Compilerbau I

Ultrakompaktmerkzettel (Muster)

von Alexander Köster

Student der Universität Siegen, CB1 2020

Letzte Aktualisierung: 25. Juli 2020

Dieser Kurs der theoretischen Informatik befasst sich mit den Grundlagen und Bauteilen eines Compilers.

Einzig für Lernzwecke erstellt.

Nicht geeignet als Klausurhilfe.
Das Erstellen einer eigenen Klausurhilfe führt zu einem besonders guten Lerneffekt.
Dieses Dokument sollte nur als Orientierung oder Vergleich dienen.
Jeder sollte seine Klausurhilfe individuell auf seinen Lernstand und seine eigenen Probleme anpassen, gut bekannte Dinge auslassen und schlecht merkbare Dinge hinzufügen.
Dieses Dokument beinhaltet viel mehr, als für eine tatsächliche Klausur durchschnittlich benötigt wird.
Für einen guten Ansatz, welche Inhalte man braucht, sollte sich an der Probeklausur orientiert werden.

© study.woalk.de

Vervielfältigung ohne ausdrückliche Erlaubnis des Autors außerhalb der originalen Website untersagt.

0 Einführung

1 Analyse-Phase

1.1 Lexikalische Analyse

1.1.1 Grundlagen: Reguläre Ausdrücke

- Um Klammern zu sparen: $* > \cdot > |$
- Def.: Für den regulären Ausdruck $e \in \mathcal{E}_\Sigma$ ist die **spezifizierte Sprache** $[\![e]\!] \subseteq \Sigma^*$ ind. def. durch:
  - ▷ $[\![\varepsilon]\!] = \{\varepsilon\}$
  - ▷ $[\![a]\!] = \{a\}$
  - ▷ $[\![e^*]\!] = ([\![e]\!])^*$, wobei für bel. Sprachen (Mengen) L gilt: $L^* := \{w_1 \cdots w_k \mid k \geq 0,\ w_i \in L\ \forall i \in \underline{k}\}$
  - ▷ $[\![e_1 | e_2]\!] = [\![e_1]\!] \cup [\![e_2]\!]$
  - ▷ $[\![e_1 \cdot e_2]\!] = [\![e_1]\!] \cdot [\![e_2]\!]$, wobei für bel. Sprachen (Mengen) L_1, L_2 gilt: $L_1 \cdot L_2 := \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$

  Bsp.: $[\![ab^*a]\!] = \{ab^n a \mid n \geq 0\}$

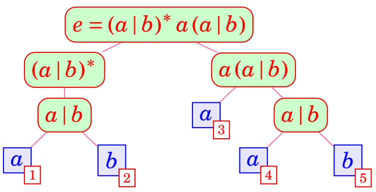
1.1.2 Grundlagen: Endliche Automaten

- **ε-NFA** (nicht-deterministischer endl. Automat mit ε-Übergängen): $A = (Q, \Sigma, \delta, I, F)$ mit:
  - ▷ $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ Übergangs-Relation

  Andere endl. Automaten:
  - ▷ Gibt es keine ε-Übergänge, ist A ein **NFA**.
  - ▷ Ist $\delta : Q \times \Sigma \to Q$ eine Funktion und $|I| = 1$, heißt A deterministisch (**DFA**).
- **Transitiver Abschluss** δ^* von δ ist die kleinste Menge δ' mit, für alle $x \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$:
  - ▷ $(p, \varepsilon, q) \in \delta'$
  - ▷ Wenn $(p, x, p_1) \in \delta' \wedge (p_1, w, q) \in \delta'$ $\Rightarrow (p, xw, q) \in \delta'$
- Satz: $\forall e \in \mathcal{E}_\Sigma$ kann (in lin. Zeit) ein ε-NFA konstruiert werden, der die Sprache $[\![e]\!]$ akzeptiert.

  Konstruktion des ε-NFA für $e \in \mathcal{E}_\Sigma$:

  1. Nummeriere die Blätter. **Bsp.:**



     Alternativ: Verwende $\mathrm{num} : \mathcal{E}_\Sigma \to \mathcal{E}_{\mathbb{N}\times\Sigma}$.

     Bsp.: $\mathrm{num}\left((a|b)^*\, a\, (a|b)\right) = ([1,a] \,|\, [2,b])^*\, [3,a] ([4,a] \,|\, [5,b])$
  2. Zustände: $\bullet r$, $r\bullet$ für alle Knoten r von e

     Anfangszustand: $\bullet e$, Endzustand: $e\bullet$

     Übergangsrelation:
     - ▷ Für Blätter $r = [i, x]$: $(\bullet r, x, r\bullet) \in \delta$
     - ▷ Für Knoten der Form $r = r_1 | r_2$:
       - $(\bullet r,\ \varepsilon,\ \bullet r_1) \in \delta$
       - $(\bullet r,\ \varepsilon,\ \bullet r_2) \in \delta$
       - $(r_1\bullet,\ \varepsilon,\ r\bullet) \in \delta$
       - $(r_2\bullet,\ \varepsilon,\ r\bullet) \in \delta$
     - ▷ Für Knoten der Form $r = r_1 \cdot r_2$:
       - $(\bullet r,\ \varepsilon,\ \bullet r_1) \in \delta$
       - $(r_1\bullet,\ \varepsilon,\ \bullet r_2) \in \delta$
       - $(r_2\bullet,\ \varepsilon,\ r\bullet) \in \delta$
     - ▷ Für Knoten der Form $r = r_1^*$:
       - $(\bullet r,\ \varepsilon,\ r\bullet) \in \delta$
       - $(\bullet r,\ \varepsilon,\ \bullet r_1) \in \delta$
       - $(r_1\bullet,\ \varepsilon,\ \bullet r_1) \in \delta$
       - $(r_1\bullet,\ \varepsilon,\ r\bullet) \in \delta$
     - ▷ Für Knoten der Form $r = r_1?$:
       - $(\bullet r,\ \varepsilon,\ r\bullet) \in \delta$
       - $(\bullet r,\ \varepsilon,\ \bullet r_1) \in \delta$
       - $(r_1\bullet,\ \varepsilon,\ r\bullet) \in \delta$

