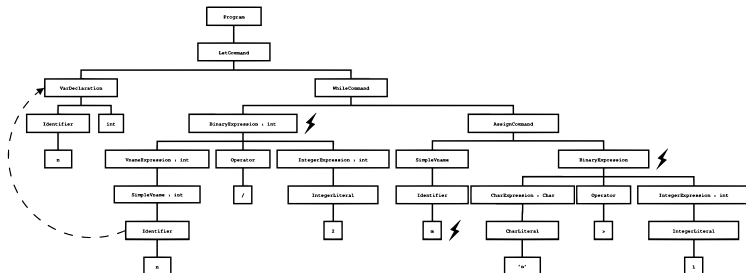
Semantik

Zusammenfassung

Kontextanalyse - Überprüfung



Erkenne Fehler in Variablen- und Typzuordnung



Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass



- 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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

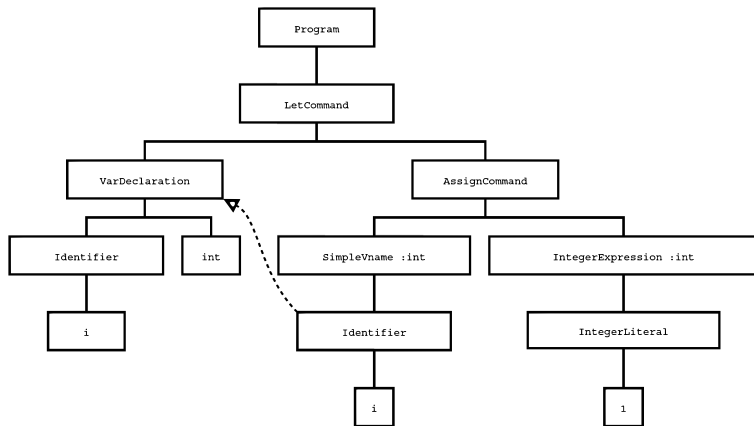
(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Code-Erzeugung 2



```
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
```

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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

