

Aufgaben und Lösungen

**LÜB INFORMATIK / Band 2 – Theorie der Informatik 2. Auflage
Kapitel 6: Übersetzung von Programmiersprachen**

ABSCHNITT 6.4.6 / Übungen

Aufgabe 6.1

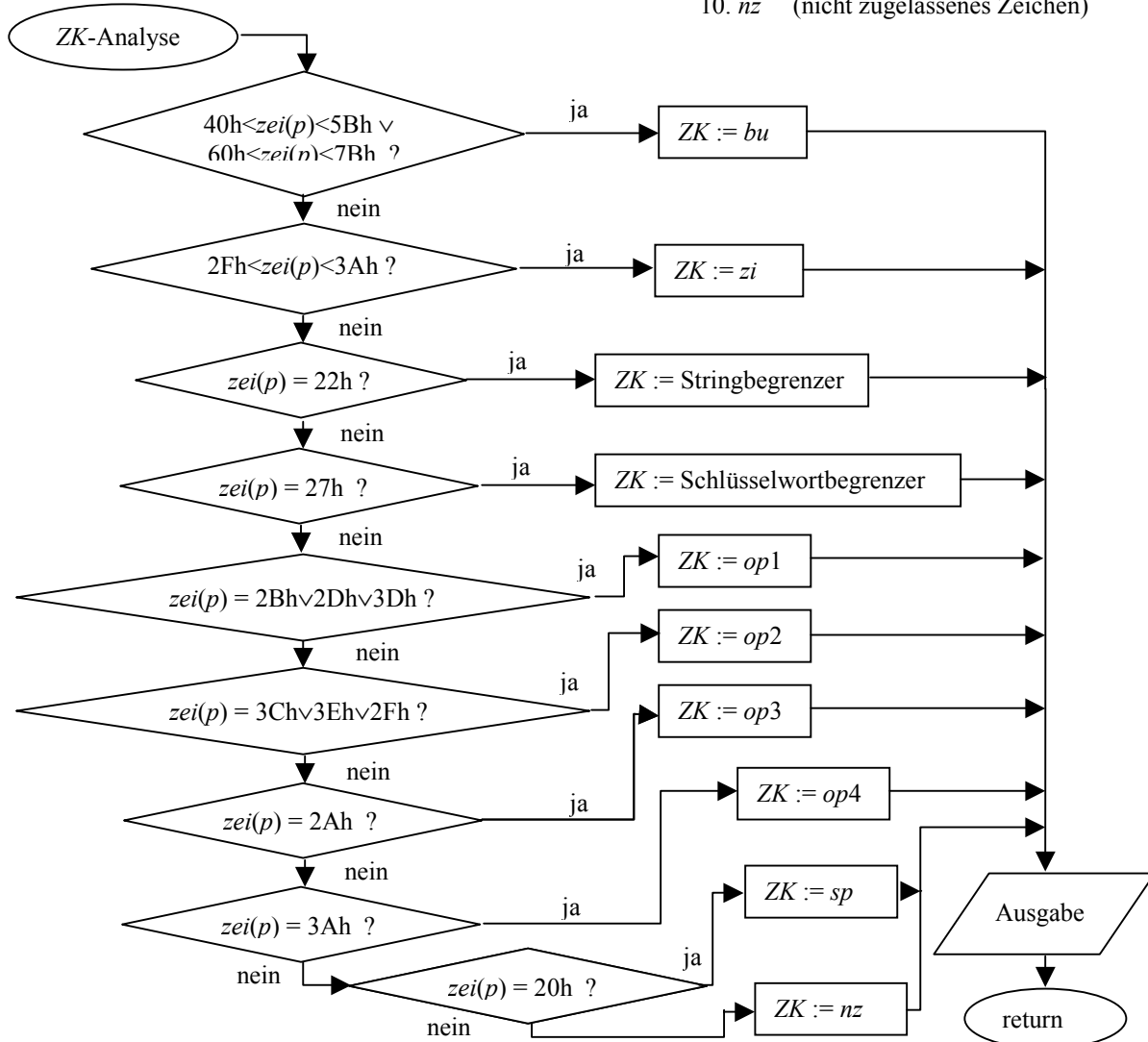
Erarbeiten Sie ein Programm, daß den ASCII-Zeichensatz in Zeichenklassen gemäß Bild 6.4 einordnet.

Statt eines Programms wird der Lösungsvorschlag als Algorithmus angegeben!

Es existiere der Analysesatz $s \perp$ mit $s \in \text{ASCII}^*$ und $|s| = n$. Die aktuelle Zeichenposition in s sei p , das aktuelle Zeichen selbst ist $zei(p)$. Die Operatoren bilden unterschiedliche Klassen ZK:

Folgende Zeichenklassen werden bestimmt:

- | | | |
|---------------------------|------------------------------------|--|
| 1. <i>bu</i> (Buchstaben) | 4. ' (Schlüsselwortbegrenzer) | 7. <i>op3</i> (Multiplikation, Potenz) |
| 2. <i>zi</i> (Ziffern) | 5. <i>op1</i> (Operatoren +, -, =) | 8. <i>op4</i> (Labelende :) |
| 3. " (Stringbegrenzer) | 6. <i>op2</i> (Operatoren <, >, /) | 9. <i>sp</i> (Leerzeichen) |
| | | 10. <i>nz</i> (nicht zugelassenes Zeichen) |

