The Conversion of Source Code to Machine Code

Silas Groh, Mik Müller

17. März 2023

Carl-Fuhlrott-Gymnasium

Einstieg & Motivation

Einstieg & Motivation

- Computerprogrammer werden oft in speziell entwickelten Programmiersprachen verfasst
- Vorteile eines hohen Abstraktionsgrades [Dan05, S. 9]:
  - Entwicklung ist einfacher und schneller
  - Programme sind portabel
  - Instandhaltung ist einfacher

Zentrales Problem

```
1   fn main() {
2       foo(2);
3   }
4
5   fn foo(n: int) {
6       let mut m = 3;
7       exit(n + m);
8   }
```

Übersetzung

```
1   ; RISC-V binary
2   0000000 457f 464c 0102
3   0000010 0002 00f3 0001
4   0000020 0040 0000 0000
5   0000030 0005 0000 0040
6   0000040 0001 0000 0005
7   0000050 0000 0001 0000
8   0000060 0124 0000 0000
9   0000070 1000 0000 0000
10  0000080 00b0 0000 0000
```

- Programme sollten einfach zu schreiben sein

$\Rightarrow$ Ein Computer muss diese jedoch auch *einfach* verarbeiten

Methoden zur Programmausführung

- Man unterscheidet zwischen *Compilern* und *Interpretern*
- Ein Compiler (auch *Übersetzer*) übersetzt die Sprache in ein zielspezifiesches Format, sodass ein Computer dieses verstehen kann
- Ein Interpreter führt das Programm direkt aus, ohne es vorher zu bearbeiten

Etappen der Übersetzung

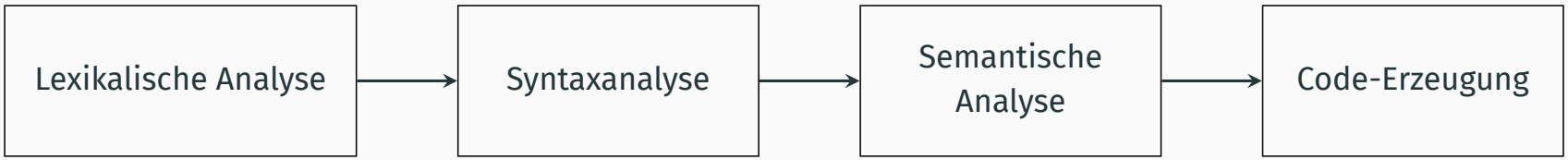

Abbildung 1 – Etappen der Übersetzung

[Wir05, S. 6–7]

Etappen der Übersetzung (angepasst)

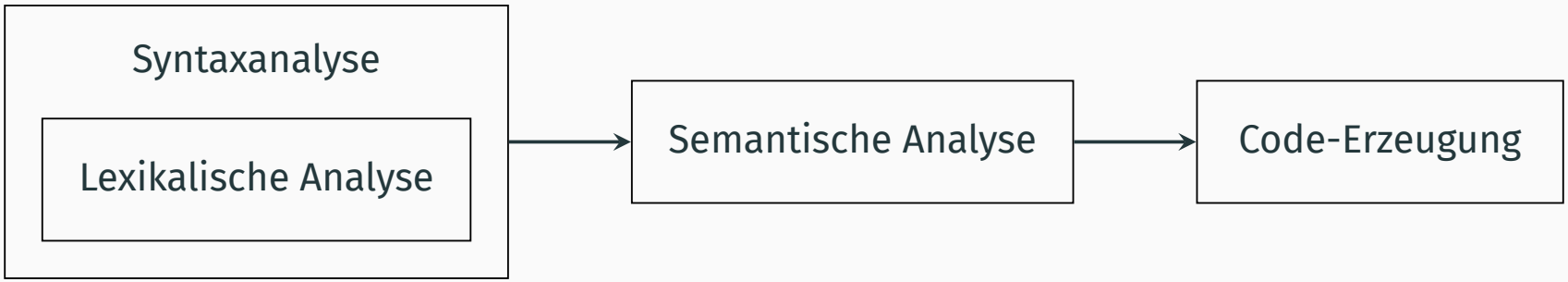

Abbildung 2 – Etappen der Übersetzung (angepasst)

[Wir05, S. 6–7]

Die Programmiersprache „rush“

```
1  fn main() {
2      exit(fib(10));
3  }
4
5  fn fib(n: int) -> int {
6      if n < 2 {
7          n
8      } else {
9          fib(n - 2) + fib(n - 1)
10     }
11 }
```

1.1 – Erzeugen von Fibonaccizahlen in rush

Fähigkeiten von rush

Tabelle 1 – Die wichtigsten Fähigkeiten von rush

Bezeichnung	rush Ausdruck
Loop	`loop { }`
While	`while x < 5 { }`
For	`for i = 0; i < 5; i += 1 { }`
If	`if true { /* */ } else { }`
Fn	`fn foo(n: int) { }`
Infix Expr	`1 + n; 5 ** 2`
Prefix Expr	`!false; -n`
Variables	`let mut answer = 42`
Cast Expr	`42 as char`

