Einführung in den Compilerbau

Prof. Dr.-Ing. Andreas Koch
David Volz, Tim Noack, Jan Braun

TECHNISCHE UNIVERSITÄT DARMSTADT

Theorie
Wintersemester 23/24
Übungsblatt 2

Abgabe bis Sonntag, 03.12.2022, 18:00 Uhr (MEZ)

Aufgabe 2.1: LL(k)-Grammatiken (14 Punkte)

Gegeben sei die Grammatik `G = ({S, W, X, Y, Z}, {a, b, c, d}, P, S)`. Die Menge `P` enthält folgende Produktionen:

```
S ::= b Y d X c W d Z a
W ::= (X Y) | a
X ::= (c b d) | (d a b) | (X X)
Y ::= (d Y d) | a
Z ::= ε | b*
```

2.1a) Anfangsmengen (4 Punkte)

Geben Sie die `starters` Menge für die Nichtterminale `W`, `X`, `Y` und `Z` an.

2.1b) Folgemengen (4 Punkte)

Geben Sie die `follow` Menge für die Nichtterminale `W`, `X`, `Y` und `Z` an.

2.1c) Nachweis der LL(1)-Eigenschaft (6 Punkte)

Entscheiden Sie nun, ob die Produktionen der Nichtterminale `W`, `X`, `Y` und `Z` jeweils die LL(1)-Bedingung erfüllen. Geben Sie hierzu Ihren **Lösungsweg** an. Dieser sollte zunächst einen Ansatz enthalten (Vereinigung und Schnitt von `starters`- und `follow`-Mengen), und dann die Inhalte der konkreten Mengen zeigen. Falls mehrere Bedingungen für ein Nichtterminal zu prüfen sind, sind **alle** Bedingungen anzugeben und jede Bedingung einzeln zu überprüfen.

2.1 a) starters[W] = {a, c, d}

starters[X] = {c, d}

starters[Y] = {d, a}

starters[Z] = {} ?

2.1 b) follow[W] = {d}

follow[X] = {c, d}

follow[Y] = {d}

follow[Z] = {a}

2.1 c) S: Nichts zu prüfen

W ::= XY | a

- starters[XY] ∩ starters[a] ⇔ {c, d} ∩ {a} = ∅ ✓

X ::= (cbd) | (dab)

- starters[cbd] ∩ starters[dab] ⇔ {c} ∩ {d} = ∅ ✓

(cbd) | (XX)

- starters[cbd] ∩ starters[XX] ⇔ {c} ∩ {c, d} = {c} X

(dab) | (XX)

- starters[dab] ∩ starters[XX] ⇔ {d} ∩ {c, d} = {d} X

Y ::= dYd | a

- starters[dYd] ∩ starters[a] ⇔ {d} ∩ {a} = ∅ ✓

Z ::= 3 | b*

enthält 3 → - starters[3] ∩ (starters[b*] ∪ follow[3|b*]

⇔ {} ∩ ({b} ∪ {b}) ⇔ {} ∩ {b} = ∅ ✓

?

Aufgabe 2.2: Bottom-Up Parsing (8 Punkte)

Gegeben sei die Grammatik `H = ({S, A, B, C}, {a, b, c, d}, P, S)`. Die Menge `P` enthält folgende Produktionen:

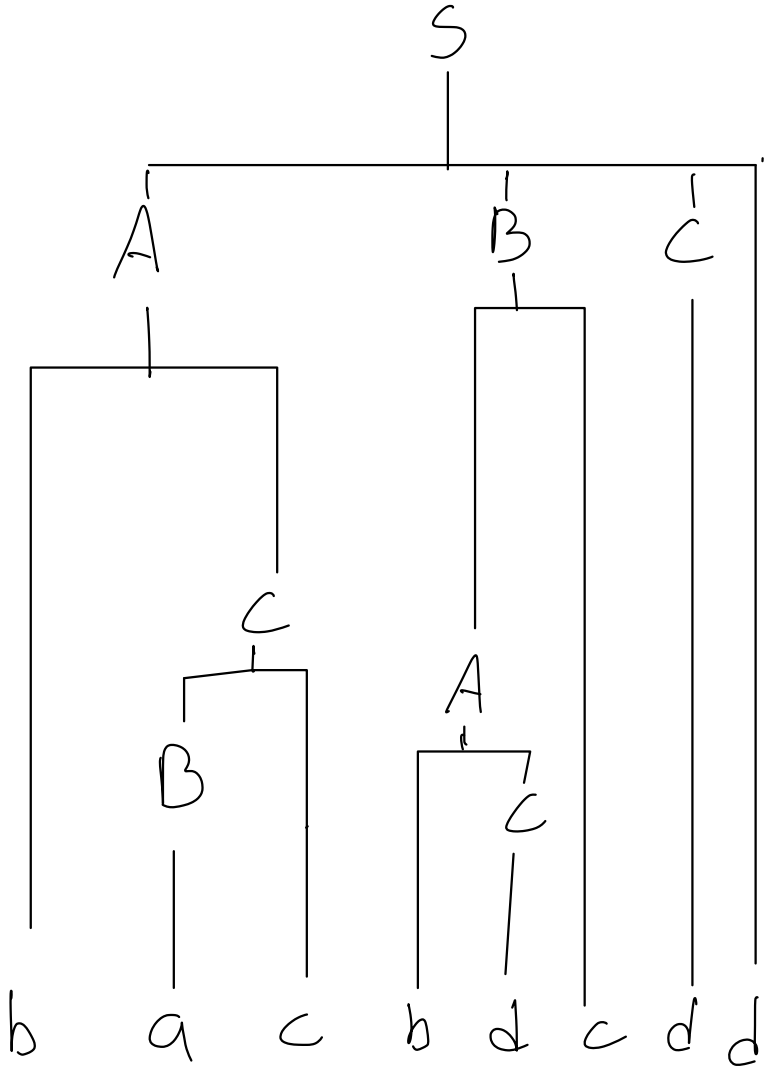
```
S ::= A B C d
A ::= b C | a C
B ::= a | A c
C ::= B c | d
```

Parsen Sie den folgenden Eingabetext mithilfe des Bottom-Up Parsing Verfahrens:

```
b a c b d c d d
```

Nutzen Sie hierzu das in der Vorlesung beschriebene Verfahren (2. Foliensatz, Folie 32). Im Falle, dass mehrere Produktionen in Frage kommen, nutzen Sie immer die Produktion mit der Sie die meisten Terminale bzw. Nichtterminale ersetzen können. Skizzieren Sie den Stack nach jedem Einfügen eines Terminals, sowie nach jeder Ersetzung durch ein Nichtterminal. **Verdeutlichen Sie wo das obere Ende (oberste Element)** des Stackes ist.

2.2



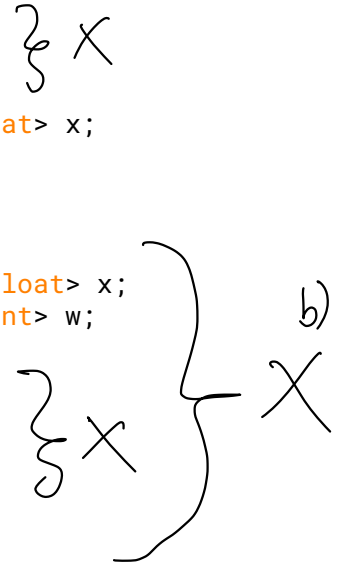
Stack:

{b}, {b, a}, {b, B},
{b, B, c}, {b, C},
{A}, {A, b} {A, b, d}
{A, b, C}, {A, A}
{A, A, c}, {A, B}
{A, B, d} {A, B, C}
{A, B, C, d}, {S}

Aufgabe 2.3: Identifikationstabelle (8 Punkte)