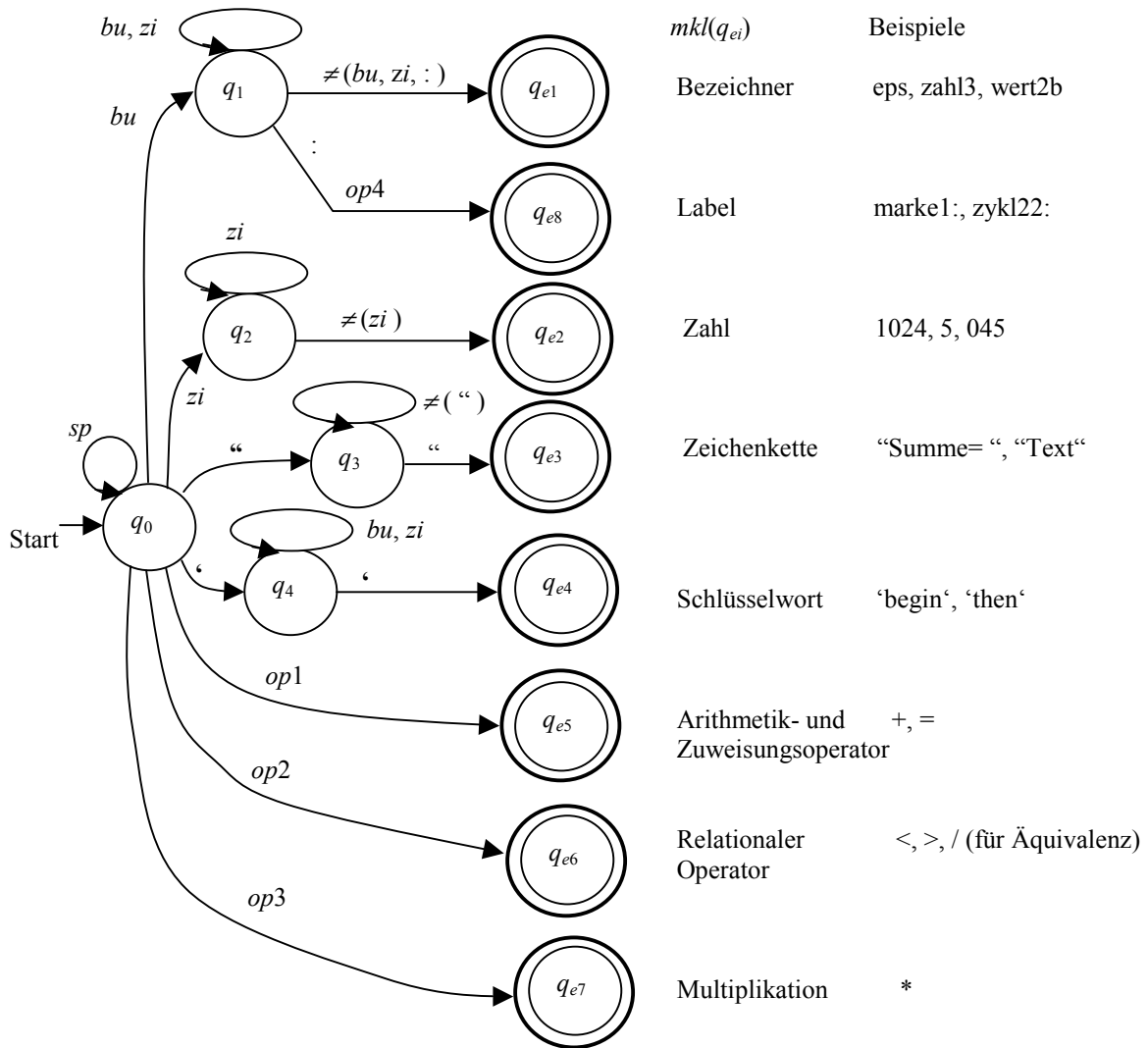


Aufgabe 6.2

Erstellen Sie eine vollständige Automatentabelle für den im Bild 6.5 angegebenen Akzeptor.

Der im Bild 6.5 angegebene Algorithmus realisiert einen Morphemerkenner (Worterkenner), der die Morpheme MO in Zusammenarbeit mit der Prozedur "ZK-Analyse" zeichenweise aufbaut und die Morphemklasse $mkl(q_{ei})$ als Attribut ausgibt. Dabei ist q_{ei} einer von i möglichen und damit das Attribut kennzeichnenden Endzuständen. Das Erreichen eines Endzustandes q_{ei} bedeutet die Akzeptanz eines mit dem Graphen beschriebenen gültigen Morphems.

Morphemgraph: $\neq (... , ... , ...)$ bedeutet: Zeichenklasse ZK ist verschieden von (...) oder (...) usw.



Automatentabelle:

Alle Endzustände q_{ei} sowie auch alle in der Automatentabelle definierbaren Fehlerzustände q_f veranlassen die Ausgabe des jeweils aufgebauten Morphems bzw. ein Fehlersignal. Die beiden Zustände q_{ei} und q_f werden deshalb in der folgenden Tabelle nicht als aktuelle Zustände AZ mit aufgenommen.

ZK AZ	<i>bu</i>	<i>zi</i>	“	‘	<i>op1</i>	<i>op2</i>	<i>op3</i>	<i>op4</i>	<i>sp</i>
q_0	q_1	q_2	q_3	q_4	q_{e5}	q_{e6}	q_{e7}	q_f	q_0
q_1	q_1	q_1	q_{e1}	q_{e1}	q_{e1}	q_{e1}	q_{e1}	q_{e8}	q_{e1}
q_2	q_{e2}	q_2	q_{e2}	q_{e2}	q_{e2}	q_{e2}	q_{e2}	q_{e2}	q_{e2}
q_3	q_3	q_3	q_{e3}	q_f	q_3	q_3	q_3	q_3	q_{e3}
q_4	q_4	q_4	q_f	q_{e4}	q_f	q_f	q_f	q_f	q_{e4}

Aufgabe 6.3

Programmieren Sie einen Scanner nach Bild 6.5 und erproben Sie die lexikalische Analyse an dem Satz:
 ‘if a1<3E-5 ‘then‘ x:=a1 ‘else‘ ‘print‘ “ende“.

Es wird empfohlen, den Analysesatz ein wenig zu vereinfachen, um die lexikalische Analyse auf Grund der Lösungen zu den Aufgaben 6.1 und 6.2 nachvollziehen zu können.

Die Programmierung wird hier durch eine Ablaufanalyse ersetzt, die mit dem im Bild 6.5 dargelegten Algorithmus nachvollzogen werden kann und auch vervollständigt werden sollte!

(Die Positionen der Leerzeichen bitte dem oben angegebenen ‘Originalsatz‘ entnehmen).

Startsituation: $s = \text{‘if‘ } a1 < 3E-5 \text{ ‘then‘ } x := a1 \text{ ‘else‘ ‘print‘ “ende“}$

\uparrow
 $P(p), p=1, MO = \{\}$ (leere Menge)

1. Zyklus: $AZ = q_0, p = 1$

$Z(p) := \text{‘}$
 $ZK :=$ Schlüsselwortbegrenzer (s. Aufgabe 6.1)
 $q = q_0$; weil $q_0 \notin F$ ist, folgt $MO := \{\}$.
 $AZ := q_4$ (s. Aufgabe 6.2) und $p := 2$

2. Zyklus:

$Z(p) := i$
 $ZK := bu$ (s. Aufgabe 6.1)
 $q = q_4$; weil $q_4 \notin F$ ist, folgt $MO := \{‘i\}$.
 $AZ := q_4$ (s. Aufgabe 6.2) und $p := 3$

3. Zyklus:

$Z(p) := f$
 $ZK := bu$ (s. Aufgabe 6.1)
 $q = q_4$; weil $q_4 \notin F$ ist, folgt $MO := \{‘if\}$.
 $AZ := q_4$ (s. Aufgabe 6.2) und $p := 4$

4. Zyklus:

$Z(p) := \text{‘}$
 $ZK :=$ Schlüsselwortbegrenzer (s. Aufgabe 6.1)
 $q = q_{e4}$; weil $q_4 \notin F$ ist, folgt $MO := \{‘if‘\}$.
 $AZ := q_4$ (s. Aufgabe 6.2) und $p := 5$

5. Zyklus:

$Z(p) := sp$
 $ZK :=$ Leerzeichen (s. Aufgabe 6.1)
 $q = q_{e4}$; weil $q_{e4} \in F$ ist, folgt $mkl(q_{e4}) := 4$ (Nummer des erreichten Endzustands)
 Morphemende erreicht, d.h., MO wird ausgegeben.

Das in MO vollständig vorliegende und mit $mkl(q_{e4})$ klassifizierte Morphem wird unter der Regie des Übersetzers in entsprechende Listen eingetragen, sofern es sich um Bezeichner oder Labels handelt.

