Theoretische Informatik
Testate

Silas Alexander Kraume
sikra111

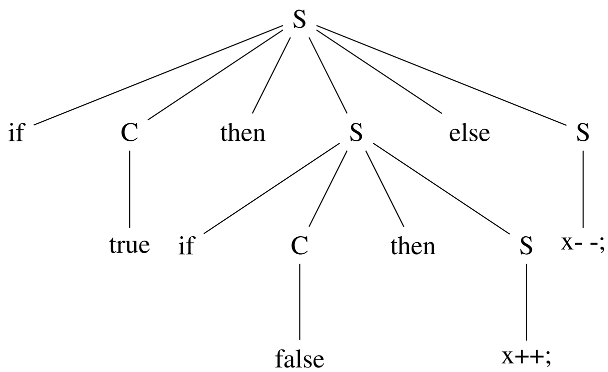
Contents

Testat 1

1.1 Aufgabe 1

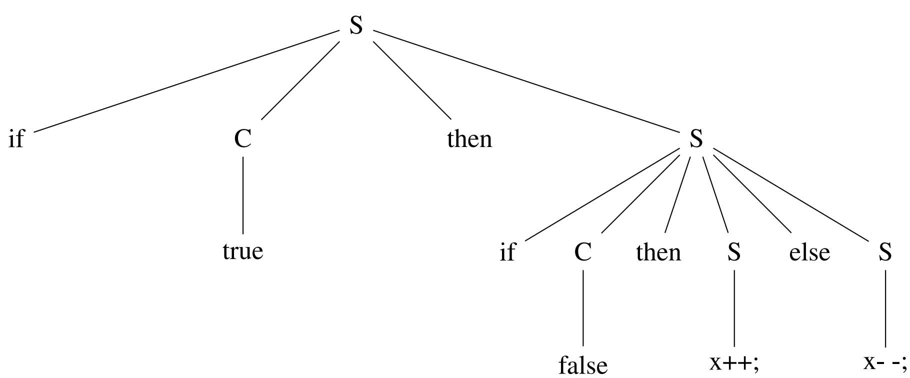
Algorithm 1:

if *C* **then**
    if *C* **then**
        S
else
    S
end



Algorithm 2:

if *C* **then**
    if *C* **then**
        S
    else
        S
    end

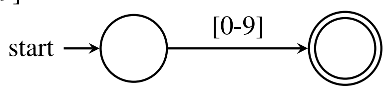


Wort: $\boxed{\text{if true then if false then x++; else x- -;}}$

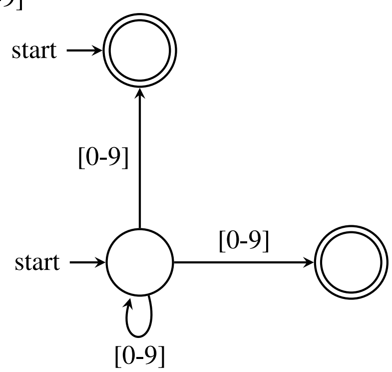
1.2 Aufgabe 2

a)

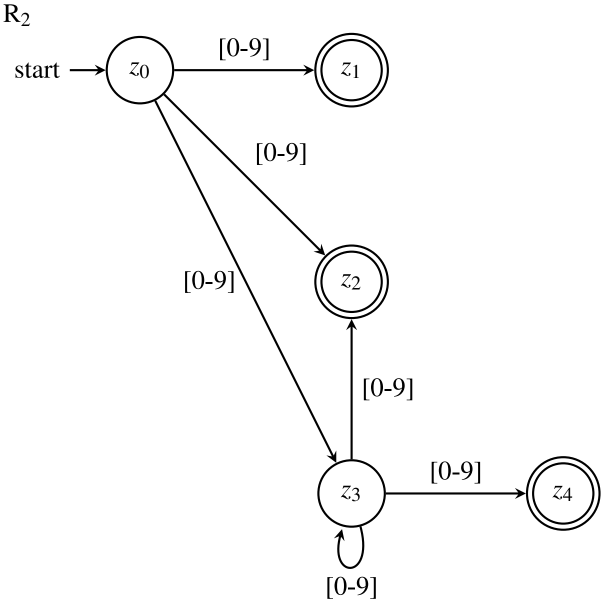
R_1: [0-9]