- **Konstruktion** des NFA *ohne* ε-Übergänge:
  - ▷ $\mathrm{empty}[r] = \mathtt{true} :\Leftrightarrow \varepsilon \in [\![r]\!]$
    - Für Blätter $r = [i, x]$ ist $\mathrm{empty}[r] := (x \equiv \varepsilon)$
    - $\mathrm{empty}[r_1 | r_2] := \mathrm{empty}[r_1] \vee \mathrm{empty}[r_2]$
    - $\mathrm{empty}[r_1 \cdot r_2] := \mathrm{empty}[r_1] \wedge \mathrm{empty}[r_2]$
    - $\mathrm{empty}\left[r_1^*\right] := \mathtt{true}$
    - $\mathrm{empty}[r_1?] := \mathtt{true}$
  - ▷ $\mathrm{first}[r] := \{i \text{ in } r \mid (\bullet r, \varepsilon, \bullet[i,x]) \in \delta^*,\ x \not\equiv \varepsilon\}$
    - Für Blätter $r = [i, x]$ ist $\mathrm{first}[r] := \{i \mid x \not\equiv \varepsilon\}$
    - $\mathrm{first}[r_1 | r_2] := \mathrm{first}[r_1] \cup \mathrm{first}[r_2]$
    - $\mathrm{first}[r_1 \cdot r_2] := \begin{cases} \mathrm{first}[r_1] \cup \mathrm{first}[r_2] & \mathrm{empty}[r_1]{=}\mathtt{true} \\ \mathrm{first}[r_1] & \mathrm{empty}[r_1]{=}\mathtt{false} \end{cases}$
    - $\mathrm{first}\left[r_1^*\right] := \mathrm{first}[r_1]$
    - $\mathrm{first}[r_1?] := \mathrm{first}[r_1]$
  - ▷ $\mathrm{next}[r] := \{i \in \mathbb{N} \mid (r\bullet, \varepsilon, \bullet[i,x]) \in \delta^*,\ x \not\equiv \varepsilon\}$

    Hier wird induktiv andersherum, also von oben, definiert:
    - Für die Wurzel e ist $\mathrm{next}[e] := \emptyset$.
    - Ist der übergeordnete Knoten $r = \ldots$

      $\ldots r_1 | r_2$, dann:
      $\mathrm{next}[r_1] := \mathrm{next}[r]$,
      $\mathrm{next}[r_2] := \mathrm{next}[r]$

      $\ldots r_1 \cdot r_2$, dann:
      $\mathrm{next}[r_1] := \begin{cases} \mathrm{first}[r_2] \cup \mathrm{next}[r] & \mathrm{empty}[r_2]{=}\mathtt{true} \\ \mathrm{first}[r_2] & \mathrm{empty}[r_2]{=}\mathtt{false} \end{cases}$,
      $\mathrm{next}[r_2] := \mathrm{next}[r]$

      $\ldots r_1^*$, dann:
      $\mathrm{next}[r_1] := \mathrm{first}[r_1] \cup \mathrm{next}[r]$

      $\ldots r_1?$, dann:
      $\mathrm{next}[r_1] := \mathrm{next}[r]$
  - ▷ $\mathrm{last}[r] := \{i \text{ in } r \mid ([i,x]\bullet, \varepsilon, r\bullet) \in \delta^*,\ x \not\equiv \varepsilon\}$
    - Für Blätter $r = [i, x]$ ist $\mathrm{last}[r] := \{i \mid x \not\equiv \varepsilon\}$
    - $\mathrm{last}[r_1 | r_2] := \mathrm{last}[r_1] \cup \mathrm{last}[r_2]$
    - $\mathrm{last}[r_1 \cdot r_2] := \begin{cases} \mathrm{last}[r_1] \cup \mathrm{last}[r_2] & \mathrm{empty}[r_2]{=}\mathtt{true} \\ \mathrm{last}[r_2] & \mathrm{empty}[r_2]{=}\mathtt{false} \end{cases}$
    - $\mathrm{last}\left[r_1^*\right] := \mathrm{last}[r_1]$
    - $\mathrm{last}[r_1?] := \mathrm{last}[r_1]$

  Der **Berry-Sethi-/Glushkow-NFA** A_e ist dann:

  Zustände: $\{q_0\} \uplus \{i \in \mathbb{N} \mid i \text{ in Blatt } [i,x],\ x \not\equiv \varepsilon\}$

  Startzustand: $I := \{q_0\}$

  Endzustände: Falls $\mathrm{empty}[e] = \mathtt{false}$, dann $F := \mathrm{last}[e]$; sonst $F := \{q_0\} \uplus \mathrm{last}[e]$.

  Übergänge: $\forall i, i' \in \mathbb{N}$:
  - ▷ $(q_0, x, i) \in \delta$, falls $i \in \mathrm{first}[e]$ und das Blatt i mit x beschriftet ist.
  - ▷ $(i, y, i') \in \delta$, falls $i' \in \mathrm{next}[[i,x]]$, das Blatt i mit x und i' mit y beschriftet ist.
- Satz: Zu jedem NFA $A = (Q, \Sigma, \delta, I, F)$ kann ein DFA $\mathcal{P}(A) = (\mathrm{Pot}(Q), \Sigma, \delta_\mathcal{P}, \{i\}, F_\mathcal{P})$ konstruiert werden mit $\mathcal{L}(A) = \mathcal{L}(\mathcal{P}(A))$.

  Teilmengen-/Potenzmengenkonstruktion:

  Zustände: $\mathrm{Pot}(Q)$ (alle Teilmengen von Q)

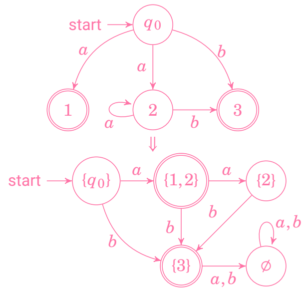
  Startzustand: $i := I \in \mathrm{Pot}(Q)$

  Endzustände: $F_\mathcal{P} := \{Q' \subseteq Q \mid Q' \cap F \neq \emptyset\}$

  Übergangsfunktion:

  $\delta_\mathcal{P}\left(Q', a\right) := \{q \in Q \mid \exists p \in Q' : (p, a, q) \in \delta\}$

  Bsp.:



- $\Rightarrow$ Satz: $\forall e \in \mathcal{E}_\Sigma$ kann ein DFA $A = \mathcal{P}(A_e)$ konstruiert werden mit $\mathcal{L}(A) = [\![e]\!]$.