Schlüsselwörter müssen aktive Programmelemente in die Morphemfolge einbinden, sie werden daher durch neue Symbole (Tokens) ersetzt, die dann später separat im Syntax-Analysebaum auftreten. Gleiches gilt z.B. für erkannte Zahlen.

Aufgabe 6.4

Berechnen Sie für die im Abschnitt 3.4.2 angegebene Grammatik G_A die *FIRST*- und *FOLLOW*-Mengen (s. Seite 96).

$$G_A = \{N_A, T_A, R_A, \sigma\} \text{ mit } N_A = \{A, T, F, E\}, T_A = \{+, -, *, /, \uparrow, (,), v, z\}, \sigma = A \text{ und}$$

$$R_A = \{(A, T), (A, +T), (A, -T), (A, A+T), (A, A-T), (T, F), (T, T*F), (T, T/F), (F, E), (F, F\uparrow E), (E, v), (E, z), (E, (A))\}.$$

1. $FIRST(G_A)$:

Die Relation R_{anf} lt. Gl. (6.1) besteht aus den Paaren (x, y) , welche aus allen Produktionsregeln von G_A zu bilden sind. Dabei sind die x alle linken Regelseiten und die y jeweils das erste Zeichen auf der rechten Regelseite. Eine Regel $x \rightarrow \epsilon$ erzeugt kein Paar (x, y) .

$$R_{anf} = \{(A, T), (A, +), (A, -), (A, A), (T, F), (T, T), (F, E), (F, F), (E, v), (E, z), (E, ())\}.$$

Die Bestimmung der transitiven Hülle R_{anf}^+ (s. Lösung zur Aufgabe 3.2) wird

$$R_{anf}^1 = R_{anf}$$

$$R_{anf}^2 = R_{anf}^1 \circ R_{anf} = \{(A, F), (A, T), (A, +), (A, -), (A, A), (T, E), (T, F), (T, T), (F, v), (F, z), (F, ()), (F, E), (F, F)\}.$$

Die in R_{anf}^2 neu hinzugekommenen Paare gegenüber R_{anf}^1 wurden in R_{anf}^2 unterstrichen. Bei den weiteren Berechnungsschritten wird zwischen aufeinanderfolgenden Iterationen ebenso verfahren.

$$R_{anf}^3 = R_{anf}^2 \circ R_{anf} = \{(A, E), (A, F), (A, T), (A, +), (A, -), (A, A), (T, v), (T, z), (T, ()), (T, E), (T, F), (T, T), (F, v), (F, z), (F, ()) (F, E), (F, F)\}.$$

$$R_{anf}^4 = R_{anf}^3 \circ R_{anf} = \{(A, v), (A, z), (A, ()), (A, E), (A, F), (A, T), (A, +), (A, -), (A, A), (T, v), (T, z), (T, ()), (T, E), (T, F), (T, T), (F, v), (F, z), (F, ()) (F, E), (F, F)\}.$$

$$R_{anf}^5 = R_{anf}^4 \circ R_{anf} = \{(A, v), (A, z), (A, ()), (A, E), (A, F), (A, T), (A, +), (A, -), (A, A), (T, v), (T, z), (T, ()), (T, E), (T, F), (T, T), (F, v), (F, z), (F, ()) (F, E), (F, F)\}.$$

R_{anf}^5 liefert keine neuen Paare und damit bricht die Iteration ab. Mit $R_{anf}^+ = R_{anf}^1 \cup R_{anf}^2 \cup R_{anf}^3 \cup R_{anf}^4$ wird

$$R_{anf}^+ = \{(A, T), (A, +), (A, -), (A, A), (T, F), (T, T), (F, E), (F, F), (E, v), (E, z), (E, ()), (A, F), (T, E), (F, v), (F, z), (F, ()), (A, E), (T, v), (T, z), (T, ()), (A, v), (A, z), (A, ())\}.$$

Die Gl. (6.2) liefert mit $FIRST(G_A) = (R_{anf}^+ \cap (N_A \times T_A)) \cup \{(t, t) \mid t \in T_A\}$ die Menge

$$FIRST(G_A) = \{(A, +), (A, -), (A, v), (A, z), (A, ()), (T, v), (T, z), (T, ()), (E, v), (E, z), (E, ()), (F, v), (F, z), (F, ()), (+, +), (-, -), (*, *), (/), (\uparrow), (\uparrow), ((, ()), ((, ()), (v, v), (z, z))\}.$$

2. $FOLLOW(G_A)$:

Die Gl. (6.3) liefert die Relation R_{nach} zu

$$R_{nach} = \{(T, A), (F, T), (E, F)\}.$$

Mit der Gl. (6.4) gewinnt man

$$R_{nachT} = \{(A, +), (A, -), (T, *), (T, /), (F, \uparrow), (A,))\}.$$

Die reflexiv transitive Hülle von R_{nach} ist

$$R_{nach}^* = R_{nach}^0 \cup R_{nach}^1 \cup R_{nach}^2 \cup R_{nach}^3$$

$$= \underbrace{\{(A, A), (T, T), (F, F), (E, E)\}}_{R_{nach}^0} \cup \underbrace{\{(T, A), (F, T), (E, F)\}}_{R_{nach}^1} \cup \underbrace{\{(F, A), (E, T), (E, A)\}}_{R_{nach}^2} \cup \underbrace{\{(E, A)\}}_{R_{nach}^3}$$

$$FOLLOW(G_A) = \{(A, +), (A, -), (A, ()), (T, *), (T, /), (F, \uparrow), (T, +), (T, -), (T, ()), (F, *), (F, /), (E, \uparrow), (F, +), (F, -), (F, ()), (E, *), (E, /), (E, +), (E, -), (E, ())\}.$$

Aufgabe 6.5

Überführen Sie die im Abschnitt 3.4.2 angegebene linksrekursive Grammatik G_A in ihre äquivalente linksrekursionsfreie Form G_A' .

In G_A (s. Seite 96) gibt es die Regeln:

$$\begin{array}{llllll}
 A \rightarrow T & (1) & A \rightarrow A + T & (4) & T \rightarrow T * F & (7) & F \rightarrow E & (9) & E \rightarrow z & (12) \\
 A \rightarrow +T & (2) & A \rightarrow A - T & (5) & T \rightarrow T / F & (8) & F \rightarrow F \uparrow E & (10) & E \rightarrow (A) & (13) \\
 A \rightarrow -T & (3) & T \rightarrow F & (6) & F \rightarrow E & (9) & E \rightarrow v & (11) & &
 \end{array}$$

Folgende Regeln sind linksrekursiv : (4), (5), (7), (8) und (10).

Zur Beseitigung der Linksrekursivitäten müssen neue, nicht zu G_A gehörende Nichtterminale N^1 eingeführt werden.. Dieses sind $N^1 = \{V, W, X, Y, Z\}$.