R_2: $[0\text{-}9]^*$

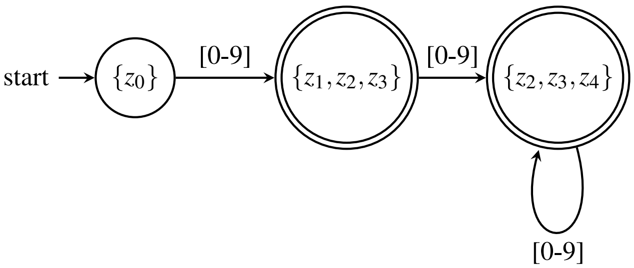


R_3: R_1R_2

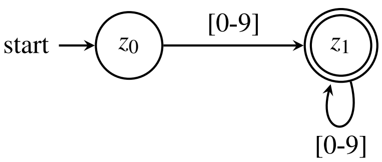


$M = (\Sigma, Z, \delta, S, E)$
$Z = \{z_0, z_1, z_2, z_3, z_4\}$, $S = \{z_0\}$, $E = \{z_1, z_2, z_4\}$

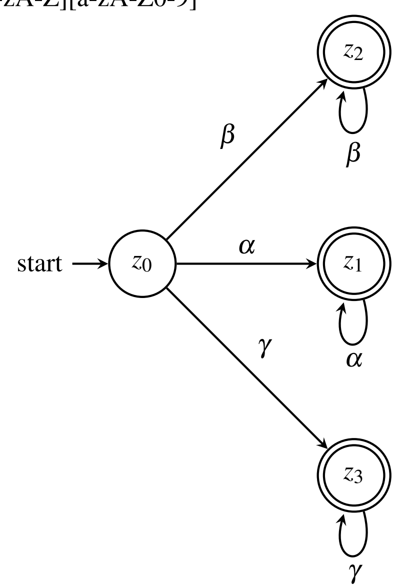
b)



$N = (\Sigma, Z, \delta, S, F)$
$Z = \{z_0, \{z_1, z_2, z_3\}, \{z_2, z_3, z_4\}, z_3\}$, $S = z_0$, $E = \{\{z_1, z_2, z_3\}, \{z_2, z_3, z_4\}, z_3\}$
c)



d)
α = [0-9][0-9]*
β = ws ws*
γ = [a-zA-Z][a-zA-Z0-9]*



1.3 Aufgabe 3

a) α steht beliebig oft, aber mindestens einmal, hintereinander:

$$\alpha^{+} \equiv \alpha(\alpha)^{*}$$
$$L(\alpha(\alpha)^{*}) = L(\alpha)L(\alpha^{*}) = L(\alpha)L(\alpha)^{*} = \{\alpha x \mid x \in \alpha^{*}\}$$

b) α tritt einmal oder gar nicht auf:

$$\alpha? \equiv (\lambda + \alpha)$$
$$L(\lambda + \alpha) = L(\lambda) \cup L(\alpha) = \{\lambda\} \cup L(\alpha)$$

c) für eine natürliche Zahl n wird α genau n-mal wiederholt:

$$\alpha^{n} \equiv \underbrace{\alpha\alpha\cdots\alpha}_{n \text{ mal}}$$
$$L\left(\underbrace{\alpha\alpha\cdots\alpha}_{n \text{ mal}}\right) = \underbrace{L(\alpha)\,L(\alpha)\cdots L(\alpha)}_{n \text{ mal}}$$

Testat 2

2.1 Aufgabe 1

Angenommen G sei regulär. Dann existiert nach dem Pumping-Lemma eine Zahl n $\geqslant$ 1, so dass für alle Wörter x $\in$ L(G) mit $|x| \geqslant$ n eine Zerlegung x = uvw mit

1. $|uv| \leqslant$ n
2. $|v| \geqslant 1$
3. $\forall i \geqslant 0$. uv^{i}w $\in$ L(G)

Wähle x = $\underbrace{((\cdots((}_{n \text{ mal}}\ \mathrm{a}\ \underbrace{+\,\mathrm{a}) + \mathrm{a})\cdots\ + \mathrm{a})}_{n \text{ mal}}$ $\in$ L(G) mit $|x|$ = 2n + 1 $\geqslant$ n.
Sei x = uvw = $(^{\mathrm{n}}\mathrm{a}\big(+\mathrm{a})\big)^{\mathrm{n}}$, also uv = $(^{\mathrm{n}}$.
Aus uv = $(^{\mathrm{n}}$ folgt ebenfalls v = $(^{\mathrm{i}}$ für ein i $\leqslant$ n und entsprechend u = $(^{\mathrm{j}}$ für ein j = n-i.
Für jedes k$\in \mathbb{N} >$ n gilt: uv$^{\mathrm{k}}$w $\notin$ L(G).
Insbesondere gilt für k = 0: uv$^{\mathrm{k}}$w = uv^{0}w = uw $\notin$ L(G).
Es folgt: G ist nicht regulär!

