Karlsruher Institut für Technologie

Praktikum Compilerbau
Sitzung 2 – Lexer

Prof. Dr.-Ing. Gregor Snelting
Matthias Braun und Sebastian Buchwald

IPD Snelting, Lehrstuhl für Programmierparadigmen

Inhalt

Altes Übungsblatt

- Infrastruktur: Alles reibungslos verlaufen?
- Vorbereitung
  - Schreiben von MiniJava Programmen - Besonderheiten?
  - Turingmächtig?
  - Untermenge von Java?

Beispielprogramme

Programm 1:

```
public class Prog1 {
   public static void main(String[] args) {
   }
}
```

Programm 2:

```
public class Prog2 {
   public static void main(String[] args) {
      System.out.println(args[0]);
   }
}
```

Beispielprogramme

Karlsruher Institut für Technologie

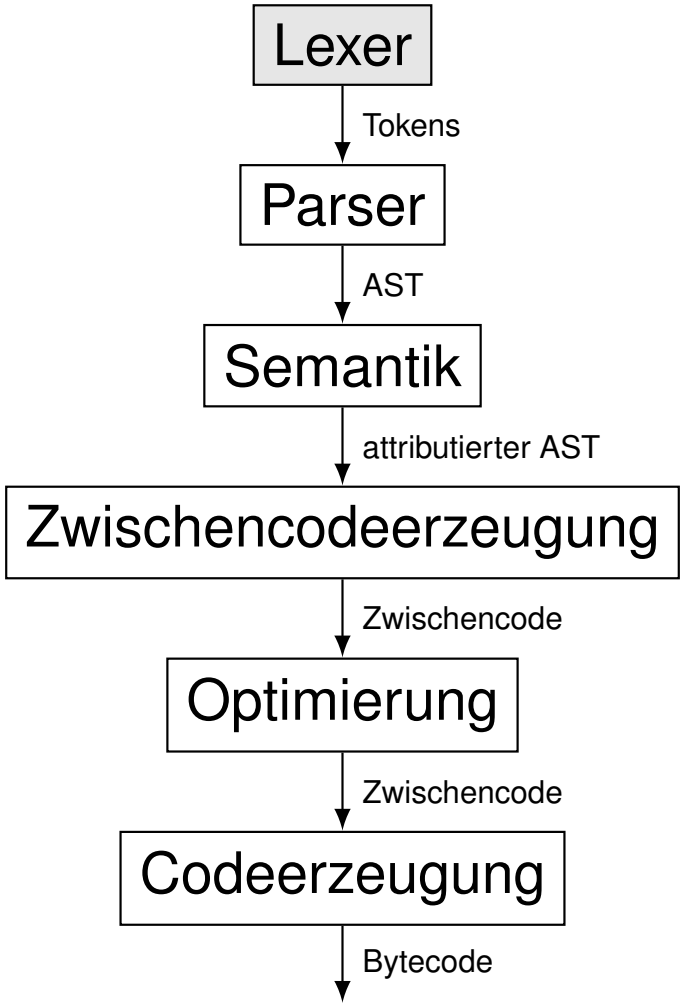
Programm 3:

```
public class Factorial {
   public int fac(int n) {
      if (n < 2)
         return 1;
      return n * fac(n-1);
   }
}

public class Prog3 {
   public static void main() {
      Factorial f = new Factorial();
      int n = f.fac(42);
      System.out.println(n);
   }
}
```

Phasen

Compiler sind meist in mehrere Phasen untergliedert. Bei uns:



Lexer

Zerteilt die Eingabe in die grundlegenden Symbole (Tokens) der Sprache.

void method(**int** a) { **return** –42; }

$\Longrightarrow$

void
identifier method
(
int
identifier a
)
{
return
–
integer literal 42
;
}
EOF

Theoretische Grundlagen

Karlsruher Institut für Technologie

Grundlagen aus Vorlesung bekannt:

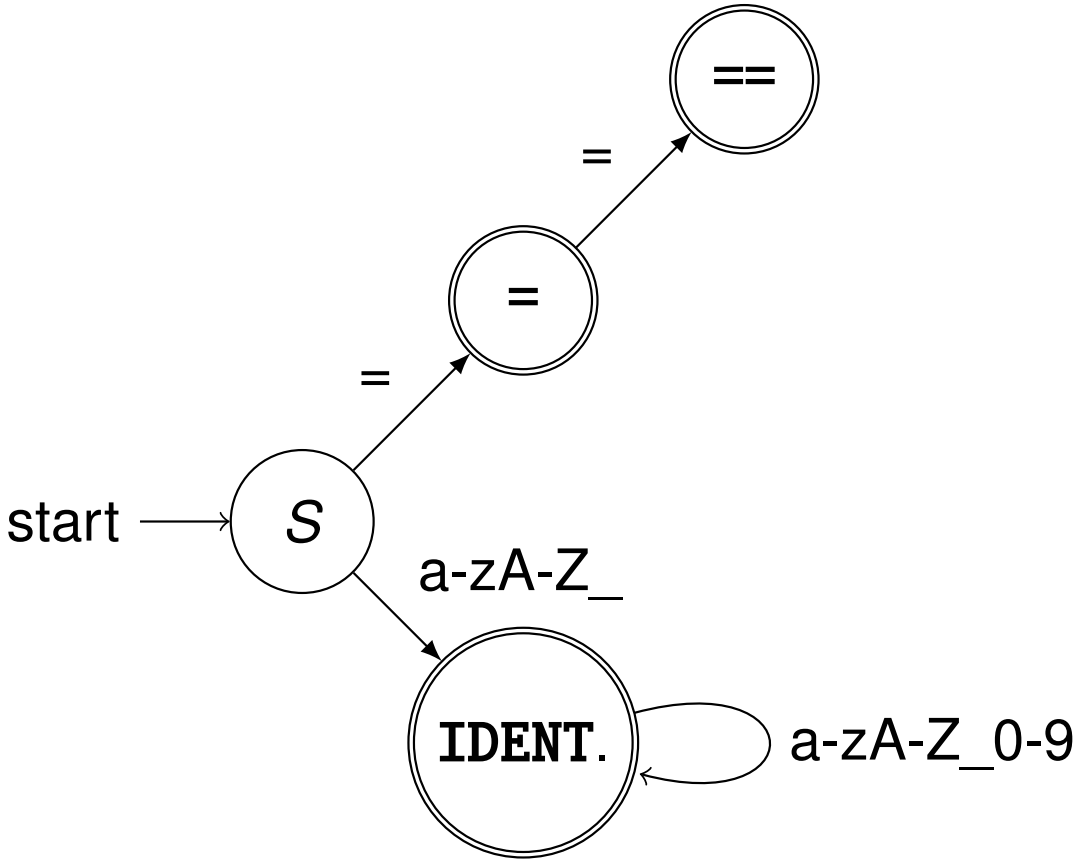
- reguläre Ausdrücke
- endliche Automaten
- deterministische endliche Automaten