1.1.3 Design eines Scanners

- Eingabe: eine Menge von Regeln:

  e_1 {$\mathtt{action}_1$}
  e_2 {$\mathtt{action}_2$}
  $\vdots$ $\vdots$
  e_k {$\mathtt{action}_k$}

  mit $k \in \mathbb{N}$, $e_i \in \mathcal{E}_\Sigma$ und $\varepsilon \notin [\![e_i]\!]$ $\forall i \in \underline{k}$.

  Ausgabe: ein Programm, das
  - ▷ von der Eingabe ein *maximales* Präfix w liest, das $e_1 | \ldots | e_k$ erfüllt,
  - ▷ das minimale i ermittelt mit $w \in [\![e_i]\!]$,
  - ▷ für w dann $\mathtt{action}_i$ ausführt.
- **Konstruktion:**
  1. Konstruiere den DFA $\mathcal{P}(A_e)$ (Pot.-A. des Berry-Sethi-NFAs) zu $e := e_1 | e_2 | \ldots | e_k$.
  2. Definiere die Mengen:
     - ▷ $F_1 := \{q \in F \mid q \cap \mathrm{last}[e_1] \neq \emptyset\}$
     - ▷ $F_2 := \{q \in (F \setminus F_1) \mid q \cap \mathrm{last}[e_2] \neq \emptyset\}$

       $\vdots$
     - ▷ $F_k := \{q \in (F \setminus (F_1 \cup \cdots \cup F_{k-1})) \mid q \cap \mathrm{last}[e_k] \neq \emptyset\}$

     Für Eingabe w gilt: $\delta^*_\mathcal{P}(q_0, w) \in F_i \Leftrightarrow$ der Scanner für w $\mathtt{action}_i$ ausführen soll.
- ***Tokenizing*-/Reps' Maximal-Munch-Algorithmus**

  INPUT: String $w \in \Sigma^+$
  $s := 1;\ k := 1$
  FORALL $q \in Q_\mathcal{P},\ i \in |w| + 1$ DO
  $\quad \mathtt{fehlversuch}[q, i] := \mathtt{false}$
  WHILE $\mathtt{true}$ DO
  $\quad q := q_0;\ p := \perp;\ S := \{(q_0, k)\}$
  $\quad$ WHILE $k \leq |w|$ AND $\delta(q, w[k]) \neq \emptyset$ DO
  $\quad\quad$ IF $\mathtt{fehlversuch}[q, k] = \mathtt{true}$ THEN
  $\quad\quad\quad$ BREAK
  $\quad\quad q := \delta(q, w[k]);\ k := k + 1$
  $\quad\quad$ IF $q \in F$ THEN
  $\quad\quad\quad p := q;\ j := k;\ S := \emptyset$
  $\quad\quad$ ELSE $\quad S := S \cup \{(q, k)\}$
  $\quad$ ENDWHILE
  $\quad$ IF $p = \perp$ THEN RETURN$(\mathit{failure})$
  $\quad$ FORALL $(q, k) \in S$ DO
  $\quad\quad \mathtt{fehlversuch}[q, k] := \mathtt{true}$
  $\quad$ let i such that $p \in F_i$
  $\quad \mathtt{write}(w[s, j-1])$
  $\quad \mathtt{action}_i()$
  $\quad s := j;\ k := j$
  $\quad$ IF $j = |w| + 1$ THEN RETURN
  ENDWHILE

- In unterschiedlichen **Scannerzuständen** können verschiedene Token-Klassen erkannt werden, **Bsp.:** Kommentare.

  Eingabe: eine Menge von Regeln:

  Bsp.:

```
<YYINITIAL> {
  /*     {yybegin(COMMENT)}
}
<COMMENT> {
  */     {yybegin(YYINITIAL)}
  .|\n   { }
}
```

1.1.4 Implementierung von DFAs

- Die **Klassifizierungsfunktion** $r: Q \to \mathbb{N}$ ist für $\mathcal{P}(A_e)$ definiert durch $r(q) := \begin{cases} 0 & q \notin F \\ i & q \in F_i \end{cases}$
- Def. Äquivalenzrel. auf Zuständen: $p \equiv_r q :\Leftrightarrow \forall w \in \Sigma^*: r(\delta^*(p,w)) = r(\delta^*(q,w))$

Konstruktion des minimierten DFA für $\mathcal{L}(A)$:
Zustände: $[q]_r \ \ \forall q \in Q$
Anfangszustand: $[q_0]_r$
Klassifizierung: $r([q]_r) := r(q)$
Übergangsfunktion: $\delta([p]_r, a) := [\delta(p,a)]_r$
Dazu: **Konstruktion der Äquiv.-klassen:**

1. $\overline{Q} := \{r^{-1}(i) \mid i \in \mathbb{N},\ r^{-1}(i) \neq \emptyset\}$ (Partition von Q in die Fasern von r)
2. Teste $\forall \overline{q} \in \overline{Q}\ \forall p_1,p_2 \in \overline{q}\ \forall a \in \Sigma$, ob $r(\delta(p_1,a)) \neq r(\delta(p_2,a))$. In dem Fall muss $\overline{q}$ entsprechend aufgeteilt werden.

Naive Implementierung: $\mathcal{O}(n^2)$, Optimierung auf $\mathcal{O}(n \cdot \log n)$ möglich.

- **Konstruktion des minimierten DFA**, alt. Alg.:
  1. Starte Tabelle aller Zustandspaare (treppenförmig, horiz. letzten Zustand weglassen, vert. 1. Zustand weglassen)
  2. Markiere $\{q,q'\}$, falls:
     - ▷ Erkennungsäq. (GTI): $q \in F$, $q' \notin F$
     - ▷ r-Äquivalenz: $r(q) \neq r(q')$
  3. ∀ unmarkierten Paare: teste $\forall a \in \Sigma$, ob $\{\delta(q,a), \delta(q',a)\}$ bereits markiert. Ja ⇒ markiere $\{q,q'\}$.

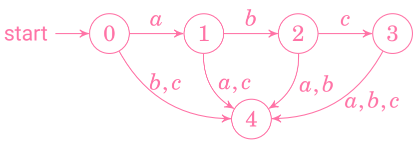
     Bsp.:

2					
3	1	1			
4	1	1			
5	1	1	1	1	
6	2	3	1	1	1
	1	2	3	4	5

1. Schritt
2. Schritt
3. Schritt

  4. Wdh., bis keine Änd. der Tabelle mehr.
  5. ∀ jetzt noch unmark. Paare gilt: $q \equiv q'$
  6. Verschmelze je äquivalente Zustandspaare, vereinige Übergänge der beiden.