$\blacksquare$

2.2 Aufgabe 2

x = uv^{3}w = abcdbcdbcdbe $\in$ *L(M)*, $|x| \geqslant p$
1. $|uv| \leqslant p$
2. $|v| \geqslant 1$
3. $\forall i \geqslant 0$. uv^{i}w $\in$ *L(M)*
$\Rightarrow$ u = ab, v = cdb, w = e
$\Rightarrow$ uv^{i}w = ab(cdb)^{i}e $\in$ *L(M)*

2.3 Aufgabe 3

a) ist regulär, ein einfacher DFA für den entsprechenden Ausdruck ist trivial.
b) + c) nicht regulär, denn HTML und Java sind endlos erweiterbar, erlauben willkürlich verschachtelte/rekursive Strukturen.

2.4 Aufgabe 4

a)
1. Entferne alle von z_0 aus nicht erreichbaren Zustände aus Z.
Alle Zustände sind von z_0 aus erreichbar.

2. Erstelle eine Tabelle aller (ungeordneten) Zustandspaare $\{z, z'\}$ von M mit $z \neq z'$.
3. Markiere alle Paare $\{z, z'\}$ mit $z \in F \Leftrightarrow z' \notin F$.

	z_0	z_1	z_2	z_3
z_4		X	X	X
z_3	X			$-$
z_2	X		$-$	$-$
z_1	X	$-$	$-$	$-$

4. Sei $\{z, z'\}$ ein unmarkiertes Paar. Prüfe für jedes a $\in \Sigma$, ob $\{\sigma(z,a), \sigma(z',a)\}$ bereits markiert ist. Ist mindestens ein Test erfolgreich, so markiere auch $\{z, z'\}$.
5. Wiederhole Schritt 4, bis keine Änderung mehr eintritt.

$$\{\sigma(z_1,0), \sigma(z_3,0)\} = \{z_2, z_4\}$$
$$\{\sigma(z_2,0), \sigma(z_3,0)\} = \{z_2, z_4\}$$
$$\{\sigma(z_1,1), \sigma(z_2,1)\} = \{z_0, z_2\}$$

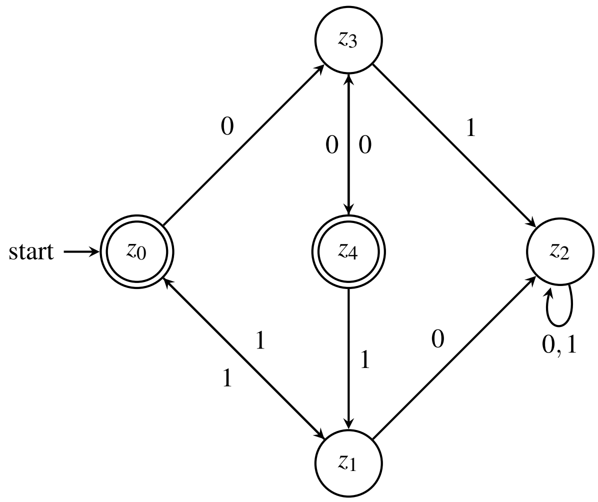
	z_0	z_1	z_2	z_3
z_4		X	X	X
z_3	X	X	X	$-$
z_2	X	X	$-$	$-$
z_1	X	$-$	$-$	$-$

6. Bilde maximale Mengen paarweise nicht disjunkter unmarkierter Zustandspaare und verschmelze jeweils alle Zustände einer Menge zu einem neuen Zustand.
Verschmelze z_0 und z_4 zu z_{04}.