Datentypen in rush

Tabelle 2 – Datentypen in rush

Bezeichnung	Instanziierung einer Variable
‘int’	`let a: int = 0;`
‘float’	`let b: float = 3.14;`
‘bool’	`let c: bool = true;`
‘char’	`let d: char = 'a';`
‘()’	`let e: () = main();`
‘!’	`let f = exit(42);`

Fakten über rush

- Im Commit `4de569a` umfasste das Projekt 17456 Zeilen Programmtext[1]
- Das Projekt enthält einen Lexer, einen Parser, fünf Compiler und einen Interpreter

[1] Leerzeilen und Kommentare werden nicht gezählt

Programmtext der einzelnen Komponenten

Tabelle 3 – Zeilen Programmtext pro Komponente

Komponente	Zeilen Programmtext
Lexer / Parser	2737
Tree-walking interpreter	578
VM compiler / runtime	1281
WASM compiler	1584
LLVM compiler	1450
RISC-V compiler	2275
x86 compiler	2751

Lexikalische & Syntaktische Analyse

- Gruppieren des Programmtextes in Tokens
- Analyse der Syntax des Programms
- Generation eines abstrakten Syntaxbaums
- Festlegen der formalen Regeln in Form einer Grammatik

```
1  Expression = Term , { ( '+' | '-' ) , Term } ;
2  Term       = Factor , { ( '*' | '/' ) , Factor } ;
3  Factor     = ( integer
4               | '(' , Expression , ')' ) , [ '**' , Factor ] ;
5  integer    = { '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8'
   ↪  | '9' }- ;
```

1.2 – Ein Beispiel für eine Grammatik

Abstrakter Syntaxbaum

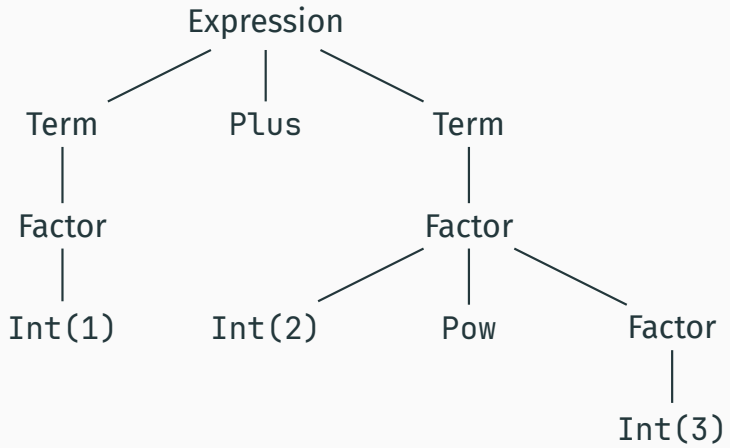

Abbildung 3 – abstakter Syntaxbaum für ‘1+2**3’

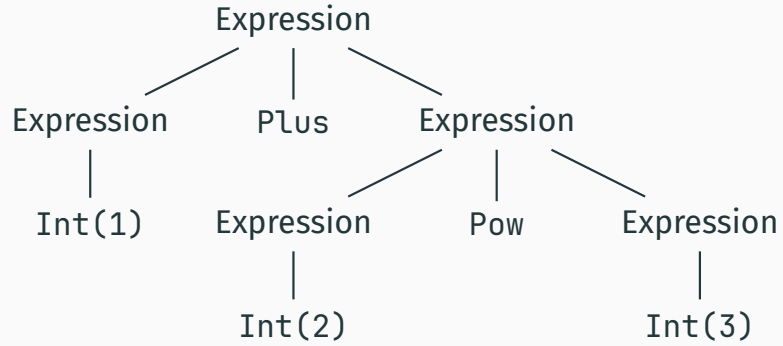

Abbildung 4 – abstakter Syntaxbaum für ‘1+2**3’, erstellt durch Pratt-Parsing

Interpreter

Interpreter

TODO: Write this

Kompilierung zu high-level Architekturen

Kompilierung zu high-level Architekturen

TODO: Write this

Kompilierung zu low-level Architekturen

Kompilierung zu low-level Architekturen

TODO: Write this

Finale Anmerkungen & Fazit

Finale Anmerkungen & Fazit

TODO: Write this

Programmblockverzeichnis

Literatur

Sivarama P Dandamudi. *Guide to RISC processors*. Ottawa, Canada: Springer International Publishing, Feb. 2005. ISBN: 0-387-21017-2.

Niklaus Wirth. *Compiler Construction*. Zürich, 2005. ISBN: 0-201-40353-6.