- Speichern der Übergangsfunktion als Tabelle indiziert mit $Q \times \Sigma$.
  Bsp.:



δ	0	1	2	3	4
a	1	4	4	4	4
b	4	2	4	4	4
c	4	4	3	4	4

- Reduktion der Tabellengröße
  - ▷ durch Zusammenfassung von Zeichen in Zeichenklassen, Bsp.: `[a-zA-Z_$]`
  - ▷ häufigsten Wert nicht speichern & durch „*default*" ersetzen, Bsp.:

δ	0	1	2	3	4
a	1				
b		2			
c			3		

  - ▷ Übereinanderlegen der Zeilen, Bsp.:

δ	0	1	2
A	1	2	3
valid	a	b	c

Passen die Zeilen nicht direkt übereinander, schieben wir diese nach rechts, bis sie passen, und merken uns die Verschiebung als displacement. Je weniger zusätzliche Spalten nötig sind, desto besser. Bsp.:

δ	0	1	2	3	4
a	1		2	1	
b	3	2		0	

⇓

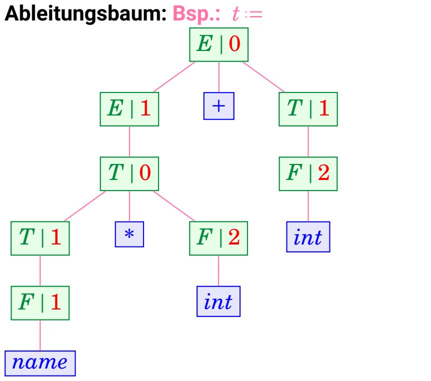
δ	0	1	2	3	4	5
a			1		2	1
b	3	2		0		

displacement$(a) := 2$,
displacement$(b) := 0$

1.2 Syntaktische Analyse

1.2.1 Grundlagen: Kontextfreie Grammatiken

- Def.: **kontextfr. Grammatik:** $G = (N,T,P,S)$
  - ▷ T: endl. Menge der Terminale
  - ▷ P: Menge der Produktionen/Regeln der Form $A \to \alpha$ mit $A \in N$, $\alpha \in (N \cup T)^*$
- Sammle & nummeriere ∀ Nichtterminal A die rechten Seiten: (A,j) ist j-te Regel von A.
  Bsp. KfGl: $E \to E + T^{\ 0} \mid T^{\ 1}$
  $T \to T * F^{\ 0} \mid F^{\ 1}$
  $F \to (E)^{0} \mid name^{\ 1} \mid int^{\ 2}$
- **Ableitungsbaum:** Bsp.: $t :=$

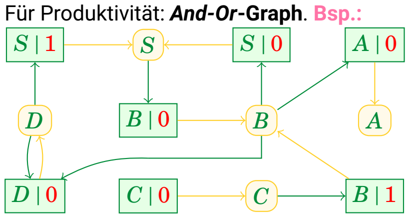


innere Knoten: Regel-Anwendungen,
Wurzel: Regel-Anwendung für A,
Blätter: Terminale oder ε
Kinder eines Knotens (B,i) entsprechen der rechten Seite der i-ten Regel von B.

- **Links-/Rechts-Ableitungen** sind solche, bei denen man stets das linkste bzw. rechteste Vorkommen eines Nichtterminals ersetzt.
  - ▷ entsprechen links-rechts- bzw. rechts-links-*preorder*-DFS-Durchlauf durch den Ableitungsbaum.
  - ▷ **Reverse Rechts-Abl.** entsprechen links-rechts-*postorder*-DFS-Durchlauf.
- Konkatenation der Blätter eines Ableitungsbaums t heißt yield(t).
  Bsp.: yield$(t) = name * int + int$
- Die Grammatik G heißt **eindeutig**, falls es $\forall w \in T^*$ max. einen Ableitungsbaum t von S gibt mit yield$(t) = w$.

1.2.2 Überflüssige Nichtterminale & Regeln

- Def.: $A \in N$ heißt...
  - ▷ **produktiv**, falls $A \to^* w$ für ein $w \in T^*$.
  - ▷ **erreichbar**, falls $S \to^* \alpha A \beta$ für geeignete $\alpha, \beta \in (T \cup N)^*$.
- Für Produktivität: ***And-Or*-Graph**. **Bsp.:**



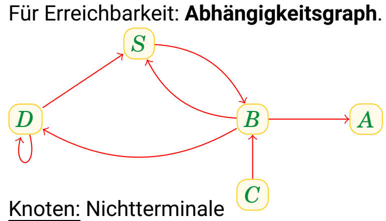
And-Knoten: ■ (nummerierte) Regeln
Or-Knoten: • Nichtterminale
Kanten: ↗ $((B,i),B)$ für alle Regeln (B,i),
↗ $(A,(B,i))$ falls $(B,i) \equiv B \to \alpha_1 A \alpha_2$

Algorithmus für Produktivität:

Subset $\langle N \rangle$ result $:= \emptyset$ (d. h. result $\subseteq N$)
int count$[P]$
Subset $\langle P \rangle$ rhs$[N]$
ForAll $A \in N$ Do rhs$[A] := \emptyset$
ForAll $(A,i) \in P$ Do
  count$[(A,i)] := 0$
  init(A,i)
  ↳ gehe rechte Seite der Produktion (A,i) durch. Für jedes versch. Nichtt. X darin: count$[(A,i)]{+}{+}$ und rhs$[X]$.add$((A,i))$.
EndForAll
Subset $\langle P \rangle$ $W := \{r \mid \text{count}[r] = 0\}$
While $W \neq \emptyset$ Do
  Sei $(A,i) \in W$ beliebig.
  Entferne (A,i) aus W.
  If $A \notin$ result Then
    result $:=$ result $\cup \{A\}$
    ForAll $r \in$ rhs$[A]$ Do
      count$[r]{-}{-}$
      If count$[r] = 0$ Do $W := W \cup \{r\}$
      EndIf
    EndForAll
  EndIf
EndWhile

Laufzeit linear in der Größe der Grammatik.
- ▷ Um den Test „$A \in$ result" einfach zu machen, repräsentiert man die Menge result durch ein Array.
- ▷ W wie auch die Mengen rhs$[A]$ werden dagegen als Listen repräsentiert.