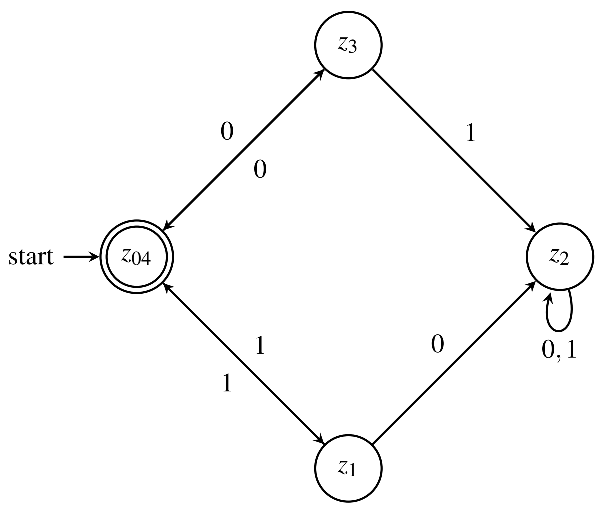
M' = (Σ, Z', σ', z_{04}, F')
$\Sigma = \{0, 1\}$
Z' = $\{z_{04}, z_1, z_2, z_3\}$
F' = $\{z_{04}\}$

σ'	z_{04}	z_1	z_2	z_3
0	z_3	z_2	z_2	z_{04}
1	z_1	z_{04}	z_2	z_2

M:



M’:



b)
Äquivalenzklassen:

$[\lambda] = \{\lambda, 00, 11, \dots\}$
[0]
[1]
$[01] = [10]$

c)
Man kann folgende Zustände zusammen legen:

z_1 und z_6
z_2 und z_4 und z_7

Testat 3

3.1 Aufgabe 1

a)
$N_\lambda = \{C\}$, da $C \to \lambda \in P_1$
$N_\lambda = \{C, D\}$, da $D \to CCC \to \lambda \in P_1$

$$\begin{aligned} P'_1 = \{&S \to AD \mid DA \mid A, \\ &A \to BC \mid B, \\ &B \to S1 \mid 0, \\ &C \to 1, \\ &D \to AB \mid CCC \mid 0C \mid CC \mid C \mid 0\} \end{aligned}$$

b)

Zyklus: $B \to C \to D \to B \ldots$
$N'_2 = \{S, A, X\}$

$$\begin{aligned} P'_2 = \{&S \to AXXS \mid a, \\ &A \to a \mid aX, \\ &X \to bA, \\ &X \to AX \mid c, \\ &X \to d\} \\ = \{&S \to AXXS \mid a, \\ &A \to a \mid aX, \\ &X \to bA \mid AX \mid c \mid d\} \end{aligned}$$

c)

Zyklus: /
S(1), A(2), B(3), C(4), D(5)

$$\begin{aligned} P'_3 = \{&S \to ABCS \mid a, \\ &A \to a \mid aB, \\ &B \to Ad \mid d \mid c \mid bA, \\ &C \to Ad \mid d \mid c, \\ &D \to d\}. \end{aligned}$$

d)

λ-frei ✓
keine einfachen Regel ✓
A $\to$ a und B $\to$ b werden übernommen.
X_a, X_b einführen:

$$\begin{aligned} P'_4 = \{ & S \to ABAS \mid X_aX_b, \\ & A \to X_aA \mid a, \\ & B \to b \mid X_aX_bX_b, \\ & X_a \to a, \\ & X_b \to b\}. \end{aligned}$$

Nichtterminalketten ersetzen:

$$\begin{aligned} P''_4 = \{ & S \to AC_0 \mid X_aX_b, \\ & A \to X_aA \mid a, \\ & B \to b \mid X_aC_2, \\ & X_a \to a, \\ & X_b \to b, \\ & C_0 \to BC_1, \\ & C_1 \to AS, \\ & C_2 \to X_bX_b\}. \end{aligned}$$

$N'_4 = \{S, A, B, X_a, X_b, C_0, C_1, C_2\}$

3.2 Aufgabe 2

a)

i	1	2	3	4	5	6
5	S, S_2					
4		C_2				
3	S		T_2			
2		C_2		S_2		
1			T_2		E_2	
0	I	C	T	S	E	S
j	*if*	*true*	*then*	x++;	*else*	x--;

b)

i	1	2	3	4	5
4					
3	S				
2		C_2			
1			T_2		
0	I	C	T	S	E
j	*if*	*true*	*then*	x++;	*else*

3.3 Aufgabe 3

a)

L = $\{(ab)^{m}c^{2m} \mid 1 \leq m \in \mathbb{N}\}$

b)

- DPDAs sind in (N)PDAs überführbar.
- Die Sprache, die ein DPDA akzeptiert ist eine Teilmenge der Sprache, die ein PDA akzeptiert.
- DPDA hat zusätzlich Endzustände definiert.
- DPDA akzeptiert eine Eingabe per Endzustand, nicht leerem Stack.
- DPDAs sind deterministisch, haben also eindeutig bestimmte Übergänge.