Die Regel (4) wird unter Einbeziehung der Regel (1) linksrekursionsfrei durch:

$$\left. \begin{array}{l} A \rightarrow A + T \\ A \rightarrow T \end{array} \right\} \begin{array}{l} A \rightarrow TV \\ V \rightarrow +TV \\ V \rightarrow \varepsilon \end{array}$$

Die Regel (5) wird unter Einbeziehung der Regel (1) linksrekursionsfrei durch:

$$\left. \begin{array}{l} A \rightarrow A - T \\ A \rightarrow T \end{array} \right\} \begin{array}{l} A \rightarrow TW \\ W \rightarrow -TW \\ W \rightarrow \varepsilon \end{array}$$

Die Regel (7) wird unter Einbeziehung der Regel (6) linksrekursionsfrei durch:

$$\left. \begin{array}{l} T \rightarrow T * F \\ T \rightarrow F \end{array} \right\} \begin{array}{l} T \rightarrow FX \\ X \rightarrow *FX \\ X \rightarrow \varepsilon \end{array}$$

Die Regel (8) wird unter Einbeziehung der Regel (6) linksrekursionsfrei durch:

$$\left. \begin{array}{l} T \rightarrow T / F \\ T \rightarrow F \end{array} \right\} \begin{array}{l} T \rightarrow FY \\ Y \rightarrow /FY \\ Y \rightarrow \varepsilon \end{array}$$

Die Regel (10) wird unter Einbeziehung der Regel (9) linksrekursionsfrei durch:

$$\left. \begin{array}{l} F \rightarrow F \uparrow E \\ F \rightarrow E \end{array} \right\} \begin{array}{l} F \rightarrow EZ \\ Z \rightarrow \uparrow EZ \\ Z \rightarrow \varepsilon \end{array}$$

Daraus ergibt sich die äquivalente linksrekursionsfreie Grammatik $G_A' = (N_A', T_A', \sigma, R_A')$ zu:

$N_A' = N_A \cup N^1 = \{A, T, F, G, V, W, X, Y, Z\}$, $T_A' = \{+, -, *, /, \uparrow, (,), v, z\} \cup \{\varepsilon\}$, $\sigma = A$ und

$$\begin{array}{llllll}
 R_A' = \{ & A \rightarrow T & T \rightarrow F & E \rightarrow v & V \rightarrow +TV & X \rightarrow *FX & Z \rightarrow \uparrow EZ \\
 & A \rightarrow +T & T \rightarrow FX & E \rightarrow z & V \rightarrow \varepsilon & X \rightarrow \varepsilon & Z \rightarrow \varepsilon \}. \\
 & A \rightarrow -T & T \rightarrow FY & E \rightarrow (A) & W \rightarrow -TW & Y \rightarrow /FY & \\
 & A \rightarrow TV & F \rightarrow E & & W \rightarrow \varepsilon & Y \rightarrow \varepsilon & \\
 & A \rightarrow TW & F \rightarrow EZ & & & &
 \end{array}$$

Die linksrekursionsfreie Grammatik enthält mehr Regeln als die linksrekursive Form!

Aufgabe 6.6

Analysieren Sie den Satz

$w_1 = \text{begin identifikator} = \text{identifikator} ; \text{end} \#$
und den Satz

$w_2 = \text{begin identifikator} = \text{identifikator} \text{end} \#$

mit dem LL(1)-Analysator nach Bild 6.7 und stellen Sie die erzeugten Syntaxbäume dar.

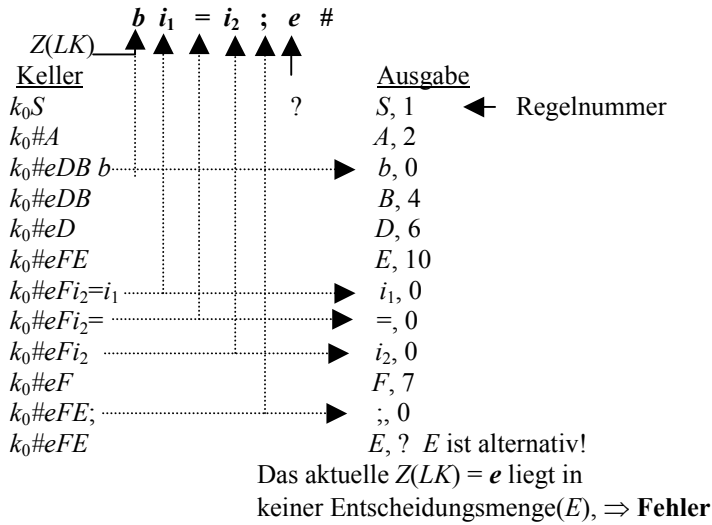
Es wird empfohlen, die Sätze w_1 und w_2 in ihren Kurzschreibweisen $b i = i ; e \#$ bzw. $b i = i e \#$ zu analysieren. Sie können dann auf die im Bild 6.6 angegebenen Entscheidungsmengen zurückgreifen ($k_0 =$ Kellerstartzeichen).

Hinweis: Im Bild 6.6 muss es heißen: $r_8: F \rightarrow \varepsilon$ Entscheidungsmenge $\{e\}$ statt $\{i, b\}$

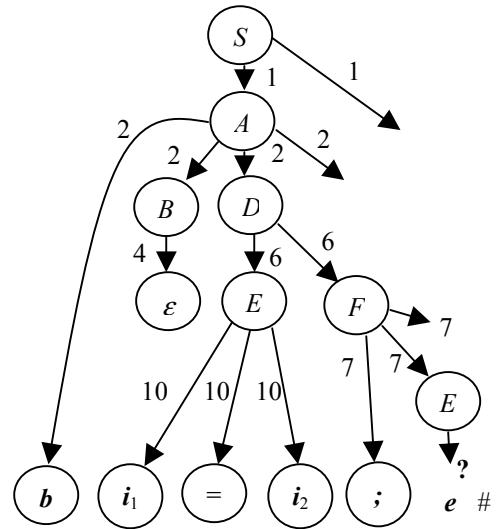
:
 $r_{10}: E \rightarrow i = i$ Entscheidungsmenge $\{i\}$ statt $\{b\}$.

Im Bild 6.7 müssen zweimal $Z(LK)$ statt $LK(Z)$ gesetzt werden. Bei der Ausgabe eines Terminals wird die Regelnummer 0 angefügt!