- **Leerheitsproblem:** $\mathcal{L}(G) \neq \emptyset \Leftrightarrow S$ produktiv
- Für Erreichbarkeit: **Abhängigkeitsgraph**. **Bsp.:**



Knoten: Nichtterminale
Kanten: (A,B), falls $B \to \alpha_1 A \alpha_2 \in P$
A ist erreichbar, falls es im Abhängigkeitsgraphen einen Pfad von A nach S gibt.

- Def.: Grammatik G heißt **reduziert**, wenn alle Nichtterminale von G produktiv & erreichbar.
- Satz: Zu jeder kontextfreien Grammatik G mit $\mathcal{L}(G) \neq \emptyset$ kann in linearer Zeit eine reduzierte Gr. G' konstruiert werden mit $\mathcal{L}(G') = \mathcal{L}(G)$.
  Konstruktion der reduzierten Grammatik:
  1. Berechne die Teilmenge $N_1 \subseteq N$ aller produktiven Nichtterminale von G.
     Da $\mathcal{L}(G) \neq \emptyset$, ist insbesondere $S \in N_1$.
  2. Entferne unproduktive Regeln:
     $P_1 := \{A \to \alpha \in P \mid A \in N_1 \wedge \alpha \in (N_1 \cup T)^*\}$
  3. Berechne die Teilmenge $N_2 \subseteq N_1$ aller produktiven & erreichbaren Nichtt. von G.
     Da $\mathcal{L}(G) \neq \emptyset$, ist insbesondere $S \in N_2$.
  4. Entferne unerreichbare Regeln:
     $P_2 := \{A \to \alpha \in P \mid A \in N_2 \wedge \alpha \in (N_2 \cup T)^*\}$

  Ergebnis: $G' := (N_2, T, P_2, S)$

1.2.3 Grundlagen: Kellerautomaten

- Def.: **Kellerautomat** (**PDA**, *push-down automaton*): $M = (Q, T, \delta, q_0, F)$ mit:
  - ▷ Q endl. Menge von Zuständen (bzw. Kellersymbolen),
  - ▷ $q_0 \in Q$ Anfangszustand,
  - ▷ $F \subseteq Q$ Menge der Endzustände,
  - ▷ $\delta \subseteq Q^+ \times (T \cup \{\varepsilon\}) \times Q^*$ endl. Menge von Übergängen (gelesener Kellerinhalt, gel. Eingabe, hinzuzufügender Kellerinhalt)
- Def.: **Konfiguration** des PDA: $(\gamma, w) \in Q^* \times T^*$
  Ein **Berechnungsschritt** des PDA wird durch die Relation $\vdash \subseteq (Q^* \times T^*)^2$ beschrieben, wobei $(\alpha\gamma, xw) \vdash (\alpha\gamma', w)$ für $(\gamma, x, \gamma') \in \delta$.
  - ▷ Relation $\vdash$ hängt vom PDA ab.
  - ▷ Reflexiv-transitive Hülle: $\vdash^*$

- PDA M heißt **deterministisch**, falls $\forall$ paarw. versch. Übergänge $(\gamma_1,x,\gamma_2),(\gamma_1',x',\gamma_2')\in\delta$ gilt: Ist γ_1 Suffix von $\gamma_1' \Rightarrow x\neq x' \wedge x\neq\varepsilon\neq x'$.
- Satz: $\forall$ kontextfr. Grammatik $G=(N,T,P,S)$ kann PDA konstr. werden mit $\mathcal{L}(G)=\mathcal{L}(M)$.

  Konstruktion 1: *Shift-Reduce*-Parser

  ▷ Eingabe sukzessive auf Keller schieben.

  ▷ Liegt oben auf dem Keller eine vollständige rechte Seite (*Handle*), wird dieses durch die zugehörige linke Seite ersetzt (reduziert).

  $M_G^{(1)}=(Q,T,\delta,q_0,\{f\})$ mit

  ▷ $Q:=T\cup N\uplus\{q_0,f\}$

  ▷ $\delta:=\{(q,x,qx)\mid q\in Q,\ x\in T\}\cup$
  $\{(q\alpha,\varepsilon,qA)\mid q\in Q,\ A\to\alpha\in P\}\cup$
  $\{(q_0S,\varepsilon,f)\}$

  Eigenschaften:

  ▷ Folge der Reduktionen entspricht einer reversen Rechts-Abl. für Eingabe.

  ▷ $M_G^{(1)}$ i. A. nicht-deterministisch.

  Konstruktion 2: *Item*-Kellerautomat

  ▷ Rekonstruiere eine Linksableitung.

  ▷ Expand. Nichtterminale mithilfe Regel.

  ▷ Verifiziere sukzessive, dass die gewählte Regel mit der Eingabe übereinstimmt.
  $\Rightarrow$ die Zustände sind jetzt *Items*.

  ▷ Ein *Item* ist eine Regel mit Punkt:
  $[A\to\alpha\bullet\beta],\quad A\to\alpha\beta\in P$
  Der Punkt gibt an, wie weit die Regel bereits abgearbeitet wurde.

  $M_G^{(2)}=(Q,T,\delta,q_0,\{f\})$

  ▷ $Q:=\{\text{alle } Items\}\cup\{[S'\to\bullet S],\ [S'\to S\bullet]\}$
  mit neuem Terminalsymbol S'.

  ▷ $q_0:=[S'\to\bullet S]$

  ▷ $f:=[S'\to S\bullet]$

  ▷ Übergänge:

  Expansionen:
  $([A\to\alpha\bullet B\beta],\varepsilon,[A\to\alpha\bullet B\beta]\,[B\to\bullet\gamma])$
  für $A\to\alpha B\beta$, $B\to\gamma\in P$,

  ***Shifts*:**
  $([A\to\alpha\bullet a\beta],a,[A\to\alpha a\bullet\beta])$
  für $A\to\alpha a\beta\in P$,

  ***Reduce*:**
  $([A\to\alpha\bullet B\beta]\,[B\to\gamma\bullet],\varepsilon,[A\to\alpha B\bullet\beta])$
  für $A\to\alpha B\beta$, $B\to\gamma\in P$,

  Items der Form $[A\to\alpha\bullet]$ heißen **vollständig**.