3.4 Aufgabe 4

M = ({a, b}, {a, b, S, A, B}, {z}, δ, z, S)
δ:

zλS $\rightarrow$ zABAS
zλS $\rightarrow$ zab
zλA $\rightarrow$ zaA
zλA $\rightarrow$ za
zλB $\rightarrow$ zb
zλB $\rightarrow$ zabb

zaa $\rightarrow$ z λ
zbb $\rightarrow$ z λ

Testat 4

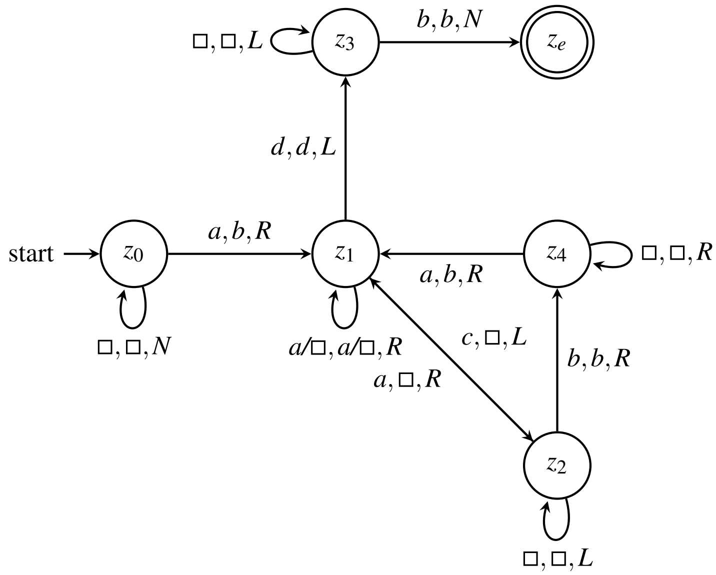
4.1 Aufgabe 1

a)

$$
\begin{array}{rl}
z_0 <div><div></div><div></div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>z_1 <div></div><div></div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}><div>z_1 </div><div></div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>z_2 <div>\square<div></div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square z_1\square<div></div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square z_1 <div></div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square<div>z_1 </div></div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square z_2 <div>\square</div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square z_1\square</div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square\square z_1 </div><div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square z_2\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square z_2\square\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square z_2\square\square\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>z_2\square\square\square\square\square<div><\widehat{/div}> & \vdash_M \\
z_2<\widehat{div}>\square\square\square\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>z_4\square\square\square\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square z_4\square\square\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square z_4\square\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square z_4\square\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square\square z_4\square<div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square\square\square z_4 <div><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square\square\square<\widehat{div}>z_1<\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square\square\square z_3<\widehat{div}><\widehat{/div}> & \vdash_M \\
<\widehat{div}>\square\square\square\square\square z_e<\widehat{div}><\widehat{/div}> &
\end{array}
$$

b)

$a \widehat{=} < div >$
$b \widehat{=} < \widehat{div} >$
$c \widehat{=} < /div >$
$d \widehat{=} < \widehat{/div} >$



L(M) akzeptiert Eingaben, bei denen jedes öffnende div ein matchendes schließendes div besitzt. Hierbei hat das letzte (schließende) div bereits einen Hut, als Vorraussetung für einen LBA, aus diesem Grund wird auch das erste (öffnende) div mit einem Hut markiert. Die TM akzeptiert zudem die leere Eingabe.

c)

LBA:
hat spezifisches Start- und Endsymbol (bzw. verdoppeltes Eingabealphabet).
verlässt den Bereich des Eingabewortes nicht.
verändert die Länge der auf dem Band stehenden Eingabe nicht.

d)

Ja, M ist ein LBA.