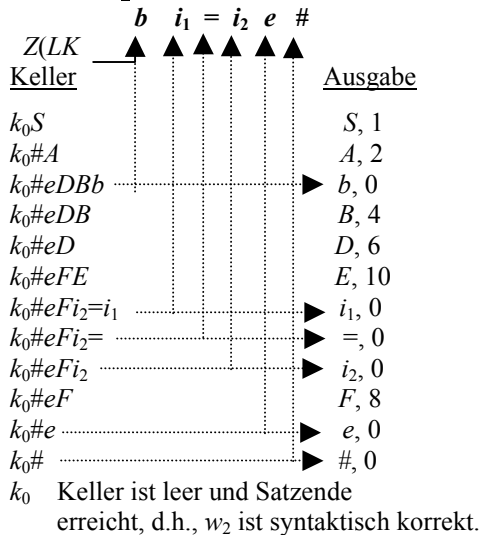
Analyse w_1 :



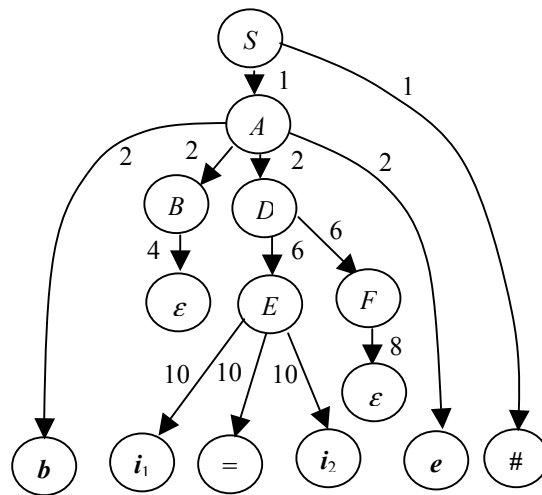
Syntaxbaum:



Analyse w_2 :



Syntaxbaum:



Aufgabe 6.7

Vervollständigen Sie die Regelprozeduren des im Abschnitt 6.4.2 auszugsweise dargestellten Programms *rek_abstieg* und analysieren Sie die gleichen Sätze wie in der Aufgabe 6.6 angegeben. Vergleichen Sie die Syntaxbäume!

Beachten Sie bitte den zur Lösung der Aufgabe 6.6 gegebenen Hinweis zur Korrektur des Bildes 6.6:

- $r_8: F \rightarrow \epsilon$ mit der Entscheidungsmenge $\{e\}$ statt $\{i, b\}$ und
- $r_{10}: E \rightarrow i=i$ mit der Entscheidungsmenge $\{i\}$ statt $\{b\}$.

Programm *rek_abstieg* (hier vollständig angegeben)

/*In jeder Prozedur sind die Entscheidungsmengen E_r der Regeln $r \in R$ enthalten*/

/*Die Prozedur *behandle* gibt immer das Paar $(Z(LK), 0)$ aus und inkrementiert LK */

program *rek_abstieg*

:

```

procedure S;           /*Regel 1*/
begin ausgabe(S, 1);
  A;
  if Z(LK) = '#' then handle(Z(LK)) else fehler;
end;

procedure A;           /*Regel 2*/
begin ausgabe(A, 2);
  if Z(LK) = 'b' then handle (Z(LK)) else fehler;
  B; D;
  if Z(LK) = 'e' then handle (Z(LK)) else fehler;
end;

procedure B;           /*Regeln 3 und 4, alternativ!*/
begin if Z(LK) = 'n' then begin ausgabe(B, 3);
  C;
  if Z(LK) = ';' then handle (Z(LK)) else fehler;
  B;
  end;
  else
  if ((Z(LK) = 'i') or (Z(LK) = 'b'))
  then ausgabe(B, 4) else fehler;

end;

procedure C;
begin ausgabe(C, 5);
  if Z(LK) = 'n' then handle (Z(LK)) else fehler;
  if Z(LK) = 'i' then handle (Z(LK)) else fehler;
end;

procedure D;
begin ausgabe(D, 6);
  E; F;
end;

procedure F;           /*Regeln 7 und 8, alternativ!*/
begin if Z(LK) = ';' then begin ausgabe(F, 7);
  handle(Z(LK);
  E; F;
  end;
  if Z(LK) = 'e' then ausgabe(F, 8) else fehler;
end;

procedure E;           /*Regeln 9 und 10, alternativ!*/
begin if Z(LK) = 'b' then begin ausgabe(E, 9);
  A;
  end;
  else
  if Z(LK) = 'i' then begin
  ausgabe(E, 10);
  handle(Z(LK);
  handle(Z(LK);
  handle(Z(LK);
  end;
  else fehler;

end;
end.

```

Analyse w_1 : $b i_1 = i_2 ; e \#$

Aufruffolge	Ausgabe
call S:	S, 1
call A:	A, 2
behandle 'b'	b, 0
call B:	B, 4
call D:	D, 6
call E:	E, 10
behandle 'i ₁ '	i ₁ , 0
behandle '='	=, 0
behandle 'i ₂ '	i ₂ , 0
call F:	F, 7
behandle ','	;, 0
call E	Fehler!

Analyse w_2 : $b i_1 = i_2 e \#$

Aufruffolge	Ausgabe
call S:	S, 1
call A:	A, 2
behandle 'b'	b, 0
call B:	B, 4
call D:	D, 6
call E:	E, 10
behandle 'i ₁ '	i ₁ , 0
behandle '='	=, 0
behandle 'i ₂ '	i ₂ , 0
call F:	F, 8
behandle 'e'	e, 0
behandle '#'	#, 0

Satzende erreicht und ausgegeben, d.h., w_2 ist syntaktisch korrekt.

Die Ausgaben sind in beiden Fällen identisch mit den in der Aufgabe 6.6 gewonnenen Darstellungen, weshalb auch die Syntaxbäume übereinstimmen.

Aufgabe 6.8

Berechnen Sie für die im Beispiel 6.6 angegebene nicht linksrekursionsfreie Grammatik die Menge $FOLLOW(N)$ und bestimmen Sie für die gleiche Grammatik alle existierenden $LR(0)$ – Informationen.

Die Grammatik wird um eine Startregel, die das Satzdekennzeichen '#' enthält, erweitert.:

$N = \{S', S, A, B\}$, $T = \{a, b, c, d, e, \#\}$, $\sigma = S'$, $R = \{(S', S\#), (S, aABe), (A, Abc), (A, b), (B, d)\}$.

Berechnung $FOLLOW(N)$ – (s. a. Lösung zur Aufgabe 6.4)

Unter Verwendung der Gln. (6.1) bis (6.7) folgt:

$$R_{anf} = \{(S', S), (S, a), (A, A), (A, b), (B, d)\}$$

$$R^+_{anf} = \{(S', S), (S', a), (S, a), (A, A), (A, b), (B, d)\}$$

$$FIRST(G) = \{(S', a), (S, a), (A, b), (B, d), (a, a), (b, b), (c, c), (d, d), (e, e), (\#, \#)\}$$

$$R_{nach} = \emptyset$$

$$R^*_{nach} = \{(S', S'), (S, S), (A, A), (B, B)\}$$

$$R_{nachT} = \{(S, \#), (A, d), (B, e), (A, b)\}$$

$$FOLLOW(N) = R^*_{nach} \circ R_{nachT} = \{(S, \#), (A, d), (A, b), (B, e)\}$$

N	FOLLOW(N)
S'	\emptyset
S	{#}
A	{b, d}
B	{e}

Die neue Startregel $S' \rightarrow S\#$ ist für die Benutzung der **LR(0) – Informationen** erforderlich, um einem Satzanalysator das Analyseende kenntlich machen zu können.

Für die Ableitung der $LR(0)$ – Informationen selbst braucht das # - Symbol aber nicht benutzt zu werden, weil es bezüglich der Grammatikregeln keine Semantik besitzt.