Beispiel

Unterscheide Bezeichner = und ==.

Beispiel

Unterscheide Bezeichner = und ==.



Implementierung 1

Karlsruher Institut für Technologie

Automat:

- Wie arbeitet man den Automat ab?
- Was tun wenn ein Endzustand erreicht ist?
- Wann Aktionen ausführen?
- Wo kommen die Attribute für **`IDENTIFIER`** und **`INTEGER_LITERAL`** her?

Endlicher Automat:

- Tabellen
- „Als Code”

Implementierung 2

Karlsruher Institut für Technologie

Ein- und Ausgabe:

- Wie sieht die Eingabe aus?
- Wie wird eingelesen? (Zeichen für Zeichen, Puffern, Memory-Mapping)
- Welche Datenstrukturen werden für Tokens benutzt? Manche Tokenarten haben zusätzliche Attribute.

API

Klar:

void init(stream input, string input_name);

void quit();

Pull-Interface:

token get_next_token();

Push-Interface:

typedef **void** (∗token_recognized_func)(token_t token);
void set_token_recognized_callback(token_recognized_func func);
void lex();

Anregung 1

```
token get_next_token() {
   int c = input.get_char();
   switch (c) {
   case EOF: return token(T_EOF);
   case ';': return token(T_SEMICOLON);
   case '=':
      c = input.get_char();
      if (c == '=')
         return token(T_EQUALEQUAL);
      input.putback(c);
      return T_EQUAL;
   case 'a'..'z', 'A'..'Z', '_':
      input.putback(c);
      return parse_identifier();
   }
}
token parse_identifier() { /* ... */}
```

Anregung 2

```
int c; void next_char();

token get_next_token() {
   switch (c) {
   case EOF: return token(T_EOF);
   case ';': return token(T_SEMICOLON);
   case '=':
      next_char();
      if (c == '=') {
         next_char();
         return token(T_EQUALEQUAL);
      }
      return T_EQUAL;
   case 'a'..'z', 'A'..'Z', '_':
      return parse_identifier();
   }
}
```

Anregung 3

```
token_recognized_func func;
int next_states[N_STATES][256];
token_t* actions[N_STATES][256];

void lex() {
   int state = 0;
   while (!feof(input)) {
      c = input.get_char();

      action = actions[state][c];
      if (action != NULL)
         func(action);
      state = next_states[state][c];
   }
   func(T_EOF);
}
```

Stringtabelle

Karlsruher Institut für Technologie

API:

symbol_t* insert(string string);

Aufgabe: Speichert Strings der Bezeichner in einer Hashmap. Jeder String wird nur einmal eingetragen.
Warum?: Erleichtert und beschleunigt Identifizierung von Bezeichnern in der semantischen Analyse.

- Vergleich zweier Referenzen/Zeiger statt Stringvergleiche
- Symbol kann temporär zusätzliche Informationen enthalten. Beispiel: Momentan gültige Definition eines Bezeichners.

Zusätzlich: Vereinfachung des Automaten möglich wenn man Schlüsselwörter in Stringtabelle einträgt und Regeln für `IDENTIFIER` benutzt. Starke Reduktion der Zustandsmenge und einfacher Code.

Testen

Bei Compilern handelt es sich normalerweise um relativ komplexe Software an die hohe Korrektheitsanforderungen gestellt werden. Unserer Erfahrung nach lassen sich diese nur durch intensives Testen sicherstellen.

Wette

Schickt uns bis nächsten Dienstagabend euren Lexer und wir werden einen Bug darin finden! Details zum Format der Ausgaben sind auf dem Übungsblatt.

Nicht Behandelt

Hier nicht behandelt und im Rahmen des Praktikums nicht notwendig:

- Fehlermeldungen und Warnungen mit Informationen zur Lokalisierung der Fehler.
- Unterschiedliche Zeichensätze (im Praktikum: Nur ASCII).

Sonstiges

- Gibts noch Fragen?
- Ansonsten einfach vorbeikommen oder Email schreiben.
- Viel Erfolg!