4.2 Aufgabe 2

f(x) = x mod 3

M = $\left(\Sigma, \Gamma, Z, \delta, z_0, \square, F\right)$

Σ = $\{0,1,2,3,4,5,6,7,8,9\}$

Γ = $\{0,1,2,3,4,5,6,7,8,9,\square\}$

Z = $\{z_0, z_1, z_2, z_e\}$

F = $\{z_e\}$

δ	z_0	z_1	z_2
0	$(z_0, \square, R)$	$(z_1, \square, R)$	$(z_2, \square, R)$
1	$(z_1, \square, R)$	$(z_2, \square, R)$	$(z_0, \square, R)$
2	$(z_2, \square, R)$	$(z_0, \square, R)$	$(z_1, \square, R)$
3	$(z_0, \square, R)$	$(z_1, \square, R)$	$(z_2, \square, R)$
4	$(z_1, \square, R)$	$(z_2, \square, R)$	$(z_0, \square, R)$
5	$(z_2, \square, R)$	$(z_0, \square, R)$	$(z_1, \square, R)$
6	$(z_0, \square, R)$	$(z_1, \square, R)$	$(z_2, \square, R)$
7	$(z_1, \square, R)$	$(z_2, \square, R)$	$(z_0, \square, R)$
8	$(z_2, \square, R)$	$(z_0, \square, R)$	$(z_1, \square, R)$
9	$(z_0, \square, R)$	$(z_1, \square, R)$	$(z_2, \square, R)$
$\square$	$(z_e, 0, N)$	$(z_e, 1, N)$	$(z_e, 2, N)$

4.3 Aufgabe 3

P1:

Kein LOOP, wegen Keyword WHILE.
Ist WHILE.
Kein GOTO, wegen Keyword DO.

P2:

Kein LOOP, weil x_2 in Loop + ";" vor END.
Kein WHILE, weil x_2 in Loop + ";" vor END.
Kein GOTO, wegen Keyword DO.

P3:

Kein LOOP, weil $x_0 := x_1$; (anstatt $x_0 := x_1$+0;).
Kein WHILE, weil $x_0 := x_1$; (anstatt $x_0 := x_1$+0;).
Kein GOTO, wegen Keyword DO.

P4:

Kein LOOP, weil x_2 in Loop.
Kein WHILE, weil x_2 in Loop.
Kein GOTO, wegen Keyword DO.

P5:

Kein LOOP, weil x_1 in Loop.
Kein WHILE, weil x_1 in Loop.
Kein GOTO, wegen Keyword DO.

P6:

Kein LOOP, wegen Keyword WHILE.
Kein WHILE, weil ";" vor END.
Kein GOTO, wegen Keyword DO.

P7:
Ist LOOP.
Ist WHILE, weil es LOOP ist.
Kein GOTO, wegen Keyword DO.

P8:
Kein LOOP, wegen Keyword WHILE.
Kein WHILE, wegen Keyword GOTO.
Kein GOTO, wegen Keyword DO.

P9:
Kein LOOP, wegen Keyword GOTO.
Kein WHILE, wegen Keyword GOTO.
Kein GOTO, wegen fehlendem Marker vor erster Anweisung (Skript is unklar, ob jede Anweisung einen Marker braucht).

Testat 5

5.1 Aufgabe 1

a)

$$\begin{aligned}
mult\left(3,2\right) &= \\
f'\left(2, mult\left(2,2\right),2\right) &= \\
f'\left(2, f'\left(1, mult\left(1,2\right),2\right),2\right) &= \\
f'\left(2, f'\left(1, f'\left(0, mult\left(0,2\right),2\right),2\right),2\right) &= \\
f'\left(2, f'\left(1, f'\left(0,0,2\right),2\right),2\right) &= \\
f'\left(2, f'\left(1, f'\left(0,0,2\right),2\right),2\right) &= \\
f'\left(2, f'\left(1, add\left(0,2\right),2\right),2\right) &= \\
f'\left(2, f'\left(1,2,2\right),2\right) &= \\
f'\left(2, add\left(2,2\right),2\right) &= \\
f'\left(2, g'\left(1, add\left(1,2\right),2\right),2\right) &= \\
f'\left(2, g'\left(1, g'\left(0, add\left(0,2\right),2\right),2\right),2\right) &= \\
f'\left(2, g'\left(1, g'\left(0,2,2\right),2\right),2\right) &= \\
f'\left(2, g'\left(1,3,2\right),2\right) &= \\
f'\left(2,4,2\right) &= \\
add\left(4,2\right) &= \\
g'\left(3, add\left(3,2\right),2\right) &= \\
g'\left(3, g'\left(2, add\left(2,2\right),2\right),2\right) &= \\
g'\left(3, g'\left(2, g'\left(1, add\left(1,2\right),2\right),2\right),2\right) &= \\
g'\left(3, g'\left(2, g'\left(1, g'\left(0, add\left(0,2\right),2\right),2\right),2\right),2\right) &= \\
g'\left(3, g'\left(2, g'\left(1, g'\left(0,2,2\right),2\right),2\right),2\right) &= \\
g'\left(3, g'\left(2, g'\left(1,3,2\right),2\right),2\right) &= \\
g'\left(3, g'\left(2,4,2\right),2\right) &= \\
g'\left(3,5,2\right) &= \\
6 &=
\end{aligned}$$