  Eigenschaften:

  ▷ Expansionen bilden Linksableitung.

  ▷ Expansionen nicht-deterministisch.

1.2.4 Vorausschau-Mengen

- Def.: Für Menge $L\subseteq T^*$ sei $\text{First}_k(L):=$
  $\{u\in L \mid |u|<k\}\cup\{u\in T^k \mid \exists v\in T^*: uv\in L\}$
  die **Menge der Präfixe** der Länge k.
- $\text{First}_k(\_)$ ist verträglich mit Vereinigung und Konkatenation:

▷ $\text{First}_k(\emptyset)=\emptyset$

▷ $\text{First}_k(L_1\cup L_2)=\text{First}_k(L_1)\cup\text{First}_k(L_2)$

▷ $\text{First}_k(L_1\cdot L_2)=$
$\text{First}_k(\text{First}_k(L_1)\cdot\text{First}_k(L_2))$
$=:\text{First}_k(L_1)\odot\text{First}_k(L_2)$

- $T^{\le k}:=\bigcup_{0\le i\le k}T^i,\quad \mathbb{D}_k:=2^{T^{\le k}}$
- $\odot:\mathbb{D}_k\times\mathbb{D}_k\to\mathbb{D}_k$ distributiv in jedem Arg.:

  ▷ $L\odot\emptyset=\emptyset,\quad \emptyset\odot L=\emptyset$

  ▷ $L\odot(L_1\cup L_2)=(L\odot L_1)\cup(L\odot L_2)$

  ▷ $(L_1\cup L_2)\odot L=(L_1\odot L)\cup(L_2\odot L)$
- Für $\alpha\in(N\cup T)^*$ sei
  $\text{First}_k(\alpha):=\text{First}_k(\{w\in T^*\mid\alpha\to^* w\})$
- Für $k\ge1$ gilt:

  ▷ $\text{First}_k(x)=\{x\}$ für $x\in T\cup\{\varepsilon\}$

  ▷ $\text{First}(\alpha_1\alpha_2)=\text{First}_k(\alpha_1)\odot\text{First}_k(\alpha_2)$
- Satz: Die Mengen $\text{First}_k(A)$ für $A\in N$ sind die kleinste Lösung des Ungleichungssystems $\text{First}_k(A)\supseteq\text{First}_k(X_1)\odot\cdots\odot\text{First}_k(X_m)$ für alle $A\to X_1\dots X_m\in P$.

  Berechnung: Fixpunktiteration (s. **??**)

1.2.5 *Top-down Parsing*

- Def.: Eine reduzierte kontextfr. Gr. heißt $\text{LL}(k)$ („***Left-to-right** parsing, **Leftmost** derivation*, Vorausschau der Länge k“), falls für verschiedene Regeln $A\to\alpha$, $A\to\alpha'\in P$ und jede Linksableitung $S\to_L^* uA\beta$ mit $u\in T^*$ gilt: $\text{First}_k(\alpha\beta)\cap\text{First}_k(\alpha'\beta)=\emptyset$.

  Bsp. KfG2: $S\to \mathit{if}\ (E)\ S\ \mathit{else}\ S \mid \mathit{while}\ (E)\ S \mid E;$
  $E\to \mathit{id}$
  ist $\text{LL}(1)$.

  Bsp. KfG3: $S\to \mathit{if}\ (E)\ S\ \mathit{else}\ S \mid \mathit{if}\ (E)\ S \mid \mathit{while}\ (E)\ S \mid E;$
  $E\to \mathit{id}$
  ist nicht $\text{LL}(k)$ für jedes $k>0$, denn: Betrachte Regeln $(S,0),(S,1)$, d. h. $\alpha=\mathit{if}\ (E)\ S\ \mathit{else}\ S$ und $\alpha'=\mathit{if}\ (E)\ S$, und die Linksableitung $S\to_L^*\mathit{if}\ (\mathit{id})\ S$, d. h. $\beta=\varepsilon$. Es gilt offensichtl.: $\text{First}_k(\alpha\beta)\cap\text{First}_k(\alpha'\beta)\supseteq\{\mathit{if}\}\neq\emptyset\ \forall k>0$.
- $\textbf{LL}(\boldsymbol{k})$**-Parser:** sieht Fenster der Länge k der Eingabe; realisiert i. W. *Item*-Kellerautomaten, tabelliert nächste zu wählende Regeln.

  Bsp.: bzgl. KfG2:

  Zustände: *Items*

Tabelle:	if	while	id
$[\dots\to\dots\bullet S\dots]$	0	1	2
$[\dots\to\dots\bullet E\dots]$	–	–	0

- Menge der möglichen nächsten k Zeichen ist: $\text{First}_k(\alpha\beta)=\text{First}_k(\alpha)\odot\text{First}_k(\beta)$, wobei:
  α die rechte Seite der passenden Regel,
  β ein möglicher rechter Kontext von A.
  $\text{First}_k(\beta)$ dynamisch akkumulieren:
- Ein **erweitertes *Item*** ist ein Paar $[A\to\alpha\bullet\gamma,L]$ für $A\to\alpha\gamma\in P$, $L\subseteq T^{\le k}$.
  L wird als Vorausschau-Menge zum Repräsentieren von $\text{First}_k(\beta)$ für den rechten Kontext β von A verwendet.
- **Erweiterter *Item*-Kellerautomat:**

  Zustände: erweiterte *Items*

  Anfangszustand: $[S'\to\bullet S,\{\varepsilon\}]$

  Endzustand: $[S'\to S\bullet,\{\varepsilon\}]$

  Übergänge:

  Expansionen:
  $([A\to\alpha\bullet B\beta,L],\varepsilon,$
  $[A\to\alpha\bullet B\beta,L]\,[B\to\bullet\gamma,\text{First}_k(\beta)\odot L])$
  für $A\to\alpha B\beta$, $B\to\gamma\in P$,

  ***Shifts*:**
  $([A\to\alpha\bullet a\beta,L],a,[A\to\alpha a\bullet\beta,L])$
  für $A\to\alpha a\beta\in P$,

  ***Reduce*:**
  $([A\to\alpha\bullet B\beta,L]\,[B\to\gamma\bullet,L'],\varepsilon,$
  $[A\to\alpha B\bullet\beta,L])$ für $A\to\alpha B\beta$, $B\to\gamma\in P$
- **Vorausschau-Tabelle:**
  $$M\Big[[A\to\alpha\bullet B\beta,L],w\Big]:=\Big\{i\ \Big|\ (B,i)=(B\to\gamma),\ w\in\text{First}_k(\gamma)\odot\text{First}_k(\beta)\odot L\Big\}$$
- Satz: reduzierte kontextfr. Gr. G ist $\text{LL}(k)$ $\Leftrightarrow$ $\forall$ Eingabewort zu jedem Zeitpunkt in der Berechnung des erw. *Item*-Kellerautomaten $\big|M\big[[A\to\alpha\bullet B\beta,L],w\big]\big|\le1$ gilt.