Die Items I_i werden über Hüllenoperationen $H[\dots]$ gefunden. Sie stellen Zustände im Reduktionsgraphen (s. Bild 6.9) dar. Insofern entsprechen die I_i formal ganz gewöhnlichen Automatenzuständen und es kann somit $I_i \equiv q_i$ angenommen werden.

Alle G -Operatoren (G von GOTO) beziehen sich auf die Zustandübergänge $G[I_i, V]$ mit $V \in N \cup T$. Ein Punkt vor, zwischen oder nach einem Zeichen aus der Menge V stellt die aktuelle Ableitungsposition bei der Hüllenbildung dar.

$$I_0 = H[S' \rightarrow \cdot S] = \begin{pmatrix} S' \rightarrow \cdot S \\ S \rightarrow \cdot aABe \end{pmatrix} \quad \text{Die Hülle von } S' \rightarrow \cdot S \text{ enthält außer dem Argument } [S' \rightarrow \cdot S] \text{ auch alle Produktionen aus } R, \text{ die die in } H[\dots] \text{ hinter dem Punkt stehenden Nichtterminale ableitet.}$$

$$I_1 = G[I_0, S] = H[S' \rightarrow S.] = [S' \rightarrow S.] \quad *$$

$$I_2 = G[I_0, a] = H[S \rightarrow a \cdot ABe] = \begin{pmatrix} S \rightarrow a \cdot ABe \\ A \rightarrow \cdot Abc \\ A \rightarrow \cdot b \end{pmatrix}$$

$$I_3 = G[I_2, A] = H \begin{pmatrix} S \rightarrow aA \cdot Be \\ A \rightarrow A \cdot bc \end{pmatrix} = \begin{pmatrix} S \rightarrow aA \cdot Be \\ A \rightarrow A \cdot bc \\ B \rightarrow \cdot d \end{pmatrix}$$

$$I_4 = G[I_2, b] = H[A \rightarrow b.] = [A \rightarrow b.] \quad *$$

$$I_5 = G[I_3, B] = H[S \rightarrow aAB \cdot e] = [S \rightarrow aAB \cdot e]$$

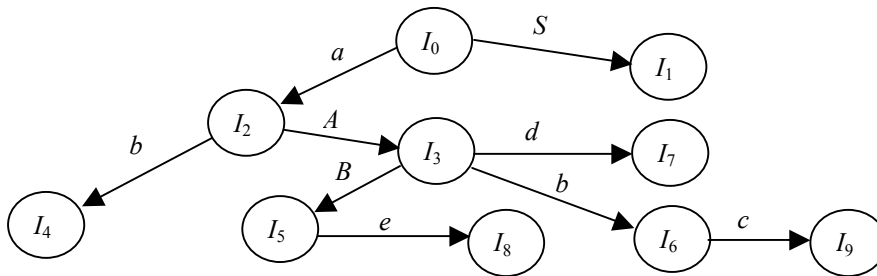
$$I_6 = G[I_3, b] = H[A \rightarrow Ab \cdot c] = [A \rightarrow Ab \cdot c]$$

$$I_7 = G[I_3, d] = H[B \rightarrow d.] = [B \rightarrow d.] \quad *$$

$$I_8 = G[I_5, e] = H[S \rightarrow aABe.] = [S \rightarrow aABe.] \quad *$$

$$I_9 = G[I_6, c] = H[A \rightarrow Abc.] = [A \rightarrow Abc.] \quad *$$

Damit sind alle in G enthaltenen Regeln vollständig durchlaufen worden, erkennbar am Ableitungspunkt hinter dem letzten Zeichen der rechten Regelseiten (siehe *). Der entsprechende Reduktionsgraph analog zum Bild 6.9 ergibt sich hier zu:



Aufgabe 6.9

Bestimmen Sie selbstständig die Tabelle 6.3 (Parsertabelle) anhand der hierfür angegebenen Algorithmen.

Hinweis: In der Tabelle 6.3 muss die rechte Regelseite ($E+Z$) in der #-Spalte eine Zeile höher gesetzt, also dem Zustand q_9 zugeordnet werden.

Aus dem Bild 6.9 können alle Zustandsübergänge (G -Operationen) für alle auftretenden Items $I_k \equiv q_k$ herausgelesen werden:

$$I_1 = G(I_0, A) \quad I_3 = G(I_2, E) \quad I_5 = G(I_3, +) \quad I_7 = G(I_5, a) \quad I_9 = G(I_6,) \quad I_{11} = G(I_{10}, Z)$$

$$I_2 = G(I_0, () \quad I_4 = G(I_2, a) \quad I_6 = G(I_5, Z) \quad I_8 = G(I_5, b) \quad I_{10} = G(I_8, +) \quad I_8 = G(I_{10}, b)$$

Für die Erstellung der Sprungtabelle werden nur die über Nichtterminale gesteuerten Übergänge benötigt.

Sprungtabelle:

$k = 0 \rightarrow I_0:$	für $X = A$ ist $G(I_0, A) = I_1$	Eintrag q_1 für q_0, A
	für $X = E$ ist $G(I_0, E) = \varepsilon$ für $X = Z$ ist $G(I_0, Z) = \varepsilon$	kein Eintrag, weil $G(I_0, E)$ und $G(I_0, Z)$ nicht existieren!
$k = 1 \rightarrow I_1:$	für $X = A, E, Z$	keine Einträge (siehe $k = 0$)
$k = 2 \rightarrow I_2:$	für $X = A$	kein Eintrag
	für $X = E$ ist $G(I_2, E) = I_3$ für $X = Z$	Eintrag q_3 für q_2, E
		kein Eintrag
$k = 3 \rightarrow I_3:$	für $X = A, E, Z$	keine Einträge
$k = 4 \rightarrow I_4:$		
$k = 5 \rightarrow I_5:$	für $X = A$ für $X = E$	kein Eintrag kein Eintrag
	für $X = Z$ ist $G(I_5, Z) = I_6$	Eintrag q_6 für q_5, Z
$k = 6 \rightarrow I_6:$	für $X = A, E, Z$	keine Einträge
\vdots		
$k = 9 \rightarrow I_9:$		
$k = 10 \rightarrow I_{10}:$	für $X = A$ für $X = E$ für $X = Z$ ist $G(I_{10}, Z) = I_{11}$	kein Eintrag kein Eintrag Eintrag q_{11} für q_{10}, Z

Aktionstabelle:

Zunächst ist $FOLLOW(N)$ zu bestimmen:

$$R_{anf} = \{(S, A), (A, (), (E, E), (E, a), (Z, b))\}$$

$$R^+_{anf} = \{(S, A), (S, (), (A, (), (E, E), (E, a), (Z, b))\}$$

$$FIRST(G) = \{(S, (), (A, (), (E, a), (Z, b))\} \cup \{(t, t) \text{ für alle } t \in T\}$$

$$R_{nach} = \{(Z, Z)\}$$

$$R^*_{nach} = \{(S, S), (A, A), (E, E), (Z, Z)\}$$

$$R_{nachT} = \{(A, \#), (E, +), (Z,))\}$$

$$FOLLOW(N) = \{(A, \#), (E, +), (Z,))\}$$

N	$FOLLOW(N)$
S	-
A	{#}
E	{+}
Z	{) }

Ferner sind alle $LR(0)$ – Items aus den Regeln von G bereitzustellen:

$$R = \begin{pmatrix} S \rightarrow A & r_1 \\ A \rightarrow (E + Z) & r_2 \\ E \rightarrow E + a & r_3 \\ E \rightarrow a & r_4 \\ Z \rightarrow b + Z & r_5 \\ Z \rightarrow b & r_6 \end{pmatrix} \quad I_0 = H[S \rightarrow \cdot A] = \overbrace{\begin{pmatrix} S \rightarrow \cdot A \\ A \rightarrow \cdot (E + Z) \end{pmatrix}}^{\text{Item von } I_0}$$

$$I_1 = G(I_0, A) = H[S \rightarrow A \cdot] = [S \rightarrow A \cdot] \quad r_1 \text{ ist abgeleitet}$$

$$\begin{aligned}
I_2 = G(I_0, () = H[A \rightarrow (. E + Z)] &= \begin{pmatrix} A \rightarrow (. E + Z) \\ E \rightarrow . E + a \\ E \rightarrow . a \end{pmatrix} \\
I_3 = G(I_2, E) = H \begin{pmatrix} A \rightarrow (E. + Z) \\ E \rightarrow E. + a \end{pmatrix} &= \begin{pmatrix} A \rightarrow (E. + Z) \\ E \rightarrow E. + a \end{pmatrix} \\
I_4 = G(I_2, a) = H [E \rightarrow a.] &= [E \rightarrow a.] \quad r_4 \text{ ist abgeleitet} \\
I_5 = G(I_3, +) = H \begin{pmatrix} A \rightarrow (E+. Z) \\ E \rightarrow E+. a \end{pmatrix} &= \begin{pmatrix} A \rightarrow (E+. Z) \\ E \rightarrow E+. a \\ Z \rightarrow . b + Z \\ Z \rightarrow . b \end{pmatrix} \\
I_6 = G(I_5, Z) = H [A \rightarrow (E + Z.)] &= [A \rightarrow (E + Z.)] \\
I_7 = G(I_5, a) = H [E \rightarrow E + a.] &= [E \rightarrow E + a.] \quad r_3 \text{ ist abgeleitet} \\
I_8 = G(I_5, b) = H \begin{pmatrix} Z \rightarrow b. + Z \\ Z \rightarrow b. \end{pmatrix} &= \begin{pmatrix} Z \rightarrow b. + Z \\ Z \rightarrow b. \end{pmatrix} \quad r_6 \text{ ist abgeleitet} \\
I_9 = G(I_6,) = H [A \rightarrow (E + Z).] &= [A \rightarrow (E + Z).] \quad r_2 \text{ ist abgeleitet} \\
I_{10} = G(I_8, +) = H [Z \rightarrow b + . Z] &= \begin{pmatrix} Z \rightarrow b + . Z \\ Z \rightarrow . b + Z \\ Z \rightarrow . b \end{pmatrix} \\
I_{11} = G(I_{10}, Z) = H [Z \rightarrow b + Z.] &= [Z \rightarrow b + Z.] \quad r_5 \text{ ist abgeleitet} \\
I_{12} = G(I_{10}, b) = H \begin{pmatrix} Z \rightarrow b. + Z \\ Z \rightarrow b. \end{pmatrix} &= \begin{pmatrix} Z \rightarrow b. + Z \\ Z \rightarrow b. \end{pmatrix} = I_8
\end{aligned}$$

Erzeugung der Aktionstabelle (Algorithmus S. 187):

Position $[q, z]$ bedeutet: In der Zeile gemäß q und der Spalte gemäß dem Zeichen z .

- $k=0$ I_0 enthält ein Item $[A \rightarrow \alpha.(\beta)]$ und es ist $G[I_0, () = I_2 \rightarrow q_2$ auf die Position $[q_0, ()]$
- $k=1$ I_1 enthält ein Item $[S \rightarrow A.]$ \rightarrow ENDE auf Position $[q_1, -]$
(s. FOLLOW(S))
- $k=2$ I_2 enthält ein Item $[E \rightarrow \alpha.a\beta]$ und es ist $G[I_2, a] = I_4 \rightarrow q_4$ auf die Position $[q_2, a]$
- $k=3$ I_3 enthält ein Item $[A \rightarrow \alpha.+ \beta]$ und es ist $G[I_3, +] = I_5 \rightarrow q_5$ auf die Position $[q_3, +]$
- $k=4$ I_4 enthält ein Item $[E \rightarrow a.]$ \rightarrow a auf Position $[q_4, +]$
(s. FOLLOW(E))
- $k=5$ I_5 enthält ein Item $[E \rightarrow \alpha.a]$ und es ist $G[I_5, a] = I_7 \rightarrow q_7$ auf die Position $[q_5, a]$
 I_5 enthält ein Item $[Z \rightarrow .b]$ und es ist $G[I_5, b] = I_8 \rightarrow q_8$ auf die Position $[q_5, b]$
- $k=6$ I_6 enthält ein Item $[A \rightarrow \alpha.)]$ und es ist $G[I_6,)] = I_9 \rightarrow q_9$ auf die Position $[q_6, b]$
- $k=7$ I_7 enthält ein Item $[E \rightarrow E + a.]$ \rightarrow $E+a$ auf Position $[q_7, +]$
(s. FOLLOW(E))
- $k=8$ I_8 enthält ein Item $[Z \rightarrow \alpha.+ \beta]$ und es ist $G[I_8, +] = I_{10} \rightarrow q_{10}$ auf Position $[q_8, +]$
 I_8 enthält ein Item $[Z \rightarrow b.]$ \rightarrow b auf die Position $[q_8,)]$
(s. FOLLOW(Z))

$k=9$ I_9 enthält ein Item $[A \rightarrow (E+Z).]$

$\rightarrow (E+Z)$ auf Position $[q_9, \#]$
(s. FOLLOW(A))

$k=10$ I_{10} enthält ein Item $[Z \rightarrow \underline{b}\beta]$ und es ist $G[I_{10}, b] = I_{10} \rightarrow q_8$ auf Position $[q_{10}, b]$

$k=11$ I_{11} enthält ein Item $[Z \rightarrow b+Z.]$ $\rightarrow b+Z$ auf Position $[q_{11}, .]$
(s. FOLLOW(Z))

ABSCHNITT 6.5.3 / Übungen

Aufgabe 6.10

Überführen Sie die arithmetischen Ausdrücke

$$\begin{aligned} w_1 &= a + b * c, \\ w_2 &= a * b + c, \\ w_3 &= 3 * 4 - 5 + 1 * 2, \\ w_4 &= a - b * c - d / (b * e) \end{aligned}$$

mit dem im Bild 6.13 dargestellten Kellerautomaten in eine semantisch äquivalente UPN-Notation.

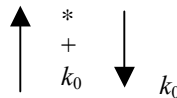
ist ein Satzendezeichen.

$w_1 = a + b * c \#$

Z(LK): $\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow$

UPN-Ausgabe: $a b c * +$

Kellersituationen: 1. 2.