f'(a,b,c) = add(id_2^3(a,b,c),id_3^3(a,b,c)) = add(b,c)
g'(a,b,c) = s(id_2^3(a,b,c)) = s(b)

b)

ite(0,T,E) = E = id_2^2(T,E)
ite(n+1,T,E) = T = id_3^4(n,ite(n,T,E),T,E)

5.2 Aufgabe 2

a)

f :

$x\backslash k$	0	1	2	3	4	5	6	7	8	9	10	11
0	$\underline{0}$	1	2	3	4	5	6	7	8	9	10	11
1	0	$\underline{0}$	1	2	3	4	5	7	8	9	10	11
2	0	0	$\underline{0}$	$\underline{0}$	$\underline{0}$	1	2	3	4	5	6	7
3	0	0	0	0	0	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	1	2
4	0	0	0	0	0	0	0	0	0	0	$\underline{0}$	$\underline{0}$

g(k) = μx[f(x,k)] = min$\{x \in \mathbb{N}|$ f(x,k) = 0 $\wedge \forall m < x$: f(m,k) definiert$\}$
g(k) = μx[f(x,k)] = min$\{x \in \mathbb{N}|$ f(x,k) = 0$\}$, denn f ist immer definiert.
1. Fall: k $> x^2 \rightarrow$ f(x,k) > 0
2. Fall: k = $x^2 \rightarrow$ f(x,k) = 0
3. Fall k $< x^2 \rightarrow$ f(x,k) = 0
$\Longrightarrow$ f(x,k) = 0, wenn x $\geq \sqrt{k} \Rightarrow$ min: $x = \sqrt{k}$

g(k) = $\lceil \sqrt{k} \rceil$

b)

partiell rekursiv = Turing-berechenbar = WHILE-berechenbar = GOTO-berechenbar

c)

Da das WHILE-Programm A nur natürliche Zahlen entgegennimmt, müssen wir B auf eine beliebige Weise als Zahlen repräsentieren (z.B. Binär, ASCII, codiert).

5.3 Aufgabe 3

a)

Die Menge der Java-Programme, die bei Eingabe "Hummelbummel" terminieren ✓
Die Menge der Java-Programme, die bei Eingabe "Hummelbummel" nicht terminieren X (Halteproblem)
Die Menge aller Java-Programme ✓
Die leere Menge ✓ (per Definition)
Die Menge der Java-Programme, die "Hummelbummel" ausgeben ✓
Die Menge der Java-Programme, die nicht "Hummelbummel" ausgeben X (Halteproblem)
Die Menge der Strings, die keine Java-Programme sind ✓ (Satz von Rice)

b)

spezielles Halteproblem:
HP entscheidbar? $\rightarrow$ TM M für HP
M: code(TM) $\rightarrow$ 0,1
TM M' = $M_{code(M')}$ = M mit $\{0 \rightarrow$ stoppt, $1 \rightarrow$ stoppt nicht$\}$
M'(code(M')) hält $\Leftrightarrow$ M(code(M')) = 0 $\Leftrightarrow$ code(M') $\notin$ HP $\Leftrightarrow$ $M_{code(M')}$(code(M')) hält nicht $\Leftrightarrow$
M'(code(M')) hält nicht $\lightning$

c)

Das Programm ist syntaktisch korrekt. ✓
Das Programm terminiert. X (Halteproblem)
Es gibt keine NullpointerException. X (Runtime-Analysis)
Alle Variablen werden vor ihrer jeweiligen Verwendung deklariert. ✓ (Static-Code-Analysis) (*1)
Alle Codezeilen sind erreichbar. X (Satz von Rice)

(*1) Statische Code Analyse garantiert nicht eine perfekte Detection Quote, durch zunehmend komplexere Programme sinkt die Wahrscheinlichkeit alle Fälle korrekt zu identifizieren (z.B. Reflection).