Gegeben sei folgender Ausschnitt aus einem MAVL-Programm:

```
...
{
  var int z;
  {
    var float z;
    var matrix<int>[1][1] y;
    {
      {
        var float x;
        var int w;
      }
      var vector<float> x;
      var int z;
      var int y;
      var float w;
      {
        var matrix<float> x;
        var vector<int> w;
        {
          var int w;
          var int y;
          var int z;
          var int x;
        }
        // HIER (a)
      }
      {
        var float x;
        var float y;
        var float z;
        // HIER (b)
      }
    }
    {
      var int w;
      var matrix<float> y;
    }
  }
  {
    var float z;
    var vector<int> z;
  }
}
...
```



2.3a) (4 Punkte)

Skizzieren Sie den Inhalt der Datenstrukturen `idents` und `scopes` zum gekennzeichneten Zeitpunkt `HIER (a)` gemäß der in der Vorlesung vorgestellten Implementierung einer Identifikationstabelle (3. Foliensatz, Folie 20f). Verwenden Sie als „Attribut“ die Zeilennummer der Deklaration, und **verdeutlichen Sie wo das obere Ende (oberste Element)** des Stacks ist.

2.3b) (4 Punkte)

Skizzieren Sie den Inhalt der Datenstrukturen `idents` und `scopes` zum gekennzeichneten Zeitpunkt `HIER (b)` gemäß der in der Vorlesung vorgestellten Implementierung einer Identifikationstabelle (3. Foliensatz, Folie 20f). Verwenden Sie als „Attribut“ die Zeilennummer der Deklaration, und **verdeutlichen Sie wo das obere Ende (oberste Element)** des Stacks ist.

2.3 a) idents = {z → (3, 5, 13, 22), y → (6, 14), x → (12, 17),
w → (15, 18)}

Scopes = {(z), (z, y), (x, z, y, w), (x, w)}

b) idents = {z → (3, 5, 13, 30), y → (6, 14, 29), x → (12, 28),
w → (15)}

Scopes = {(z), (z, y), (x, z, y, w), (x, y, z)}

Aufgabe 2.4: Grammatiktransformationen (2 Punkte)

Gegeben sei folgender Auszug aus einer Grammatik zur Beschreibung von typischen Ausdrücken in Programmiersprachen:

```
...
S ::= (S '+' T) | T
T ::= (T '*' U) | U
...
```

2.4a) (1 Punkt)

Eliminieren Sie die Linksrekursion in der S-Produktion durch Verwendung des Kleene-Sterns.

2.4b) (1 Punkt)

Alternativ könnten Sie die Linksrekursion in der S-Produktion auch folgendermaßen loswerden: `S ::= (T '+' S) | T`

Warum eignet sich dieses Vorgehen nicht für MAVL?

Aufgabe 2.5: Kontextuelle Einschränkungen (4 Punkte)

Geben Sie die Regeln zur Überprüfung der kontextuellen Einschränkungen für die folgende Codezeile gemäß der Sprachspezifikation an.

```
name = someRecord@member
```

Verwenden Sie eine kurze und präzise textuelle Beschreibung der Regeln in Ihren eigenen Worten (Kopieren Sie keine Textpassagen aus der MAVL-Spezifikation!). Es ist kein bestimmter Formalismus gefordert.

Aufgabe 2.6: Typregeln (4 Punkte)

Finden und erläutern Sie die vier Typfehler, die sich in das folgende MAVL-Programm eingeschlichen haben.

```
1  function matrix<int>[2][2] zeroMat() {
2    return [[0.0, 0.0], [0.0, 0.0]];
3  }
4
5  function bool isSorted(vector<int>[10] array, float median) {
6    foreach (val float v : array) {
7      if (v < median)
8        return 1;
9      else
10       v = 5;
11   }
12   return true;
13 }
```

2.4 a) $S ::= (T\ '+'\ T)* \mid T$

b) ?

2.5 Die Name muss eine Zeichenkette sein. Die Datentypen müssen gleich sein,

?

2.6 Zeile 1-2: Die Methode muss ein Matrix aus Integer zurückgeben, aber die Methode gibt ein Matrix aus TYP double zurück

Zeile 6: Variable vom TYP float ifiziert durch ein Array vom TYP int.

Zeile 8: wird ein int statt ein boolean zurückgegeben

Zeile 10: Neues Wert für ein val.