Ziel: Verbesserung des erzeugten Codes in Bezug auf bestimmte Gütemaße

Einleitung

Auswirkungen der Zielmaschine

Aufbau von Compilern

Optimierung

Organisatorisch

Syntax

(Mini-) Triangle

Kontextuelle Einschränkungen

Semantik

Zusammenfassung

Beispiele für Optimierung



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;  
}
```

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass



Dozent

Andreas Koch
Sprechstunde

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

Web-Site

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



1. Teil: **Fast vollständig**

Programming Language Processors in Java

von David Watt und Deryck Brown, Prentice-Hall 2000

2. Teil: Auszugsweise

Engineering a Compiler

von Keith Cooper und Linda Torczon, Elsevier 2004

Advanced Compiler Design and Implementation

von Steven Muchnick, Morgan-Kaufman 1997

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



- Fließender Übergang zwischen
 - Vorlesung
 - Praktischen Arbeiten
- Anfangs mehr Vorlesungen (4+0 SWS)
- Dann Vorlesungen und praktische Experimente
 - Dann 2+2 SWS
- Praktische Arbeiten
 - **Erweitern** eines bestehenden Compilers
 - Einfache Programmiersprache: Triangle
 - Zielmaschine: Simulierte Stack-Maschine
 - Programmierung überwiegend in Java

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung



Zwei Varianten

1 IV4 (6CP): 18 Plätze

- 50 % Einzelprüfung: Klausur von 60 Minuten Dauer
 - Semestermittig gelegen
- 50 % Ergebnisse der praktischen Arbeiten
 - Gruppenarbeit maximal in 3er Gruppen
 - Gruppenweise Kolloquien (30 Min.) und Abschlußvortrag (10-15 Min.)

2 V3 (4.5CP): Restliche Teilnehmer

- Zwei Klausuren von je 60 Minuten
 - Semestermittig und gegen Semesterende

Prüfungsmodus Bachelor/Master vs. Diplom

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass



Überblick über kompletten Compiler (ca. 6-7 Wochen)

- Von Front-End (Lexing/Parsing)
- ... bis Back-End (Code-Erzeugung)
- Schwerpunkt: Hintere Phasen
 - Dort Komplexität bei modernen Compilern

Dieser Teil lehnt sich an an die Veranstaltungen

- IMT3052 von Ivar Farup, Universität Gjøvik, Norwegen
- Vertalerbouw 2004 von Theo Ruys, Universität Twente, Niederlande

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass



Optimierung (ca. 5-6 Wochen)

- Andere IRs
- IR-IR Transformationen
- Implementierung ausgewählter Verfahren

Nicht behandelt werden

- Entwurf von Programmiersprachen
- Übersetzung von objektorientierten Sprachen
- Komplexe Laufzeitsysteme
 - Garbage Collection
 - Just-In-Time Compiler
 - ➔ VL Virtuelle Maschinen, Dr. Haupt

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



PLPJ, Kapitel 1

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



Beschreibt die Satzstruktur von korrekten Programmen

- $n := n + 1;$
Syntaktisch korrektes Statement in Triangle
- “Ein Kreis hat zwei Ecken.”
Syntaktisch korrekte Aussage in Deutsch

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung



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

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung



- 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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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
- Mögliche Vorgehensweisen
 - 1 Mathematische Mengennotation
 - 2 Reguläre Ausdrücke
 - 3 Kontextfreie Grammatik

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass



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 **xyy, xyy, xxxyy, ...**
- $L = \{\mathbf{x}^n \mathbf{y}^n | n > 0\}$ beschreibt **xy, xxyy, ...**, aber z.B. nicht **xxy**

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Reguläre Ausdrücke (REs)



Erweitere Zeichenketten aus dem Alphabet um Operatoren

- | zeigt Alternativen an
- * zeigt Null oder mehr Vorkommen des vorangehenden Zeichens an
- ε 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, **ab**, **abab**, **ababab**, ...
- $L = \mathbf{a(b|\varepsilon)}$ ergibt **a** oder **ab**

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Mächtigkeit von REs?



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).

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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 , die beschreiben, wie Nicht-Terminalsymbole aus Terminalsymbolen zusammengesetzt sind.

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Kontextfreie Grammatiken (CFGs) 2



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

BNF Beispiel 1



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

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

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

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

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

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

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

$\mathbf{S} \rightarrow \mathbf{xS} \rightarrow \mathbf{xxS} \rightarrow \mathbf{xxyB} \rightarrow \mathbf{xxyyB} \rightarrow \mathbf{xxyyy}$

➡ Ja, da sie sich aus S herleiten läßt.

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

BNF Beispiel 2



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

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

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

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

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

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

Ist die Zeichenkette \mathbf{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.

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Mehrdeutigkeit in BNF



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:

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



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

Eindeutige Produktionen für die gleiche CFG:

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

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

(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

Lokale Deklarationen

Konstante (häßliches "~"!)

Variable kann in `getint` verändert werden

Folge von Anweisungen
zwischen `begin/end`

`else` ist erforderlich (darf aber leer sein)

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Produktionen der Mini-Triangle CFG 1



```
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
...

```

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Produktionen der Mini-Triangle CFG 2