  Hierbei ist $[A\to\alpha\bullet B\beta,L]$ das aktuelle oberste Kellersymbol und w besteht aus den nächsten k Zeichen der Eingabe (bzw. $k'<k$ Zeichen, falls Rest der Eingabe nur noch k' lang).
- erw. *Item*-Kellerautomat mit k-Vorausschau-tabelle erlaubt deterministische Rekonstruktion einer Linksableitung für $\text{LL}(k)$-Grammatik.

  Bsp. KfG4: $S\to\varepsilon\mid aSb$

Vorausschau-Tabelle:	ε	a	b
$[S'\to\bullet S,\{\varepsilon\}]$	0	1	–
$[S\to a\bullet Sb,\{\varepsilon\}]$	–	1	0
$[S\to a\bullet Sb,\{b\}]$	–	1	0

- Hängt auszuwählende Regel nicht von den Erweiterungen der *Items* ab, kann man den *Item*-Kellerautomat auch ohne Erw. benutzen.

  Hängt auszuw. Regel nur von der aktuellen Vorausschau w ab, heißt G auch **stark** $\textbf{LL}(\boldsymbol{k})$.

  Bsp.: Die Grammatik aus KfG4 ist anhand der Vorausschau-Tabelle offensichtl. stark $\text{LL}(k)$.
- Def.: $\text{Follow}_k(A):=\bigcup\{\text{First}_k(\beta)\mid S\to_L^* uA\beta\}$
- Def.: Reduzierte kontextfr. Grammatik G heißt **stark** $\textbf{LL}(\boldsymbol{k})$, falls für je zwei verschiedene Regeln $A\to\alpha, A\to\alpha'\in P$ gilt:
  $\text{First}_k(\alpha)\odot\text{Follow}_k(A)\cap$
  $\text{First}_k(\alpha')\odot\text{Follow}_k(A)=\emptyset$

  Bsp.: KfG4 ist tatsächlich stark $\text{LL}(k)$, denn:
  $\text{Follow}_1(S)=\{\varepsilon,b\}\Rightarrow$
  $\text{First}_1(\varepsilon)\odot\text{Follow}_1(S)=\{\varepsilon\}\odot\{\varepsilon,b\}=\{\varepsilon,b\}$
  $\text{First}_1(aSb)\odot\text{Follow}_1(S)=\{a\}\odot\{\varepsilon,b\}=\{a\}$
- Ist G stark $\text{LL}(k)$, können wir die Vorausschau-Tabelle mit Nichtterminalen (statt erw. *Items*) indizieren:
  $$M[B,w]:=\begin{cases} i & (B,i)=(B\to\gamma)\wedge\\ & w\in\text{First}_k(\gamma)\odot\text{Follow}_k(B)\\ - & \text{keine solche Regel existiert}\end{cases}$$

  Bsp.: bzgl. KfG4:

	ε	a	b
S	0	1	0

- Satz:

  ▷ Jede stark-$\text{LL}(k)$-Gr. ist auch $\text{LL}(k)$.
  Gilt i. A. nicht andersherum! Ausnahme:

  ▷ Jede $\text{LL}(1)$-Gr. ist bereits stark $\text{LL}(1)$.
- Zu jeder $\text{LL}(k)$-Grammatik kann eine äquivalente starke $\text{LL}(k)$-Gr. konstruiert werden.
- **Berechnung von** $\textbf{Follow}_{\boldsymbol{k}}(\boldsymbol{B})$**:**

  Stelle Ungleichungssystem auf:
  $\text{Follow}_k(S)\supseteq\{\varepsilon\}$ (gilt immer)
  $\text{Follow}_k(B)\supseteq$
  $\text{First}_k(X_1)\odot\text{First}_k(X_m)\odot\text{Follow}_k(A)$
  für $A\to\alpha BX_1\dots X_m\in P$

  Kleinste Lösung ist $\text{Follow}_k(B)$.
- Größe der auftretenden Mengen steigt mit k rapide. In praktischen Systemen wird darum meist nur der Fall $k=1$ implementiert.

1.2.6 Schnelle Berechnung von Vorausschau-Mengen

- Im Fall $k=1$ lassen sich First_1 und Follow_1 besonders effizient berechnen.
- Seien $L_1,L_2\subseteq T\cup\{\varepsilon\}$ mit $L_1\neq\emptyset\neq L_2$. Dann ist:
  $$L_1\odot L_2=\begin{cases}L_1 & \varepsilon\notin L_1\\ (L_1\setminus\{\varepsilon\})\cup L_2 & \text{sonst}\end{cases}$$
- Ist G reduziert, so ist $\forall A:\text{First}_1(A)\neq\emptyset$.
- Def.: $\text{empty}(X)=\texttt{true}:\Leftrightarrow X\to^*\varepsilon$
- Def.: die $\boldsymbol{\varepsilon}$**-freien** $\textbf{First}_{\mathbf{1}}$**-Mengen** seien:
  $\text{F}_\varepsilon(a):=\{a\}$ für $a\in T$,
  $\text{F}_\varepsilon(A):=\text{First}_1(A)\setminus\{\varepsilon\}$ für $A\in N$.
- **Berechnung:** Ungleichungssystem für $\text{F}_\varepsilon(A)$:

  $\text{F}_\varepsilon(A)\supseteq\text{F}_\varepsilon(X_j)$
  falls $A\to X_1\dots X_m\in P$, $j\in\underline{m}$,
  $\text{empty}(X_1)\wedge\cdots\wedge\text{empty}(X_{j-1})$.