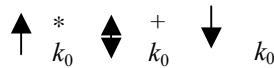


$w_2 = a * b + c \#$

Z(LK): $\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow$

UPN-Ausgabe: $a b * c +$

Kellersituationen: 1. 2. 3.

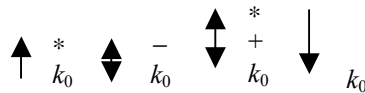


$w_3 = 3 * 4 - 5 + 1 * 2 \#$

Z(LK): $\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow$

UPN-Ausgabe: $3 4 * 5 - 1 2 * +$

Kellersituat.: 1. 2. 3. 4.

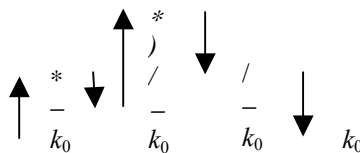


$w_4 = a - b * c - d / (b * e)$

Z(LK): $\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow$

UPN-Ausgabe: $a b c * - d b e * / -$

Kellersituat.: 1. 2. 3. 4.



Aufgabe 6.11

Erzeugen Sie für den gegebenen arithmetischen Ausdruck eine *UPN*-Struktur, einen Binärbaum und die Zielsprache im Assemblercode (s. a. Bild 6.14): $w = ((3+7-5) / (8+1)) - ((6*(4-9) / (2*7/6)))$.

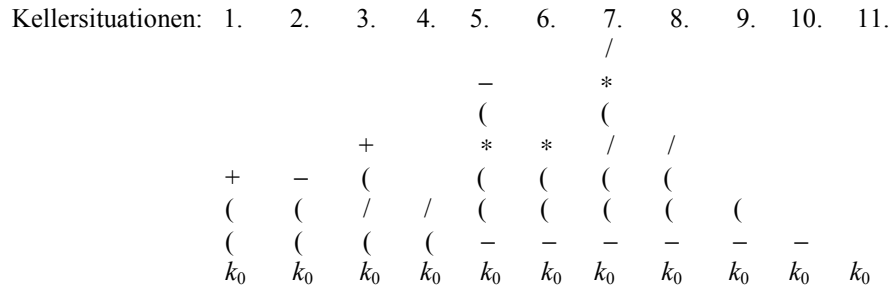
Hinweis:

In dieser Aufgabenstellung bitte korrigieren: ‘Bild 6.14’ statt ‘Bild 6.15’ und am Ende von w ‘)))’ statt ‘))’.

UPN – Struktur:

$$w = ((3 + 7 - 5) / (8 + 1)) - ((6 * (4 - 9) / (2 * 7 / 6))) \#$$

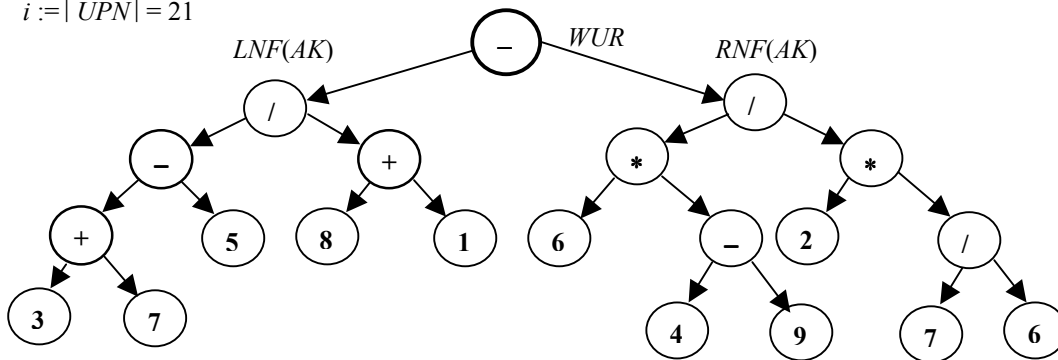
UPN – Ausgabe: 3 7 + 5 - 8 1 + / 6 4 9 - * 2 7 6 / * / -



Binärbaum:

Hinweis: Benutzen Sie den auf der Seite 198 angegebenen Algorithmus. Korrigieren Sie die 2. und 3. Zeile der Berechnungsvorschrift in: *begin while* {*RNF*(*AK*)≠0 & *LNF*(*AK*)≠0 & *AK* ≠ *WUR*} *do* *AK* := *Vorgänger*(*AK*);

$i := |UPN| = 21$



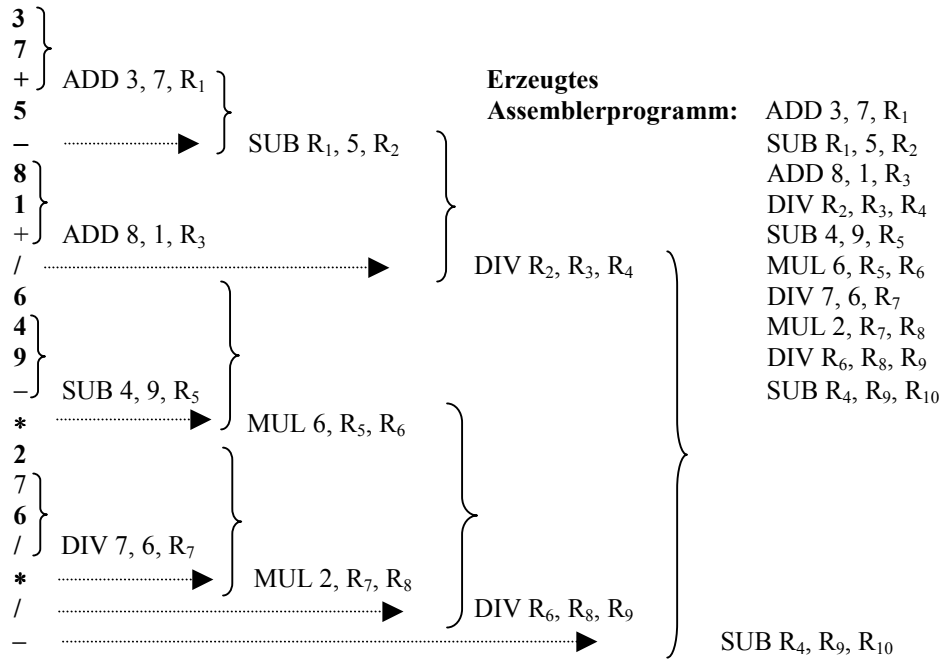
Zielsprache im Assemblercode:

Die Register R_i werden laufend von 1 beginnend “vergeben“ und der dritte Befehlsparameter nimmt immer das Operationsergebnis auf. Eine Optimierung der jeweils erforderlichen Registeranzahl ist möglich!

Die zuvor aufgebaute *UPN*-Struktur wird als Keller generiert und vom Kellerboden beginnend abgebaut. Jeder angetroffene Operator erzeugt einen Assemblerbefehl, der die zwei unmittelbar darüberliegenden Operanden bzw. Registerinhalte verknüpft.

UPN-Ausdruck im Keller

Keller-
boden →



E N D E