```
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 ::= + | - | * | / | < | > | =
```

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Produktionen der Mini-Triangle CFG 3



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-
Triangle)

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Produktionen der Mini-Triangle CFG 4



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



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.

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

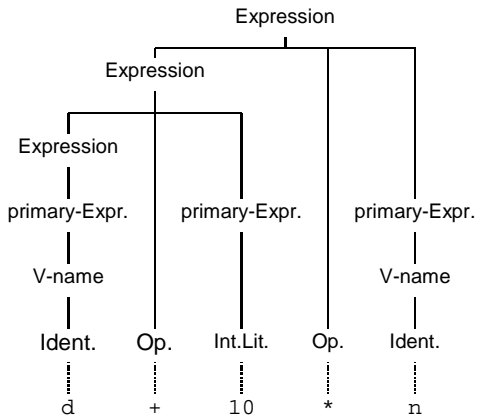
(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Expression-Baum für $d + 10 * n$



Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorische

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

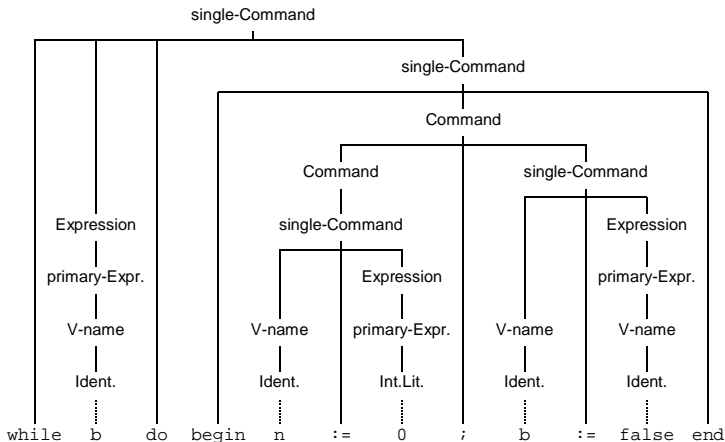
Semantik

Zusammenfassung

single-Command-Baum



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



Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Konkrete und abstrakte Syntax



- 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
- $V \leftarrow E$

- ... können alle das gleiche bedeuten: Eine Zuweisung von **E** nach **V**

➔ Für weitere Verarbeitung Darstellung vereinfachen!

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Auszug aus der abstrakten Syntax 1



Command

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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Auszug aus der abstrakten Syntax 2



Expression

::= Integer-Literal

| V-name

| Operator Expression

| Expression Op Expression

V-name ::= Identifier

IntegerExp

VnameExp

UnaryExp

BinaryExp

SimpleVName

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

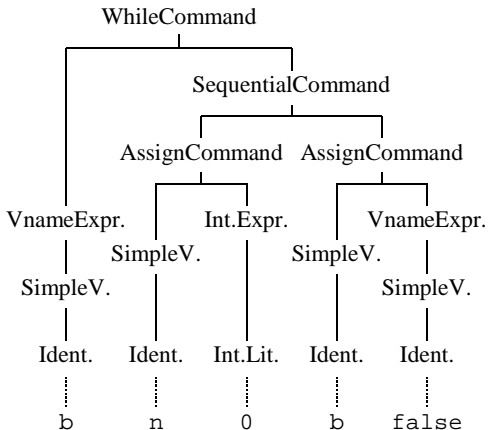
Semantik

Zusammenfassu

Beispiel abstrakte Syntax



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



Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

AST als Zwischendarstellung 1



- 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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



Schlechter geeignet für maschinennahe Analysen und Transformationen

- Ausnutzung von Maschinenregistern
 - Ausnutzung von speziellen Maschinenbefehlsfolgen
- ➔ Hier: Konzentration auf **maschinunabhängige** Ebene
- (D)AST ist Hauptrepräsentation
 - Für einzelne Bearbeitungsschritte: Andere IRs

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Kontextuelle Einschränkungen: Geltungsbereiche



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung

Beispiele Geltungsbereiche



```
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 von m keine Bindung
von m existiert: Fehler!

Verwendung von m

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

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)

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass

Beispiele Typen



```
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**.

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassung



Semantik beschreibt die Bedeutung eines Programmes zur Ausführungszeit. Allgemeine Terminologie

Anweisungen

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

- Ändern der Werte von Variablen
- Ein-/Ausgabeoperationen

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu

Beispiele Semantik von (Mini)Triangle 1



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 op zu einem Wert v_3 verknüpft.
- 4 v_3 ist das Ergebnis der BinaryExp

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass

Beispiele Semantik von (Mini)Triangle 2



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
- 4 Der Geltungsbereich für I ist der eingeschlossene Block (LetCmd)
- 5 Am Ende des Blockes wird die Bindung aufgehoben
- 6 ... und der Speicherbereich wieder freigegeben

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu



- 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
 - Teils eigenartige Syntax . . .
- Beschreibung in PLPJ, Anhang B

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfass



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

Einleitung

Auswirkungen
der
Zielmaschine

Aufbau von
Compilern

Optimierung

Organisatorisch

Syntax

(Mini-)
Triangle

Kontextuelle
Ein-
schränkungen

Semantik

Zusammenfassu