  Bsp.: bzgl. Grammatik aus KfG1:
  $\text{empty}(E)=\text{empty}(T)=\text{empty}(F)=\texttt{false}$
  Ungleichungssystem:

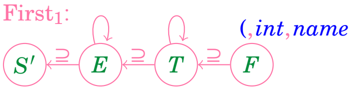
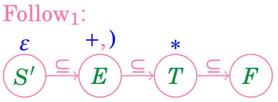
  ▷ $\text{F}_\varepsilon(S')\supseteq\text{F}_\varepsilon(E)$

  ▷ $\text{F}_\varepsilon(E)\supseteq\text{F}_\varepsilon(E)$

  ▷ $\text{F}_\varepsilon(E)\supseteq\text{F}_\varepsilon(T)$

- ▷ $F_\varepsilon(T) \supseteq F_\varepsilon(T)$
- ▷ $F_\varepsilon(T) \supseteq F_\varepsilon(F)$
- ▷ $F_\varepsilon(F) \supseteq \{(, name, int\}$

- **Ungleichungssystem zu $\mathbf{Follow_1(A)}$:**
  $\mathrm{Follow}_1(S) \supseteq \{\varepsilon\}$
  $\mathrm{Follow}_1(B) \supseteq F_\varepsilon(X_j)$
  falls $A \to \alpha B X_1 \dots X_m \in P$,
  $\mathrm{empty}(X_1) \wedge \dots \wedge \mathrm{empty}(X_{j-1})$
  $\mathrm{Follow}_1(B) \supseteq \mathrm{Follow}_1(A)$
  falls $A \to \alpha B X_1 \dots X_m \in P$,
  $\mathrm{empty}(X_1) \wedge \dots \wedge \mathrm{empty}(X_m)$
  Bsp.: bzgl. Grammatik aus KfG1:
  - ▷ $\mathrm{Follow}_1(S') \supseteq \{\varepsilon\}$
  - ▷ $\mathrm{Follow}_1(E) \supseteq \mathrm{Follow}_1(S')$
  - ▷ $\mathrm{Follow}_1(E) \supseteq \{+,)\}$
  - ▷ $\mathrm{Follow}_1(T) \supseteq \{*\}$
  - ▷ $\mathrm{Follow}_1(T) \supseteq \mathrm{Follow}_1(E)$
  - ▷ $\mathrm{Follow}_1(F) \supseteq \mathrm{Follow}_1(T)$

- Die Form dieser Ungleichungssysteme ist:
  $x \sqsupseteq y$ bzw. $x \sqsupseteq d$
  für Variablen x, y und $d \in \mathbb{D}$. Solche Systeme heißen **reine Vereinigungsprobleme** und sind lösbar mit linearem Aufwand.
  Berechnung:
  - ▷ Konstruiere den **Variablen-Abhängigkeitsgraph** zum Ungleichungssystem.
    Bsp.: bzgl. KfG1 (s. o. für deren Ungl.-S.):
    First_1:
    
    Follow_1:
    
  - ▷ Innerhalb einer **starken Zusammenhangskomponente** (SZK), d. h. einem maximalen Teilgraph, in dem man von jedem zu jedem Knoten kommen kann, haben alle Variablen den gleichen Wert.
  - ▷ Hat eine SZK keine eingehenden Kanten, erhält man ihren Wert, indem man alle Werte innerhalb der SZK vereinigt.
  - ▷ Gibt es eingehende Kanten, muss man zusätzlich die Werte an deren Startknoten hinzufügen.

1.2.7 *Bottom-up*-Analyse

- Viele Grammatiken sind nicht $\mathrm{LL}(k)$!
  Wie parsen wir also diese?
- Def.: Grammatik G heißt **links-rekursiv**, falls $A \to^+ A\beta$ für ein $A \in N$, $\beta \in (T \cup N)^*$.
  Bsp.: KfG1 ist links-rekursiv.
- Satz: Ist G reduziert & links-rekursiv, dann ist G nicht $\mathrm{LL}(k)$ $\forall k$.
- $\alpha\gamma$ ist **zuverlässiges Präfix** für das vollst. *Item* $[B \to \gamma \bullet]$, wenn folgende Berechnung von $M_G^{(1)}$ existiert:
  $(q_0 \alpha\gamma, v) \vdash (q_0 \alpha B, v) \vdash^* (q_0 S, \varepsilon)$
  Dies ist der Fall genau dann, wenn $S \to_R^* \alpha B v$.
- Das *Item* $[B \to \gamma \bullet \beta]$ heißt **gültig** für α' $\Leftrightarrow$ $S \to_R^* \alpha B v$ mit $\alpha' = \alpha\gamma$.
- **Berechnung der Menge zuverlässiger Präfixe** durch NFA, den **charakterist. Automat** $c(G)$:
  Zustände: *Items*
  Anfangszustand: $[S' \to \bullet S]$
  Endzustände: $\{[B \to \gamma \bullet] \mid B \to \gamma \in P\}$
  Übergänge:
  1. $([A \to \alpha \bullet X\beta], X, [A \to \alpha X \bullet \beta])$
     für $X \in (N \cup T)$, $A \to \alpha X \beta \in P$
  2. $([A \to \alpha \bullet B\beta], \varepsilon, [B \to \bullet\gamma])$
     für $A \to \alpha B\beta, B \to \gamma \in P$

- Den **kanonischen $\mathbf{LR(0)}$-Automaten** $\mathrm{LR}(G)$ erhält man aus $c(G)$, indem man ε-Übergänge entfernt & Teilmengenkonstruktion anwendet.
  Alternative Konstruktion von $\mathbf{LR(G)}$:
  Hilfsfunktion $\delta_\varepsilon^*(q) := q \cup$
  $\left\{[B \to \bullet\gamma] \;\middle|\; \exists [A \to \alpha \bullet B'\beta'] \in q, \beta \in (N \cup T)^* : B' \to^* B\beta\right\}$
  Dann, DFA:
  Zustände: Mengen von *Items*
  Anfangszustand: $\delta_\varepsilon^* \{[S' \to \bullet S]\}$
  Endzustände: $\{q \mid \exists A \to \alpha \in P : [A \to \alpha \bullet] \in q\}$
  Übergänge: $\delta(q, X) := \delta_\varepsilon^* \left\{[A \to \alpha X \bullet \beta] \;\middle|\; [A \to \alpha \bullet X\beta] \in q\right\}$

© 2020 study.woalk.de